OpenSCAD
(http://www.openscad.org/)

KOMUTLARI

Dursun Murat Özden

Ankara 2013

AÇIKLAMA SATIRI

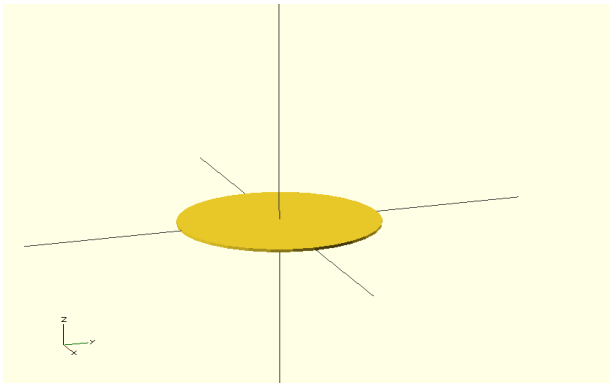
Satırın başına // konulduğu zaman program o satırdaki yazıları icra etmez, herhangi bir komut satırını iptal etmek veya açıklama yazmak için kullanılır

```
// küre yapımı
// sphere (5);
```

DAİRE (circle)

Aşağıdaki komut 40mm çapında bir daire çizer. Çap değerini verirken r=40 şeklinde yazmak sonucu değiştirmez, ancak komutun daha anlaşılır olmasını sağlar.

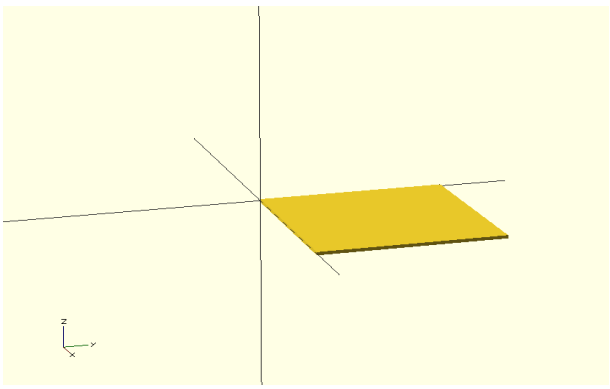
```
circle (40);

veya

circle (r=40);
```

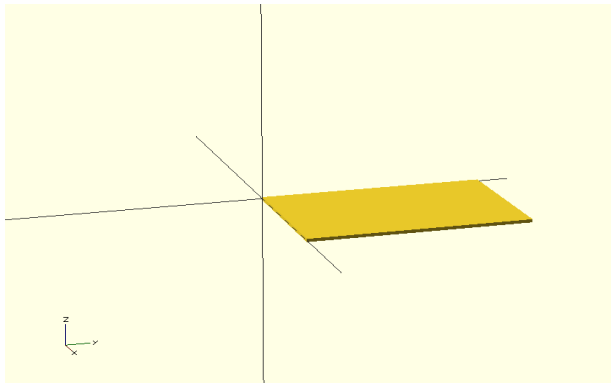


KARE (square)

Aşağıdaki komut kenarları 50mm olan bir kare çizer. Dikdörtgen çizmek için dikdörtgenin ölçülerini köşeli parantez içinde yazmak yeterli olacaktır

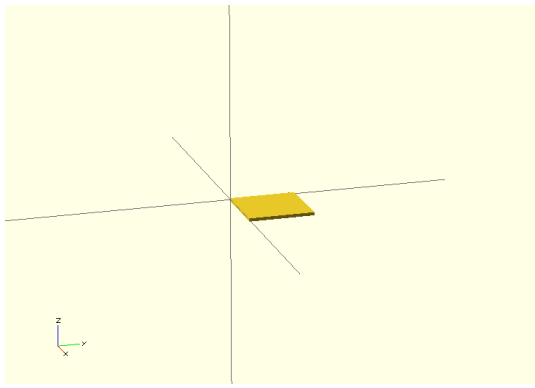
```
square (50);
```

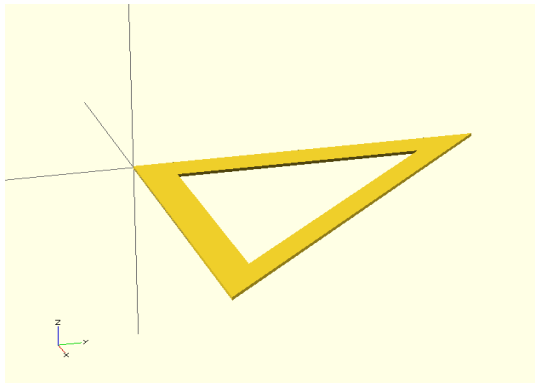


square ([40, 60]);	

POLİGON (polygon)

Poligon noktalar ve bu noktaları birleştiren çizgilerden meydana gelmektedir. Nokta gerçekte XY düzlemindeki bir koordinatı temsil etmektedir. Koordinatları X=2, Y=5 olan bir noktanın gösterme biçimi [2,5] şeklindedir.
Daha iyi anlaşılması için kenar ölçüleri 5mm olan bir kare düşünelim. Karenin bir köşesi koordinat ekseninin orijininde olsun. Bu köşenin koordinatı [0,0] olacaktır. Saat yönünde düşünürsek 5mm kenar ölçüsüne sahip kareye ait diğer köşelerin koordinatları ise [5,0], [5,5], [0,5] olacaktır. Dolayısıyla kareyi temsil eden bir poligon çizmek istersek koordinatların tamamı [0,0],[5,0],[5,5], [0,5] olacaktır.
Programın koordinatları belli bu noktalardan bir kare çizebilmesi için birinci noktadan başlayarak sırayla her nokta arasına bir çizgi çizmesi gerekir. Buna çizim yolu (path) denilmektedir ve birinci nokta 1 olarak değil 0 olarak belirtilmektedir. Kareye ait yolu tarif edebilmek için yol komutu [0,1,2,3] şeklinde olacaktır.
Poligon ise belirtilen noktalar arasında çizgiler çizmek suretiyle elde edilmektedir. Örneğin aşağıdaki komut kenarları 20mm olan bir kare oluşturmaktadır. Komuttaki köşeli parantez içindeki ilk rakam seti koordinatları, ikinci rakam seti ise yolu göstermektedir.

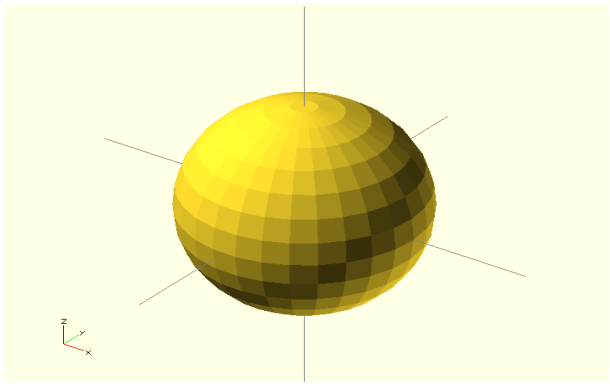
polygon([[0,0],[20,0],[20,20],[0,20]] , [[0,1,2,3]]);	

polygon(points=[[0,0],[100,0],[0,100],[10,10], [80,10],[10,80]], paths=[[0,1,2],[3,4,5]]);	

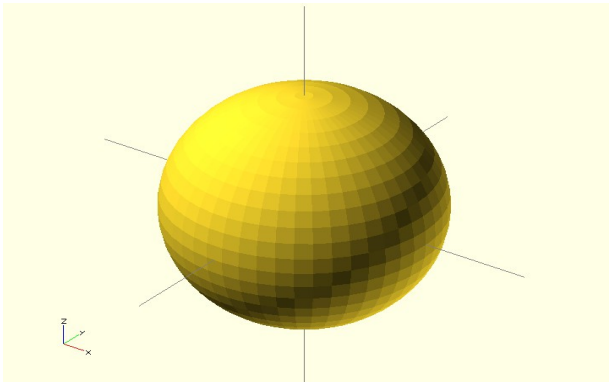
Koordinatlarını belirtmek suretiyle istenilen şekil çizilebilir.

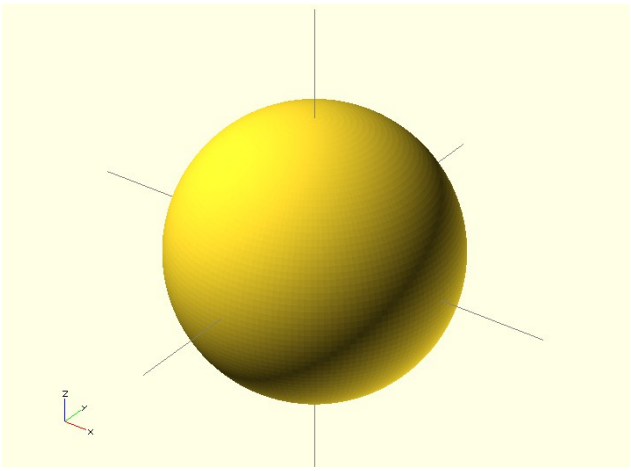
KÜRE (sphere)

Aşağıdaki komut 15 mm çapında bir küre oluşturur.

sphere (15);	

Görüldüğü gibi çözünürlük çok iyi değil, eğer çizimin çözünürlüğünü artırmak istersek komuta özel parametrelerden $fn eklememiz gerekir. $fn değerini ne kadar artırırsak çözünürlük o kadar iyi olur, ancak programın çizimi tamamlaması daha uzun sürer.

sphere (15); $fn = 50;	

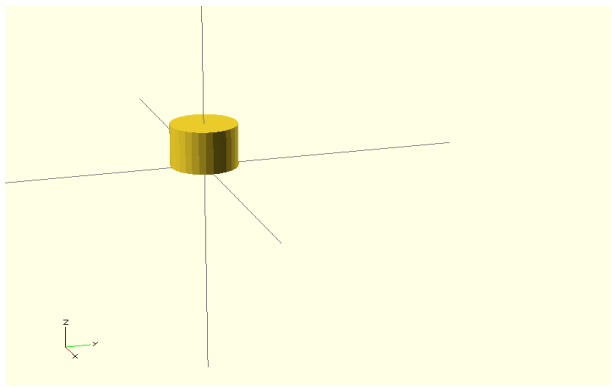
sphere (15); $fn = 150;	

$fn özel parametresi aşağıdaki komutlar da dahil bütün çizimlerde kullanılabilir.

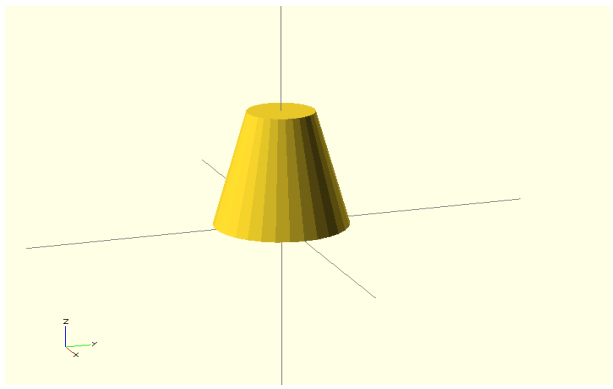
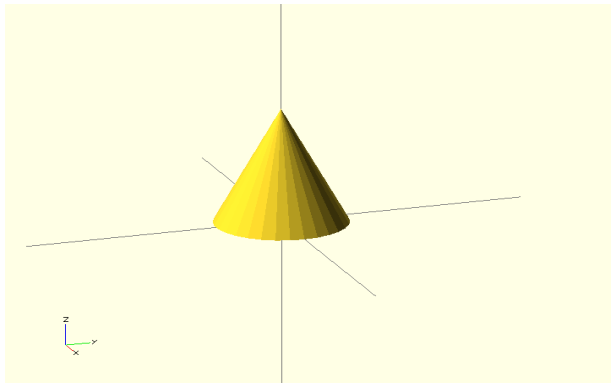
SİLİNDİR (cylinder)

Bu komut cylinder (h, r1, r2) şeklinde kullanılır. İlk terim, h, yapılacak silindirin yüksekliğini, r1, alt taban çapını, r2, üst taban çapını simgeler.

Aşağıdaki komut yüksekliği 20mm, alt taban çapı 10mm, üst taban çapı 10mm olan bir silindir oluşturur. Komutu yazarken değişkenleri de yazmak sonucu değiştirmez, ancak komut satırının daha anlaşılır olmasını sağlar. Örneğin, h=20, r1=10 ve r2=10

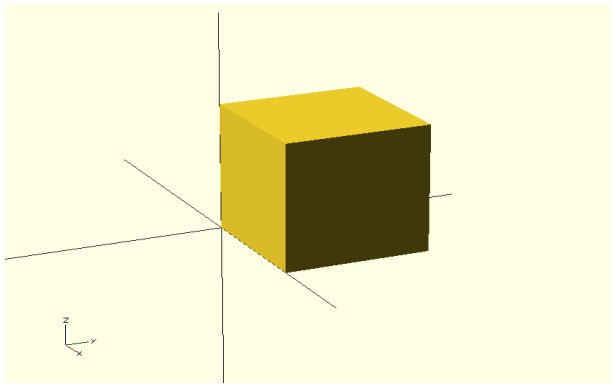
cylinder (20, 10, 10); veya cylinder (h=20, r1=10, r2=10);	

Bu komutu kullanarak taban çapları farklı şekiller oluşturmak mümkündür. Örneğin, üst taban çapını küçülterek bir kesik koni veya üst taban çapını sıfır yaparak bir koni oluşturulabilir.

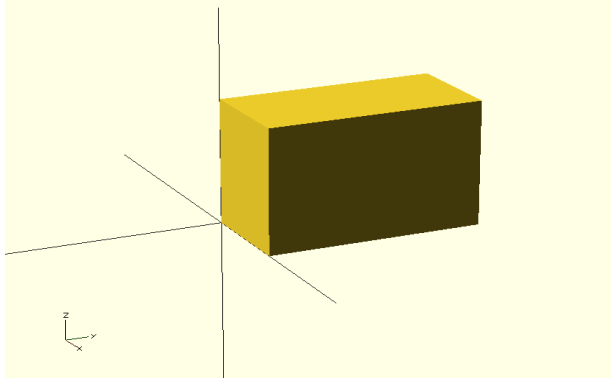
cylinder (h=40, r1=20, r2=10);	
cylinder (h=40, r1=20, r2=0);	

KÜP (cube)

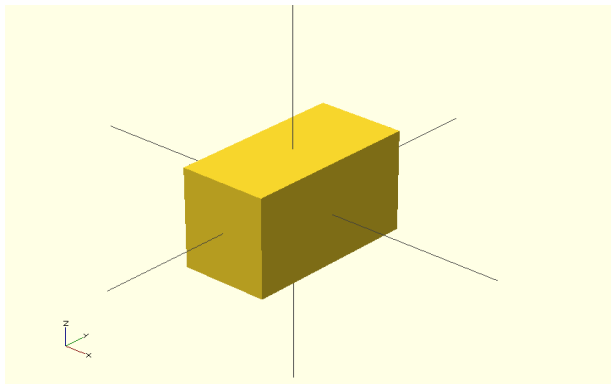
Aşağıdaki komut bir köşesi orijinde olacak şekilde kenarları 40mm olan bir küp çizmektedir. İkinci komutta basit olarak X, Y ve Z ekseni üzerinde çizilecek şeklin ölçüleri verilmektedir.

cube(40);
 	
veya
 	
cube([40,40,40]);
	

Komutta verilen üç ölçü de 40 yani eşit olduğu için çizilen şekil bir küp olmaktadır. Bu ölçüleri değiştirerek istediğimiz ölçüde dikdrötgen prizma çizilebilir.
Aşağıdaki komut eni 30mm, boyu 60mm, yüksekliği 40mm olan bir dökdörtgen prizma çizer.

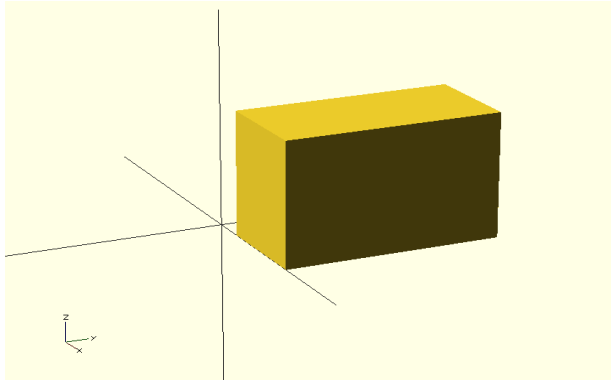
cube([30,60,40]);	

Çizdiğimiz şekli eksenlerin merkezine almak için komut satırına *center=true* ekliyoruz.

cube([30,60,40], center=true);	

KONUMLANDIRMA (translate)

Bu komut çizilen şeklin X, Y ve Z eksenlerine göre hareket ettirilmesi ve belirlenen koordinata konumlandırılması için kullanılmaktadır. Yukarıda yaptığımız dikdörtgen prizmayı örnek alırsak, orijinde olan şeklimizi eksenler üzerinde hareket ettirmek için istediğimiz konumun koordinatlarını vermeliyiz. Örneğin aşağıdaki komut prizmayı X ekseni üzerinde 20mm hareket ettirmektedir.

translate([20,0,0]) cube([30,60,40]);	

Komutta köşeli parantez içinde gösterilen ve sırasıyla X, Y ve Z ekseni değerlerini ifade eden

rakamları değiştirerek şeklimizi bu eksenler üzerinde istediğimiz noktaya konumlandırabiliriz. Aşağıda bu komuta ait değişik örnekler bulunmaktadır:

```
translate([10,20,30]) cylinder(20,5,5);
```

```
translate([10,20,30]) cube(5);
```

```
translate([10,20,30]) cylinder(20,5,10);
```

```
translate([10,20,30]) cylinder(20,5,0);
```

```
translate([10,20,30]) cube([4,8,16]);
```

DÖNDÜRME (rotate)

Bir şeklin eksenler dikkate alınarak derece cinsinde döndürülmesi için kullanılmaktadır. Komutta köşeli parantez içinde yer alan rakamlar sırasıyla X, Y ve Z eksenine göre şeklin kaç derece döndürüleceğini göstermektedir.
Önce bir dikdörtgen prizma çizelim, eni 40mm, boyu 60mm ve yüksekliği 50mm olsun

```
cube([40,60,50]);
```

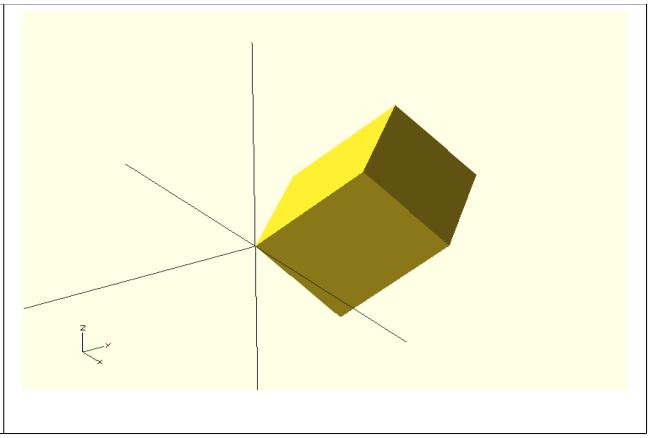
Çizdiğimiz bu prizmayı sırasıyla X, Y ve Z ekseni boyunca 45 derece döndürmek üzere komutlarımızı yazalım

```
rotate([45,0,0]) cube([40,60,50])
```

```
rotate([0,45,0]) cube([40,60,50])
```

```
rotate([0,0,45]) cube([40,60,50])
```

Aşağıdaki komut aynı dikdörtgen prizmayı Y ve Z eksenlerine göre 45 derece döndürmektedir.

`rotate([0,45,45]) cube([40,60,50]);`	

Aşağıda bu komuta ait değişik örnekler bulunmaktadır:

```
rotate([45,90,135]) sphere(5);
```

```
rotate([45,90,135]) cylinder(20,5,5);
```

```
rotate([45,90,135]) cube(5);
```

```
rotate([45,90,135]) cylinder(20,5,10);
```

```
rotate([45,90,135]) cylinder(20,5,0);
```

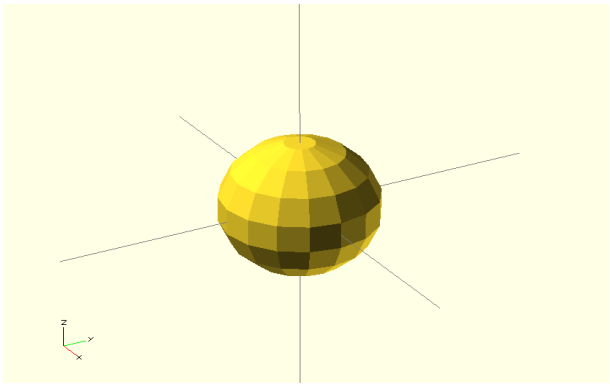
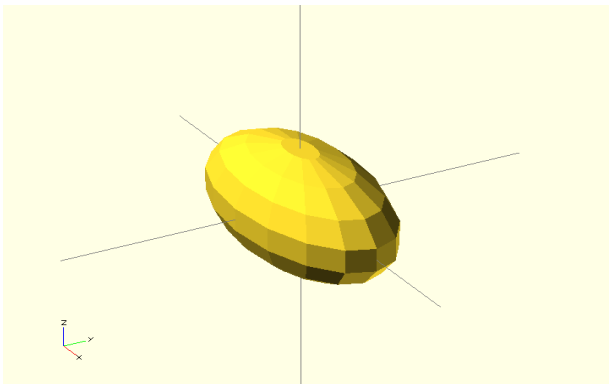
```
rotate([45,90,135]) cube([4,8,16]);
```

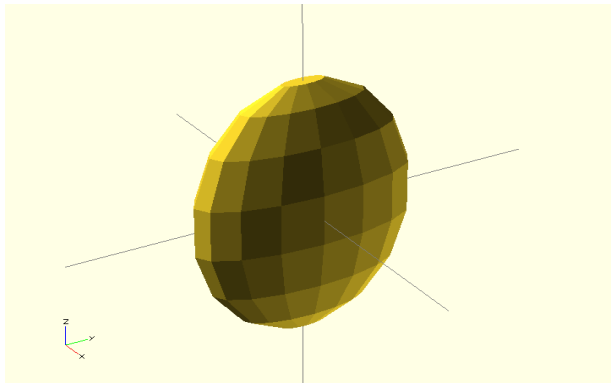
ÖLÇEKLENDİRME (scale)

Bu komut çizdiğimiz her hangi bir şekli X, Y veya Z eksenlerine göre yüzde olarak ölçeklendirmek için kullanılmaktadır. Komut satırına yazılan rakamlar şeklin hangi oranda ölçeklendirileceğini gösterir, 1 herhangi bir değişiklik yapmaz, 2 şekli iki katına çıkarır yani %200 oranında büyütür, 0.5 ise % 50 oranında küçültür.
Çapı 5 mm olan bir küre çizelim

```
sphere(5);
```

ve bu küreyi farklı eksenler üzerinde ve değişik oranlarda ölçeklendirelim:

`sphere(5);`	
`scale([2,1,1]) sphere(5);`	

scale([0.5,1.5,2]) sphere(5);	

X ekseninde % 200 büyütmek için:

scale([2,1,1]) sphere(5);

X ekseninde % 50 küçültmek için:

scale([0.5,1,1]) sphere(5);

Y ekseninde % 200 büyütmek için:

scale([1,2,1]) sphere(5);

Z ekseninde % 200 büyütmek için:

scale([1,1,2]) sphere(5);

X ekseninde % 50, Y ekseninde % 150 ve Z ekseninde % 200 ölçeklendirmek için:

scale([0.5,1.5,2]) sphere(5);

Aşağıdaki örnekler ise değişik şekilleri değişik orankarda ölçeklendirmek üzere verilmiştir.

scale([0.5,1.5,2]) sphere(5);

scale([0.5,1.5,2]) cylinder(20,5,5);

scale([0.5,1.5,2]) cube(5);

scale([0.5,1.5,2]) cylinder(20,5,10);

scale([0.5,1.5,2]) cylinder(20,5,0);

scale([0.5,1.5,2]) cube([4,8,16]);

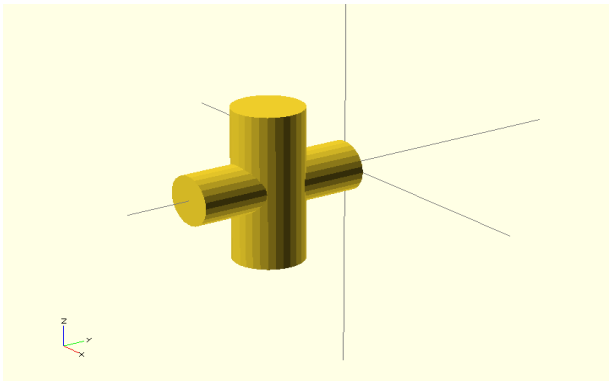
Birinci bölümde temel şekilleri oluşturmayı, bu şekilleri eksenler üzerinde hareket ettirmeyi ve ölçeklendirmeyi öğrendik. Bu bölümde yukarıda anlatılan komutları kullanarak birden çok şekil çizmeyi bu şekillere ait komutlaradaki değişkenlerle oynayarak farklı şekiller oluşturmayı göreceğiz.

Önce 50mm yüksekliğinde, alt ve üst çapları 10mm olan bir silindir çizelim, bu silindir Y ve Z eksenlerinden 25mm uzakta olsun:

```
translate([0,-25,-25]) cylinder(50,10,10);
```

Şimdi 50mm yüksekliğinde, alt ve üst çapları 10mm olan bir silindir daha çizelim ve bunu birinci silindirin içinden 90 derece açıyla geçirelim.

```
rotate([90,0,0]) cylinder(50,8,8);
```

//
translate([0,-25,-25]) cylinder(50,10,10);
rotate([90,0,0]) cylinder(50,8,8);
	

Hepsi bu kadar...

Şimdi iki farklı üç boyutlu şeklin birleştirilmesi için gereken komutlar incelenecektir.

ÇIKARMA (difference, intersection)

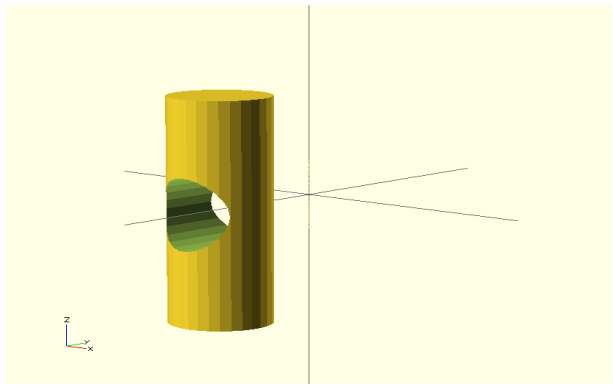
Bu komut çizilen bir şekil içinden diğer bir şekli çıkarmak üzere kullanılmaktadır. Yukarıda verilen komut satırına çıkarma komutu ekleyerek sonucu görelim;

```
difference() { translate([0,-25,-25]) cylinder(50,10,10); rotate([90,0,0]) cylinder(50,8,8); }
```

Komut satırı bize Y ve Z eksenlerine göre 25mm uzakta olmak üzere yüksekliği 5mm, alt ve üst çapları 10mm olan bir silindir çizileceğini, daha sonra yüksekliği 50mm, alt ve üst çapları 8mm olan ikinci bir silindir daha çizileceğini, ve bu ikinci silindirin X ekseni esas alınarak birinci silindir içinden çıkarılacağını söylemektedir.

Aşağıda bu komut satırı tarafından oluşturulan şekil verilmiştir.

```
difference() {
translate([0,-25,-25]) cylinder(50,10,10);
rotate([90,0,0]) cylinder(50,8,8);
}
```



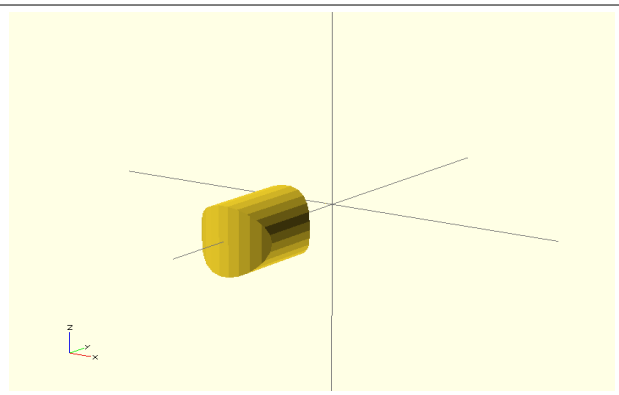
Bu komutu kullanırken çıkarma işleminin yapılabilmesi için oluşturulan iki şeklin aynı yüzey veya kenarı kullanmamasına dikkat edilmelidir.

Eğer yalnızca çıkarılan parçayı görmek istersek komut satırında *difference* yerine *intersection* komutunu kullanmamız gerekir.

```
intersection() { translate([0,-25,-25]) cylinder(50,10,10); rotate([90,0,0]) cylinder(50,8,8); }
```

Bu komut bize yukarıda anlatılanın tersine iki silindirden çıkarılan parçanın kendisini çizecektir.

```
intersection() {
translate([0,-25,-25]) cylinder(50,10,10);
rotate([90,0,0]) cylinder(50,8,8);
}
```

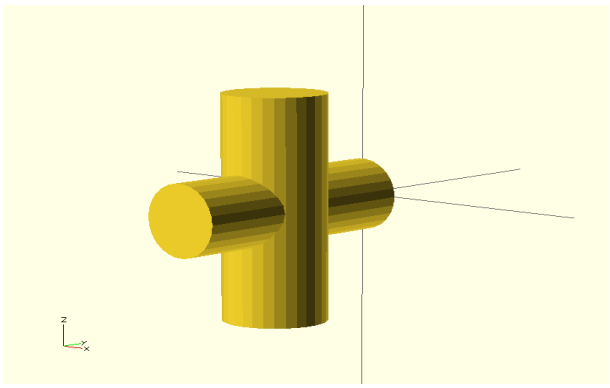


BİRLEŞTİRME (union)

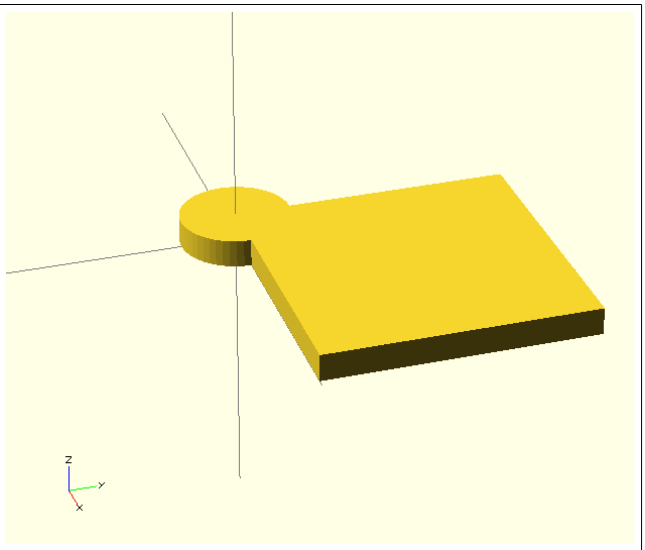
Eğer iki farklı şekli alıp bunları birleştirmek istersek *union* komutu kullanılacaktır.

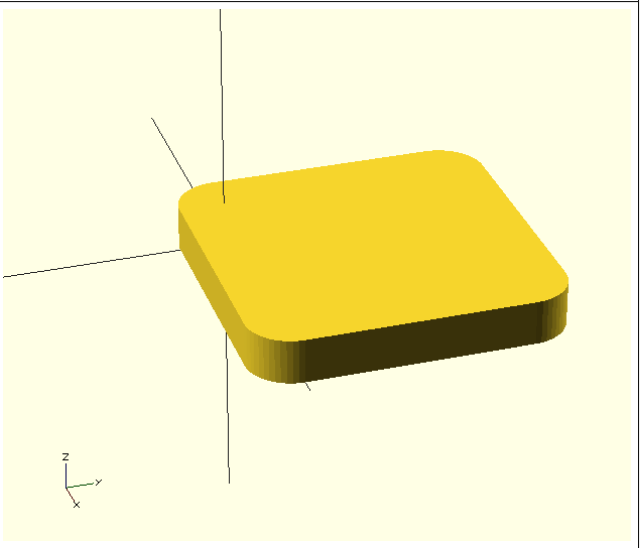
union() { translate([0,-25,-25]) cylinder(50,10,10); rotate([90,0,0]) cylinder(50,8,8); }

Bu komut iki farklı silindiri birleştirerek farklı bir parça çizmektedir.

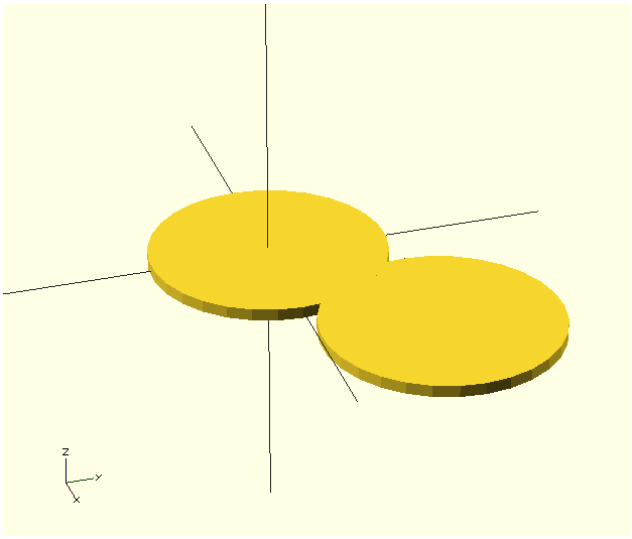
union() {
translate([0,-25,-25]) cylinder(50,10,10);
rotate([90,0,0]) cylinder(50,8,8);
}
	

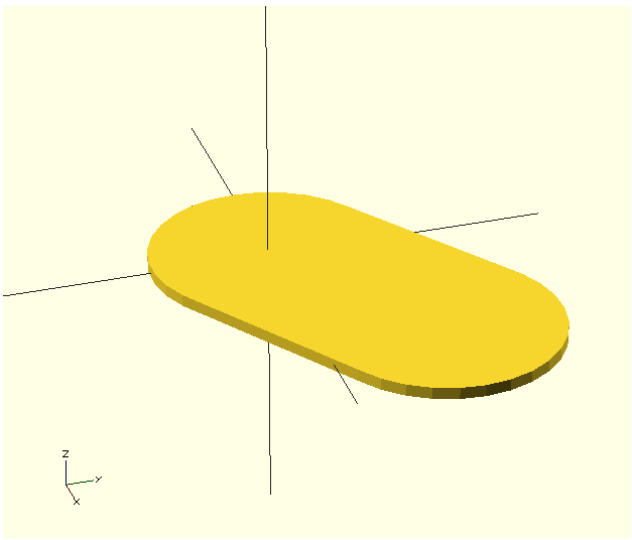
MINKOWSKI

$fn=50;
cube([10,10,1]);
cylinder(r=2,h=1);
	

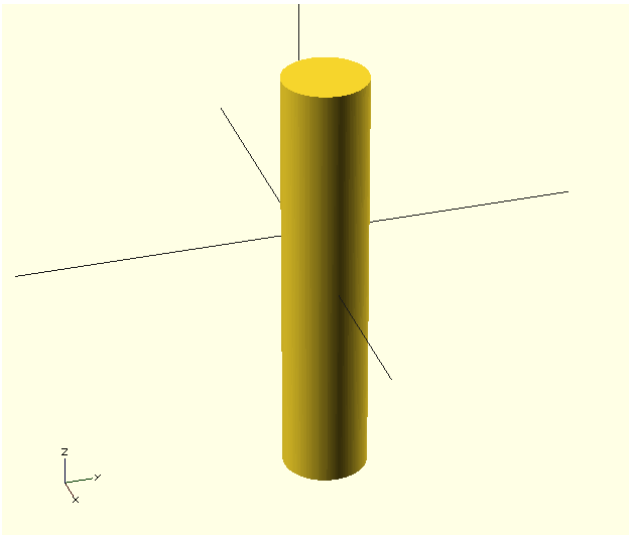
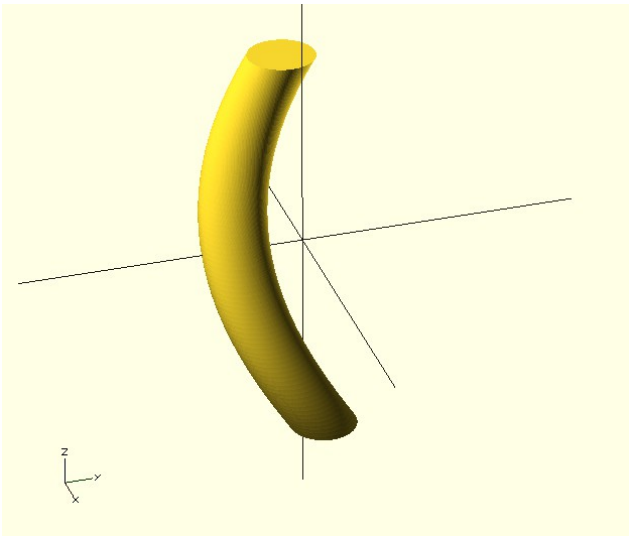
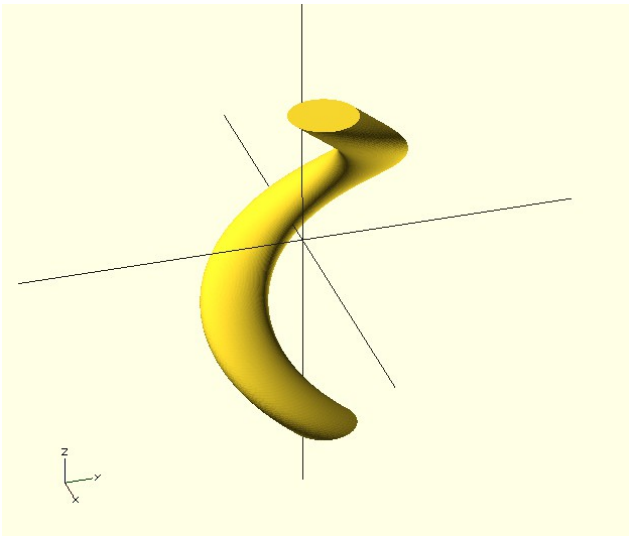
$fn=50;
minkowski()
{
cube([10,10,1]);
cylinder(r=2,h=1);
}
	

HULL

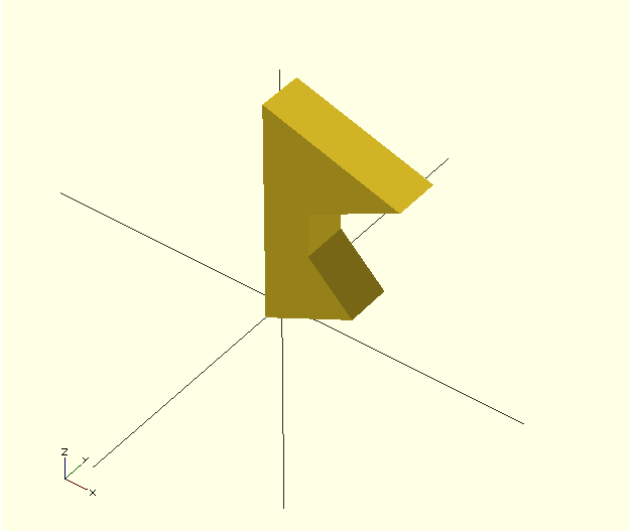
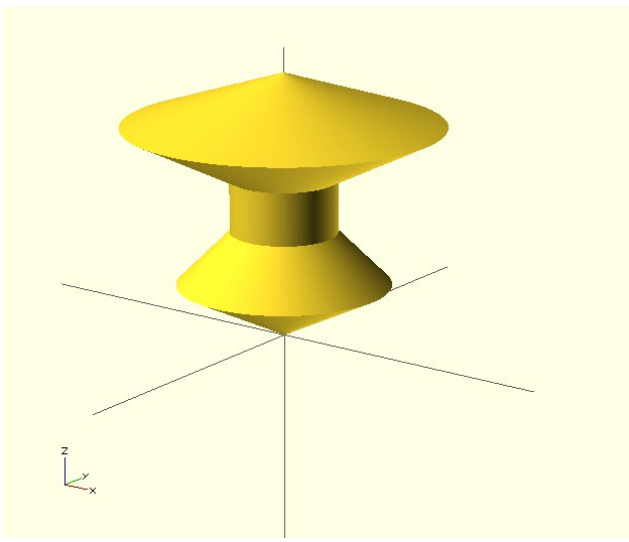
translate([15,10,0]) circle(10); circle(10);	

hull() {
translate([15,10,0]) circle(10);
circle(10);
}
	

TWIST

$fn=150; linear_extrude(height = 10, center = true, convexity = 10, twist = 0) translate([2, 0, 0]) circle(r = 1);	
$fn=150; linear_extrude(height = 10, center = true, convexity = 10, twist = 180) translate([2, 0, 0]) circle(r = 1);	
$fn=150; linear_extrude(height = 10, center = true, convexity = 10, twist = 360) translate([2, 0, 0]) circle(r = 1);	

EXTRUDING POLYGON

rotate([90,0,0]) polygon(points=[[0,0],[2,1],[1,2],[1,3],[3,4],[0,5]]);	
rotate_extrude($fn=200) polygon(points=[[0,0],[2,1],[1,2],[1,3],[3,4],[0,5]]);	

MODÜL OLUŞTURMA (modul)

Bu komutla modül oluşturup birden fazla işlemi bu modül içerisinde yapabiliriz. Örneğin eni 5mm, genişliği 10mm ve yüksekliği 15mm olan basit bir kutu yapmak için;

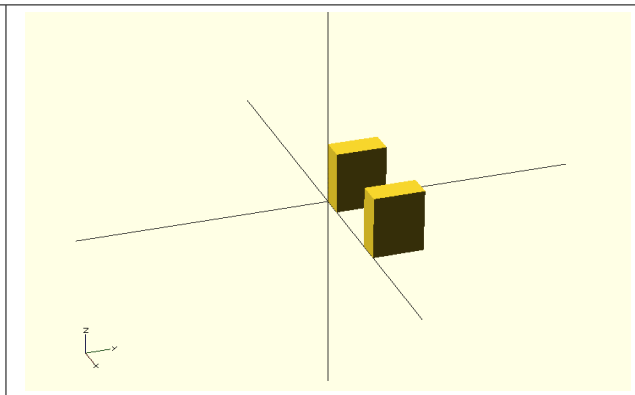
```
cube([5,10,15]);
```

komut satırını kullanıyoruz. Eğer aynı kutudan iki adet yapmak istersek bu işlemleri kutu isimli bir modül içerisinde yapabiliriz.

```
module kutu()
{
cube([5,10,15]);
}
kutu();
kutu();
```

Bu komut kümesi iki adet aynı ölçüde kutu oluşturacaktır, ancak her ikisi de aynı koordinatlarda yer alacağından tek kutu gibi görünecektir. İkinci kutuyu görebilmek için konumunu değiştirmemiz gerekir. İkinci kutuyu X ekseni boyunca 20mm uzağa konumlandırmak için *translate* komutunu kullanıyoruz.

```
module kutu()
{
cube([5,10,15]);
}
translate([20,0,0]) kutu();
kutu();
```

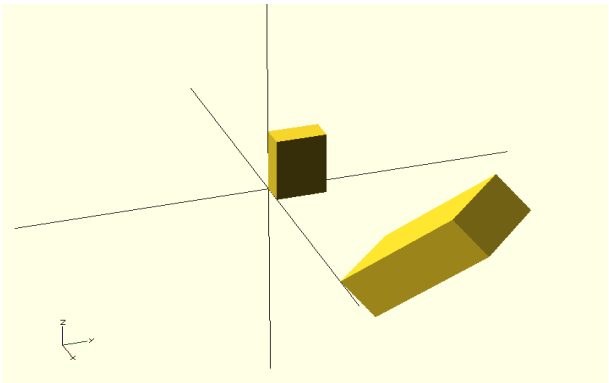


Şimdi aynı ölçülerde fakat farklı konumlarda iki adet kutu çizmiş olduk.

Modül içerisinde bu kutulara farklı işlemler yapılabilir, yerleri ve boyutları değiştirilebilir, döndürülebilir, farklı şekiller eklenebilir.

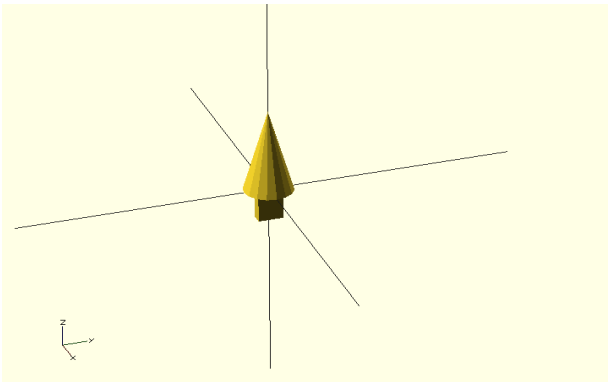
İkinci kutuyu iki katı büyütelim, Y ve Z eksenlerine göre 45 derece döndürelim.

```
module kutu()
{
cube([5,10,15]);
}
scale([2,2,2]) translate([20,0,0])
rotate([0,45,45]) kutu();
kutu();
```



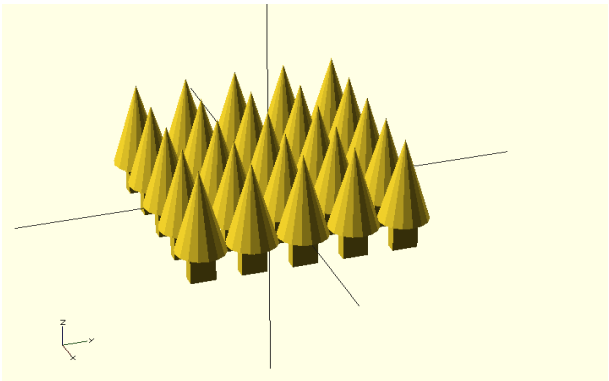
Modül içinde modül kullanarak farklı işlemler yapılabilir. Bir ağaç yapalım;

```
module yaprak()
{
cylinder(20,5,0);
}
module kutu()
{
cube([5,10,15]);
}
module agac()
{
yaprak();
scale([0.5,0.5,0.5]) translate([-2.5,-5,-15])
kutu();
}
agac();
```



Ağaç yapabildiğimize göre orman da yapabiliriz.

```
module yaprak()
{
cylinder(20,5,0);
}
module kutu()
{
cube([5,10,15]);
}
module agac()
{
yaprak();
scale([0.5,0.5,0.5]) translate([-2.5,-5,-15])
kutu();
}
module agaclar()
{
```



```
translate([20,0,0]) agac();
translate([10,0,0]) agac();
agac();
translate([-10,0,0]) agac();
translate([-20,0,0]) agac();
}
module orman()
{
translate([0,20,0]) agaclar();
translate([0,10,0]) agaclar();
agaclar();
translate([0,-10,0]) agaclar();
translate([0,-20,0]) agaclar();
};
orman();
```

MODÜLLERDE DEĞİŞKEN KULLANMA

Daha önceki bölümlerde şekil oluştururken rakamları değişkenlerle tanımlamayı görmüştük. Örneğin;

```
cylinder(h = 20, r1 = 10, r2 = 5);
```

komutu, bize 20mm yüksekliğinde bir silindirden, alt taban çapı 10mm, üst taban çapı 5mm olan bir kesik koni çizmektedir.
Aşağıdaki gibi bir modül yazarsak;

```
module kutu(en,boy,yukseklik)
{
cube([en,boy,yukseklik]);
}
kutu(5,10,15);
translate([-40,0,0]) kutu(20,25,30);
```

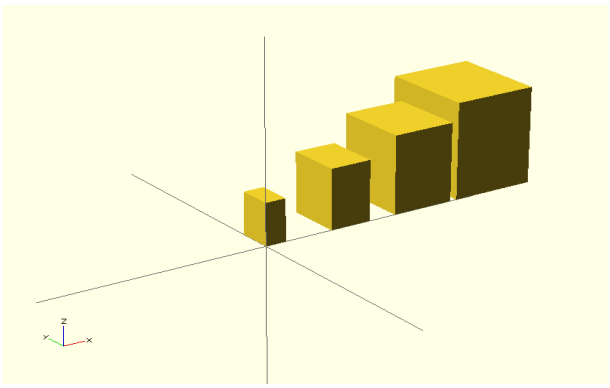
eni 5mm, boyu 10mm ve yüksekliği 15mm olan birinci kutu ile eni 20mm, boyu 25mm ve yüksekliği 30mm olan ikinci kutu çizeriz, ikinci kutu X eksenine göre merkezden -40mm uzakta olacaktır. Burada dikkat edilmesi gereken nokta modül içerisinde *cube* komutuna değişken atayarak farklı ölçülerde ve konumlarda iki adet kutu çizebilmemizdir.
Çizeceğimiz kutu ölçülerinin sadece kutunun en ölçüsüne bağlı olarak değişmesini istersek, yani her zaman kutunun boyunun eninden 2mm fazla, yüksekliğinin ise eninden 4mm fazla olmasını istersek modülü en değişkenine bağlı kurup kutuyu oluşturan *cube* komutuna en değişkenine bağlı değerler verebiliriz.

```
module kutu1(en)
{
cube([en, en + 2, en + 4]);
}
kutu1(3);
translate([10,0,0]) kutu1(6);
```

Bu komut kümesi bize eni 3mm ve 6mm olan iki kutu çizecektir. Birinci kutunun boyu 5mm,

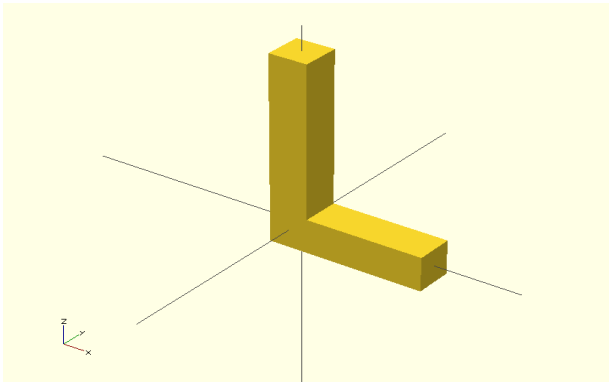
yüksekliği 7mm olacaktır. İkinci kutunun boyu 8mm, yüksekliği ise 10mm olacaktır. Yani kutu ölçüleri her seferinde belirlediğimiz en değerine bağlı olarak ve her zaman sabit artışlarla değişecektir.
Modüle istediğimiz kadar kutu komutu ekleyerek istediğimiz sayıda ve farklı ölçüde kutu çizebiliriz.

```
module kutu1(en)
{
cube([en, en + 2, en + 4]);
}
kutu1(3);
translate([10,0,0]) kutu1(6);
translate([20,0,0]) kutu1(9);
translate([30,0,0]) kutu1(12);
```



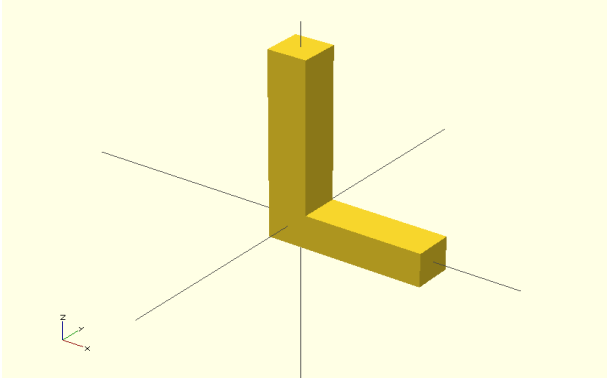
Değişken kullanma ile ilgili başka bir örenk yapalım. Aşağıdaki komut kümesi basit bir L harfi çizer.

```
// Basit L harfi çizimi, sadece rakamlarla

// Dikey parça
translate([0,0,5]) {
cube([2,2,10], center=true);
}

// Yatay parça
translate([3,0,0]) {
cube([8,2,2], center=true);
}
```



Çizdiğimiz L harfinde değişiklik yapmak istediğimizde komut satırlarına harfin ölçüsü olarak atadığımız en, boy, yükseklik değerleri ile oynamak gerekecektir. Başlangıçta değişken kullanmak çizim üzerinde daha fazla düşünmeğe ve komut satırı yazmaya yol açmasına rağmen aynı çizimi değişken kullanarak yaparsak ileride yapılacak ölçü değişiklikleri çok daha kolaya olacaktır.

```
// Basit L harfi çizimi, değişken kullanarak

// Değişkenler
dusey_parca_uzunlugu = 10;
kesit_genisligi = 0.2*dusey_parca_uzunlugu;
yatay_parca_uzunlugu =
0.8*dusey_parca_uzunlugu;

// Düşey parça
translate([0,0,dusey_parca_uzunlugu/2]) {
cube([kesit_genisligi, kesit_genisligi,
dusey_parca_uzunlugu], center=true);
}

// Yatay parça
translate([yatay_parca_uzunlugu/2-
kesit_genisligi/2,0,0]) {
cube([yatay_parca_uzunlugu, kesit_genisligi,
kesit_genisligi], center=true);
}
```

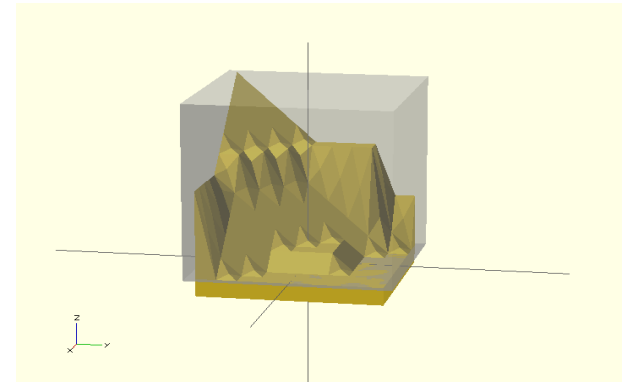


YÜZEY (surface)

Bir veri dosyasından veri okuyarak yüzey oluşturmak için aşağıdaki örnekte gösterildiği gibi surface komutu kullanılır. Kullanılan veri dosyası (surface.dat) OpenSCAD klasöründe yer almalıdır. Veri dosyası aşağıda gösterilmiştir.

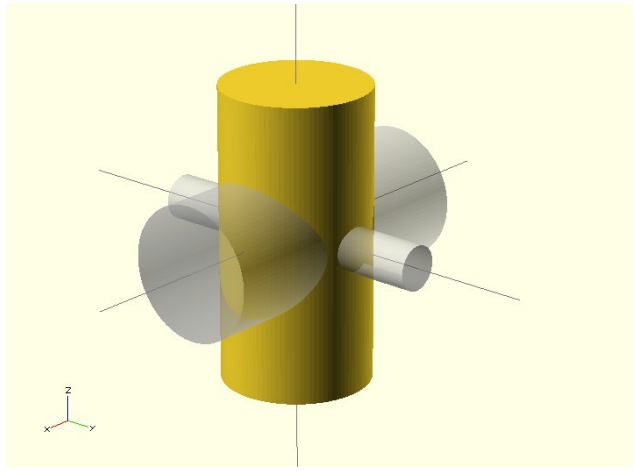
```
#surface.dat
10 9 8 7 6 5 5 5 5 5
9 8 7 6 6 4 3 2 1 0
8 7 6 6 4 3 2 1 0 0
7 6 6 4 3 2 1 0 0 0
6 6 4 3 2 1 1 0 0 0
6 6 3 2 1 1 1 0 0 0
6 6 2 1 1 1 1 0 0 0
6 6 1 0 0 0 0 0 0 0
3 1 0 0 0 0 0 0 0 0
3 0 0 0 0 0 0 0 0 0
```

```
//surface.scad
surface(file = "surface.dat", center = true,
convexity = 5);
%translate([0,0,5])cube([10,10,10], center
=true);
```

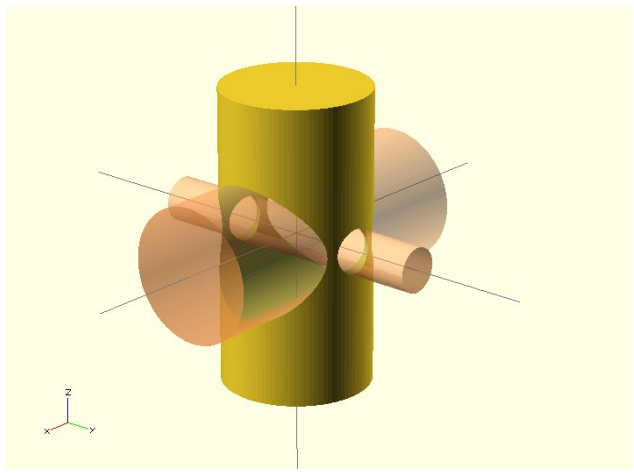


% VE # KULLANIMI

```
difference() {
// start objects
cylinder (h = 4, r=1, center = true, $fn=100);
// first object that will subtracted
% rotate ([90,0,0]) cylinder (h = 4, r=0.3, center
= true, $fn=100);
// second object that will be subtracted
% rotate ([0,90,0]) cylinder (h = 4, r=0.9, center
= true, $fn=100);
}
```



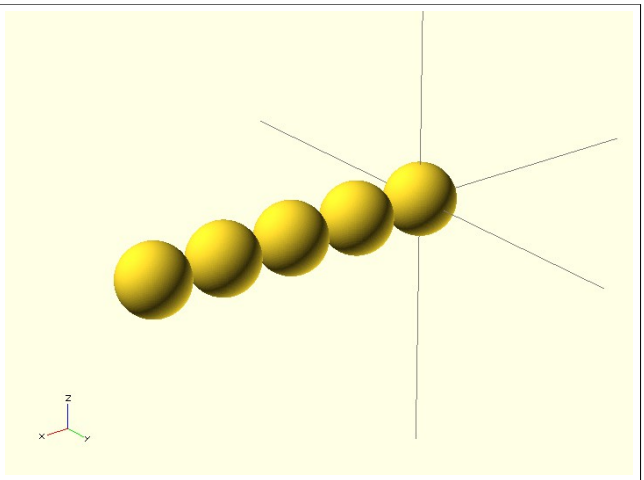
```
difference() {
// start objects
cylinder (h = 4, r=1, center = true, $fn=100);
// first object that will subtracted
# rotate ([90,0,0]) cylinder (h = 4, r=0.3, center
= true, $fn=100);
// second object that will be subtracted
# rotate ([0,90,0]) cylinder (h = 4, r=0.9, center
= true, $fn=100);
}
```



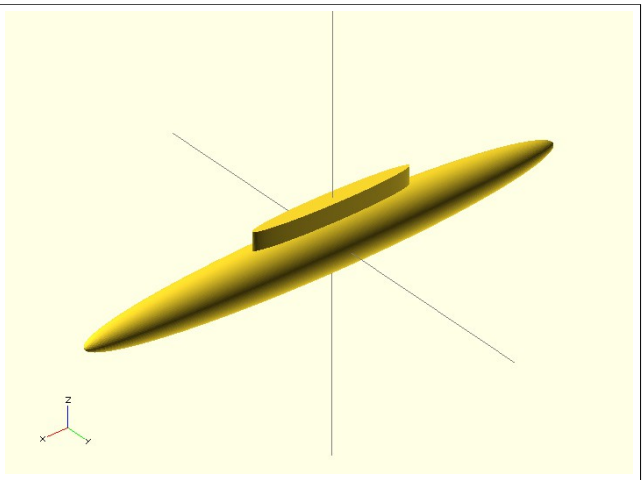
DÖNGÜ

Bir işlemi birden fazla ve belirli ölçütlere göre tekrar etmek için kullanılır. Aşağıda değişik örnekler verilmiştir.

```
$fn = 150;

module lineup(num, space) {
for (i = [0 : num-1])
translate([ space*i, 0, 0 ]) child(0);
}

lineup(5, 65) sphere(30);
```



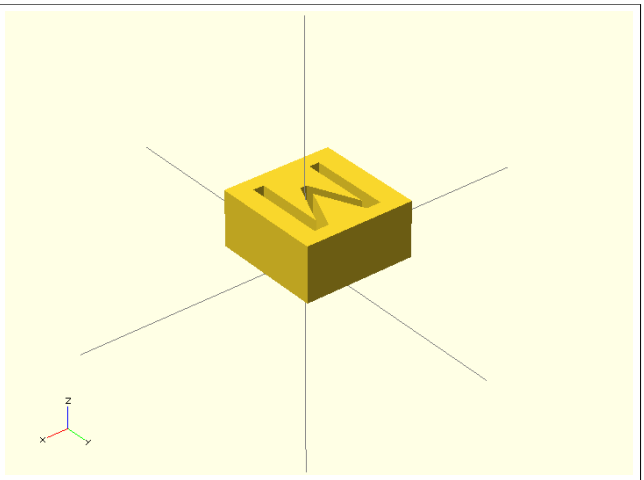
```
$fn = 150;

module elongate() {
for (i = [0 : $children-1])
scale([10 , 1, 1 ]) child(i);
}

elongate() { sphere(30); cube([10,10,10]);
cylinder(r=10,h=50); }
```



IMPORT

Klasördeki bir dosyayı içe aktarmak için kullanılır. İçe aktarılacak dosya yolu yazılmalıdır.

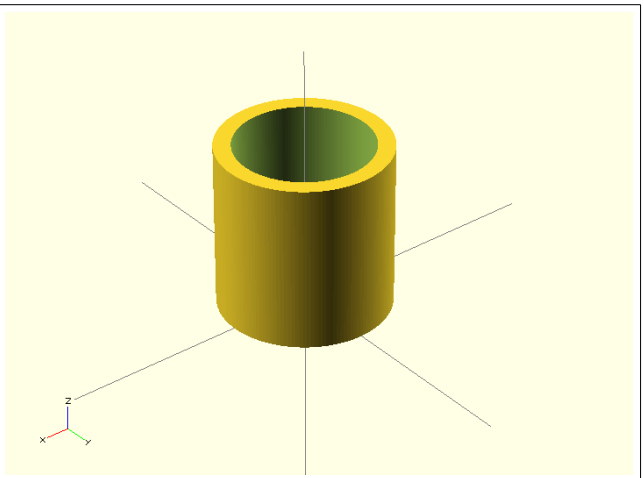
```
import("examples/example012.stl");
```



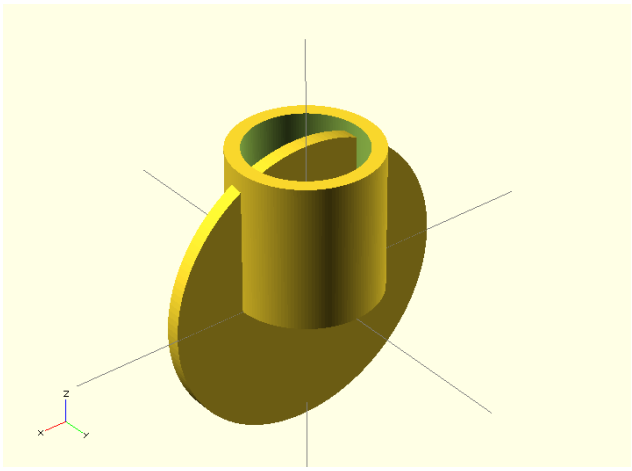
INCLUDE ve USE KULLANIMI

Önce aşağıdaki komut kümesini yazarak kaydedelim, dosya adı ring.scad olsun. Daha sonra klasördeki bu dosyayı kullanarak yeni bir kod yazalım, önce ring.scad dosyasını içe aktarmak için *include* komutunu kullanalım, daha sonra ise ring.scad dosyasını içe aktarmak için *use* komutunu kullanalım ve sonuçları karşılaştıralım.

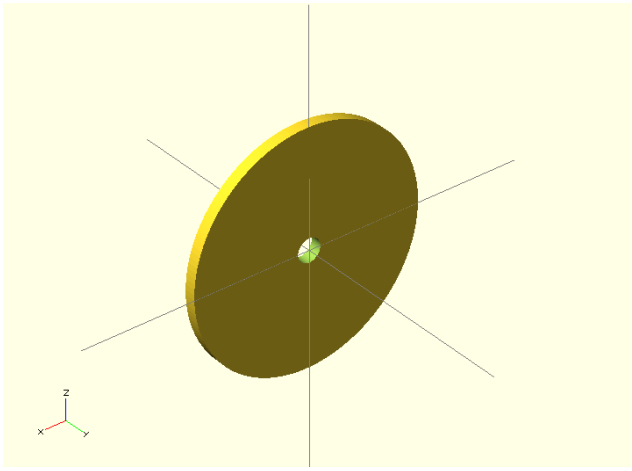
```
$fn = 150;

module ring(r1, r2, h) {
difference() {
cylinder(r = r1, h = h);
translate([ 0, 0, -1 ]) cylinder(r = r2, h = h+2);
}
}


ring(5, 4, 10);
```



```
$fn = 150;

include <ring.scad>;
rotate([90, 0, 0]) ring(10, 1, 1);
```



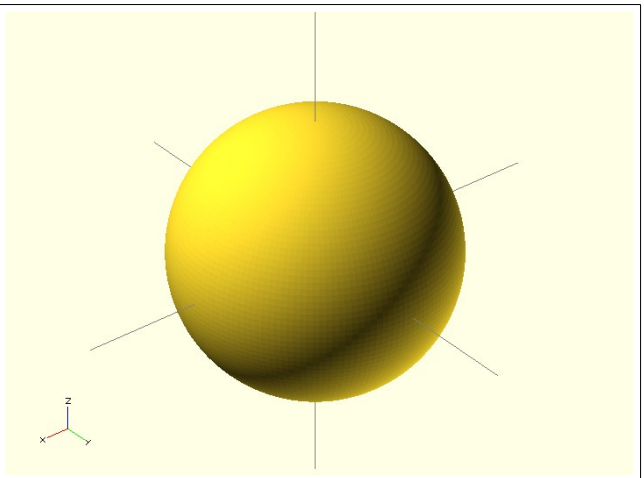
```
$fn = 150;

use <ring.scad>;
rotate([90, 0, 0]) ring(10, 1, 1);
```



KULLANICI TANIMLI FONKSİYONLAR VE ECHO KULLANIMI

Çapı 20mm olan bir küre çizeceğiz. Tanımladığımız fonskiyonda kürenin yarı çapını çapına bağlı olarak belirleyelim. Programın açıklama kısmında çap ve yarıçap değerlerinin görünmesi için *echo* komutunu kullanalım. Açıklama kısmında ECHO: "Cap ", 20, " mm, yaricap ", 10, " mm dir." bilgisi görünecektir.

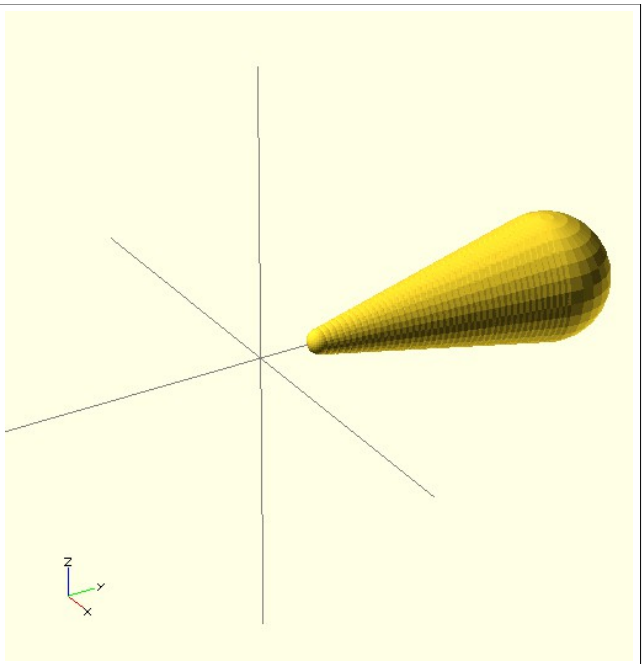
```
$fn = 150;

cap=20;
function yaricap(cap) = cap / 2;
echo("Cap ", cap, " mm, yaricap ", yaricap(cap),
" mm dir.");

sphere(cap);
```



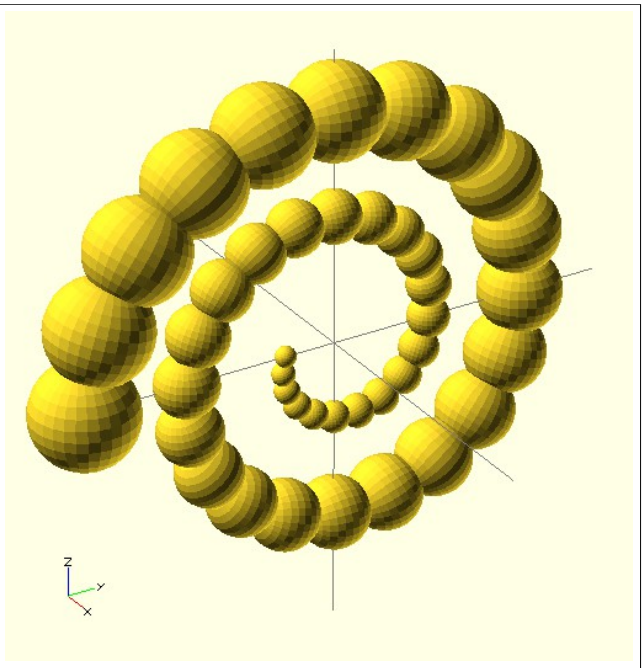
ASSIGN ve ROTATE

Değişkenlere değer atamak için *assign* komutu kullanılır. Aşağıdaki birinci örnekte çizilecek küreler için açı ve mesafe değerleri *assign* komutu ile atanmıştır. Bu örnekte artan çap değerleri ile birbirinde belirli mesafede yer alan küreler çizilecektir.
Sonraki örnek ise aynı küre grubunun belirli bir açı ile döndürülmesi görülmektedir, bu işlem *rotate* komutu ile yapılmıştır.

```
for (i = [10:50])
assign (angle = i*360/20, distance = i*10, r =
i*2) {
//rotate(angle, [1, 0, 0])
translate( [0, distance, 0] ) sphere(r = r);
}
```

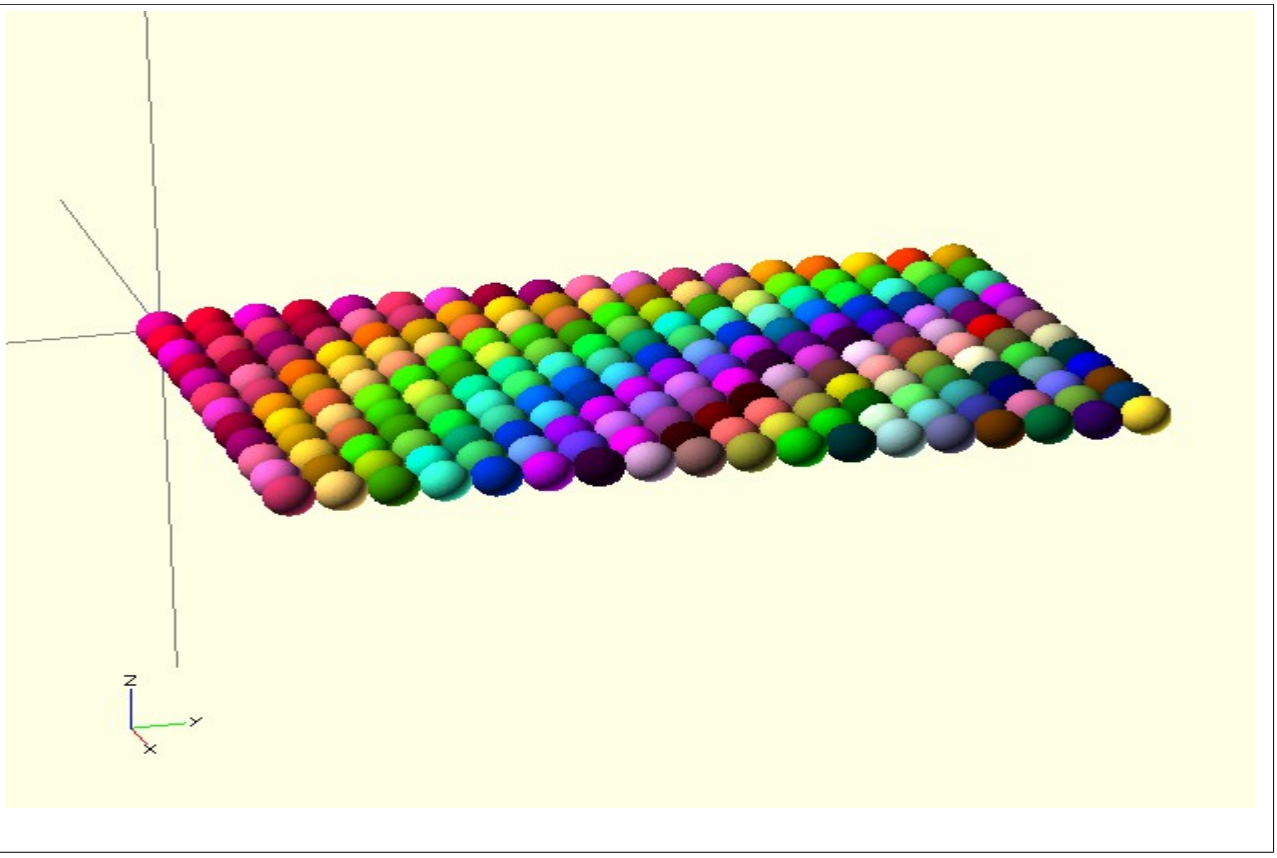


```
for (i = [10:50])
assign (angle = i*360/20, distance = i*10, r =
i*2) {
rotate(angle, [1, 0, 0])
translate( [0, distance, 0] ) sphere(r = r);
}
```



RENK

Çizimlerin renkli olarak görünmesi *color* komutu ile yapılmaktadır. Ancak OpenSCAD renk kodlarını 0 ile 1 arasında değerler olarak kabul etmektedir. Aşağıdaki örnekte 216 adet renk oluşturmak üzere 12x18 boyutunda bir renk matrisi oluşturulmuş ve bu matris kullanılarak renkli küreler çizilmiştir.



Örneğe ait kod aşağıda verilmiştir.

```
visibonecolorstest();
PPR = [255/255, 0/255, 102/255]; /* Pink Pink Red */
PPM = [255/255, 0/255, 153/255]; /* Pink Pink Magenta */
RRP = [255/255, 0/255, 51/255]; /* Red Red Pink */
MMP = [255/255, 0/255, 204/255]; /* Magenta Magenta Pink */
DRP = [204/255, 0/255, 51/255]; /* Dark Red Pink */
DMP = [204/255, 0/255, 153/255]; /* Dark Magenta Pink */
LRP = [255/255, 51/255, 102/255]; /* Light Red Pink */
LMP = [255/255, 51/255, 204/255]; /* Light Magenta Pink */
DPR = [153/255, 0/255, 51/255]; /* Dark Pink Red */
DPM = [153/255, 0/255, 102/255]; /* Dark Pink Magenta */
LPR = [255/255, 102/255, 153/255]; /* Light Pink Red */
LPM = [255/255, 102/255, 204/255]; /* Light Pink Magenta */
MPR = [204/255, 51/255, 102/255]; /* Medium Pink Red */
MPM = [204/255, 51/255, 153/255]; /* Medium Pink Magenta */
OOY = [255/255, 153/255, 0/255]; /* Orange Orange Yellow */
OOR = [255/255, 102/255, 0/255]; /* Orange Orange Red */
YYO = [255/255, 204/255, 0/255]; /* Yellow Yellow Orange */
RRO = [255/255, 51/255, 0/255]; /* Red Red Orange */
DYO = [204/255, 153/255, 0/255]; /* Dark Yellow Orange */
DRO = [204/255, 51/255, 0/255]; /* Dark Red Orange */
LYO = [255/255, 204/255, 51/255]; /* Light Yellow Orange */
LRO = [255/255, 102/255, 51/255]; /* Light Red Orange */
DOY = [153/255, 102/255, 0/255]; /* Dark Orange Yellow */
DOR = [153/255, 51/255, 0/255]; /* Dark Orange Red */
LOY = [255/255, 204/255, 102/255]; /* Light Orange Yellow */
LOR = [255/255, 153/255, 102/255]; /* Light Orange Red */
MOY = [204/255, 153/255, 51/255]; /* Medium Orange Yellow */
MOR = [204/255, 102/255, 51/255]; /* Medium Orange Red */
SSG = [102/255, 255/255, 0/255]; /* Spring Spring Green */
SSY = [153/255, 255/255, 0/255]; /* Spring Spring Yellow */
GGS = [ 51/255, 255/255, 0/255]; /* Green Green Spring */
YYS = [204/255, 255/255, 0/255]; /* Yellow Yellow Spring */
DGS = [ 51/255, 204/255, 0/255]; /* Dark Green Spring */
DYS = [153/255, 204/255, 0/255]; /* Dark Yellow Spring */
LGS = [102/255, 255/255, 51/255]; /* Light Green Spring */
LYS = [204/255, 255/255, 51/255]; /* Light Yellow Spring */
DSG = [ 51/255, 153/255, 0/255]; /* Dark Spring Green */
DSY = [102/255, 153/255, 0/255]; /* Dark Spring Yellow */
LSG = [153/255, 255/255, 102/255]; /* Light Spring Green */
LSY = [204/255, 255/255, 102/255]; /* Light Spring Yellow */
MSG = [102/255, 204/255, 51/255]; /* Medium Spring Green */
MSY = [153/255, 204/255, 51/255]; /* Medium Spring Yellow */
TTC = [ 0/255, 255/255, 153/255]; /* Teal Teal Cyan */
TTG = [ 0/255, 255/255, 102/255]; /* Teal Teal Green */
CCT = [ 0/255, 255/255, 204/255]; /* Cyan Cyan Teal */
GGT = [ 0/255, 255/255, 51/255]; /* Green Green Teal */
DCT = [ 0/255, 204/255, 153/255]; /* Dark Cyan Teal */
DGT = [ 0/255, 204/255, 51/255]; /* Dark Green Teal */
LCT = [ 51/255, 255/255, 204/255]; /* Light Cyan Teal */
```

LGT = [51/255, 255/255, 102/255]; /* Light Green Teal */
DTC = [0/255, 153/255, 102/255]; /* Dark Teal Cyan */
DTG = [0/255, 153/255, 51/255]; /* Dark Teal Green */
LTC = [102/255, 255/255, 204/255]; /* Light Teal Cyan */
LTG = [102/255, 255/255, 153/255]; /* Light Teal Green */
MTC = [51/255, 204/255, 153/255]; /* Medium Teal Cyan */
MTG = [51/255, 204/255, 102/255]; /* Medium Teal Green */
AAB = [0/255, 102/255, 255/255]; /* Azure Azure Blue */
AAC = [0/255, 153/255, 255/255]; /* Azure Azure Cyan */
BBA = [0/255, 51/255, 255/255]; /* Blue Blue Azure */
CCA = [0/255, 204/255, 255/255]; /* Cyan Cyan Azure */
DBA = [0/255, 51/255, 204/255]; /* Dark Blue Azure */
DCA = [0/255, 153/255, 204/255]; /* Dark Cyan Azure */
LBA = [51/255, 102/255, 255/255]; /* Light Blue Azure */
LCA = [51/255, 204/255, 255/255]; /* Light Cyan Azure */
DAB = [0/255, 51/255, 153/255]; /* Dark Azure Blue */
DAC = [0/255, 102/255, 153/255]; /* Dark Azure Cyan */
LAB = [102/255, 153/255, 255/255]; /* Light Azure Blue */
LAC = [102/255, 204/255, 255/255]; /* Light Azure Cyan */
MAB = [51/255, 102/255, 204/255]; /* Medium Azure Blue */
MAC = [51/255, 153/255, 204/255]; /* Medium Azure Cyan */
VVM = [153/255, 0/255, 255/255]; /* Violet Violet Magenta */
VVB = [102/255, 0/255, 255/255]; /* Violet Violet Blue */
MMV = [204/255, 0/255, 255/255]; /* Magenta Magenta Violet */
BBV = [51/255, 0/255, 255/255]; /* Blue Blue Violet */
DMV = [153/255, 0/255, 204/255]; /* Dark Magenta Violet */
DBV = [51/255, 0/255, 204/255]; /* Dark Blue Violet */
LMV = [204/255, 51/255, 255/255]; /* Light Magenta Violet */
LBV = [102/255, 51/255, 255/255]; /* Light Blue Violet */
DVM = [102/255, 0/255, 153/255]; /* Dark Violet Magenta */
DVB = [51/255, 0/255, 153/255]; /* Dark Violet Blue */
LVM = [204/255, 102/255, 255/255]; /* Light Violet Magenta */
LVB = [153/255, 102/255, 255/255]; /* Light Violet Blue */
MVM = [153/255, 51/255, 204/255]; /* Medium Violet Magenta */
MVB = [102/255, 51/255, 204/255]; /* Medium Violet Blue */
OWM = [51/255, 0/255, 51/255]; /* Obscure Weak Magenta */
ODM = [102/255, 0/255, 102/255]; /* Obscure Dull Magenta */
DFM = [153/255, 0/255, 153/255]; /* Dark Faded Magenta */
DHM = [204/255, 0/255, 204/255]; /* Dark Hard Magenta */
M = [255/255, 0/255, 255/255]; /* Magenta */
DWM = [102/255, 51/255, 102/255]; /* Dark Weak Magenta */
DDM = [153/255, 51/255, 153/255]; /* Dark Dull Magenta */
MFM = [204/255, 51/255, 204/255]; /* Medium Faded Magenta */
LHM = [255/255, 51/255, 255/255]; /* Light Hard Magenta */
MWM = [153/255, 102/255, 153/255]; /* Medium Weak Magenta */
LDM = [204/255, 102/255, 204/255]; /* Light Dull Magenta */
LFM = [255/255, 102/255, 255/255]; /* Light Faded Magenta */
LWM = [204/255, 153/255, 204/255]; /* Light Weak Magenta */
PDM = [255/255, 153/255, 255/255]; /* Pale Dull Magenta */
PWM = [255/255, 204/255, 255/255]; /* Pale Weak Magenta */
OWR = [51/255, 0/255, 0/255]; /* Obscure Weak Red */
ODR = [102/255, 0/255, 0/255]; /* Obscure Dull Red */

DFR = [153/255, 0/255, 0/255]; /* Dark Faded Red */
DHR = [204/255, 0/255, 0/255]; /* Dark Hard Red */
R = [255/255, 0/255, 0/255]; /* Red */
DWR = [102/255, 51/255, 51/255]; /* Dark Weak Red */
DDR = [153/255, 51/255, 51/255]; /* Dark Dull Red */
MFR = [204/255, 51/255, 51/255]; /* Medium Faded Red */
LHR = [255/255, 51/255, 51/255]; /* Light Hard Red */
MWR = [153/255, 102/255, 102/255]; /* Medium Weak Red */
LDR = [204/255, 102/255, 102/255]; /* Light Dull Red */
LFR = [255/255, 102/255, 102/255]; /* Light Faded Red */
LWR = [204/255, 153/255, 153/255]; /* Light Weak Red */
PDR = [255/255, 153/255, 153/255]; /* Pale Dull Red */
PWR = [255/255, 204/255, 204/255]; /* Pale Weak Red */
OWY = [51/255, 51/255, 0/255]; /* Obscure Weak Yellow */
ODY = [102/255, 102/255, 0/255]; /* Obscure Dull Yellow */
DFY = [153/255, 153/255, 0/255]; /* Dark Faded Yellow */
DHY = [204/255, 204/255, 0/255]; /* Dark Hard Yellow */
Y = [255/255, 255/255, 0/255]; /* Yellow */
DWY = [102/255, 102/255, 51/255]; /* Dark Weak Yellow */
DDY = [153/255, 153/255, 51/255]; /* Dark Dull Yellow */
MFY = [204/255, 204/255, 51/255]; /* Medium Faded Yellow */
LHY = [255/255, 255/255, 51/255]; /* Light Hard Yellow */
MWY = [153/255, 153/255, 102/255]; /* Medium Weak Yellow */
LDY = [204/255, 204/255, 102/255]; /* Light Dull Yellow */
LFY = [255/255, 255/255, 102/255]; /* Light Faded Yellow */
LWY = [204/255, 204/255, 153/255]; /* Light Weak Yellow */
PDY = [255/255, 255/255, 153/255]; /* Pale Dull Yellow */
PWY = [255/255, 255/255, 204/255]; /* Pale Weak Yellow */
OWG = [0/255, 51/255, 0/255]; /* Obscure Weak Green */
ODG = [0/255, 102/255, 0/255]; /* Obscure Dull Green */
DFG = [0/255, 153/255, 0/255]; /* Dark Faded Green */
DHG = [0/255, 204/255, 0/255]; /* Dark Hard Green */
G = [0/255, 255/255, 0/255]; /* Green */
DWG = [51/255, 102/255, 51/255]; /* Dark Weak Green */
DDG = [51/255, 153/255, 51/255]; /* Dark Dull Green */
MFG = [51/255, 204/255, 51/255]; /* Medium Faded Green */
LHG = [51/255, 255/255, 51/255]; /* Light Hard Green */
MWG = [102/255, 153/255, 102/255]; /* Medium Weak Green */
LDG = [102/255, 204/255, 102/255]; /* Light Dull Green */
LFG = [102/255, 255/255, 102/255]; /* Light Faded Green */
LWG = [153/255, 204/255, 153/255]; /* Light Weak Green */
PDG = [153/255, 255/255, 153/255]; /* Pale Dull Green */
PWG = [204/255, 255/255, 204/255]; /* Pale Weak Green */
OWC = [0/255, 51/255, 51/255]; /* Obscure Weak Cyan */
ODC = [0/255, 102/255, 102/255]; /* Obscure Dull Cyan */
DFC = [0/255, 153/255, 153/255]; /* Dark Faded Cyan */
DHC = [0/255, 204/255, 204/255]; /* Dark Hard Cyan */
C = [0/255, 255/255, 255/255]; /* Cyan */
DWC = [51/255, 102/255, 102/255]; /* Dark Weak Cyan */
DDC = [51/255, 153/255, 153/255]; /* Dark Dull Cyan */
MFC = [51/255, 204/255, 204/255]; /* Medium Faded Cyan */
LHC = [51/255, 255/255, 255/255]; /* Light Hard Cyan */

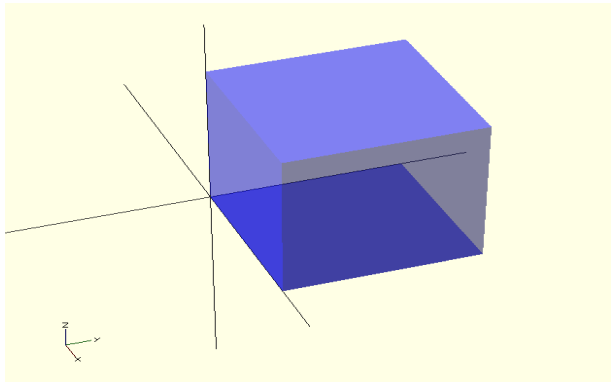
MWC = [102/255, 153/255, 153/255]; /* Medium Weak Cyan */
LDC = [102/255, 204/255, 204/255]; /* Light Dull Cyan */
LFC = [102/255, 255/255, 255/255]; /* Light Faded Cyan */
LWC = [153/255, 204/255, 204/255]; /* Light Weak Cyan */
PDC = [153/255, 255/255, 255/255]; /* Pale Dull Cyan */
PWC = [204/255, 255/255, 255/255]; /* Pale Weak Cyan */
OWB = [0/255, 0/255, 51/255]; /* Obscure Weak Blue */
ODB = [0/255, 0/255, 102/255]; /* Obscure Dull Blue */
DFB = [0/255, 0/255, 153/255]; /* Dark Faded Blue */
DHB = [0/255, 0/255, 204/255]; /* Dark Hard Blue */
B = [0/255, 0/255, 255/255]; /* Blue */
DWB = [51/255, 51/255, 102/255]; /* Dark Weak Blue */
DDB = [51/255, 51/255, 153/255]; /* Dark Dull Blue */
MFB = [51/255, 51/255, 204/255]; /* Medium Faded Blue */
LHB = [51/255, 51/255, 255/255]; /* Light Hard Blue */
MWB = [102/255, 102/255, 153/255]; /* Medium Weak Blue */
LDB = [102/255, 102/255, 204/255]; /* Light Dull Blue */
LFB = [102/255, 102/255, 255/255]; /* Light Faded Blue */
LWB = [153/255, 153/255, 204/255]; /* Light Weak Blue */
PDB = [153/255, 153/255, 255/255]; /* Pale Dull Blue */
PWB = [204/255, 204/255, 255/255]; /* Pale Weak Blue */
ODP = [102/255, 0/255, 51/255]; /* Obscure Dull Pink */
DDP = [153/255, 51/255, 102/255]; /* Dark Dull Pink */
LDP = [204/255, 102/255, 153/255]; /* Light Dull Pink */
PDP = [255/255, 153/255, 204/255]; /* Pale Dull Pink */
DHP = [204/255, 0/255, 102/255]; /* Dark Hard Pink */
LHP = [255/255, 51/255, 153/255]; /* Light Hard Pink */
ODO = [102/255, 51/255, 0/255]; /* Obscure Dull Orange */
DDO = [153/255, 102/255, 51/255]; /* Dark Dull Orange */
LDO = [204/255, 153/255, 102/255]; /* Light Dull Orange */
PDO = [255/255, 204/255, 153/255]; /* Pale Dull Orange */
DHO = [204/255, 102/255, 0/255]; /* Dark Hard Orange */
LHO = [255/255, 153/255, 51/255]; /* Light Hard Orange */
ODS = [51/255, 102/255, 0/255]; /* Obscure Dull Spring */
DDS = [102/255, 153/255, 51/255]; /* Dark Dull Spring */
LDS = [153/255, 204/255, 102/255]; /* Light Dull Spring */
PDS = [204/255, 255/255, 153/255]; /* Pale Dull Spring */
DHS = [102/255, 204/255, 0/255]; /* Dark Hard Spring */
LHS = [153/255, 255/255, 51/255]; /* Light Hard Spring */
ODT = [0/255, 102/255, 51/255]; /* Obscure Dull Teal */
DDT = [51/255, 153/255, 102/255]; /* Dark Dull Teal */
LDT = [102/255, 204/255, 153/255]; /* Light Dull Teal */
PDT = [153/255, 255/255, 204/255]; /* Pale Dull Teal */
DHT = [0/255, 204/255, 102/255]; /* Dark Hard Teal */
LHT = [51/255, 255/255, 153/255]; /* Light Hard Teal */
ODA = [0/255, 51/255, 102/255]; /* Obscure Dull Azure */
DDA = [51/255, 102/255, 153/255]; /* Dark Dull Azure */
LDA = [102/255, 153/255, 204/255]; /* Light Dull Azure */
PDA = [153/255, 204/255, 255/255]; /* Pale Dull Azure */
DHA = [0/255, 102/255, 204/255]; /* Dark Hard Azure */
LHA = [51/255, 153/255, 255/255]; /* Light Hard Azure */
ODV = [51/255, 0/255, 102/255]; /* Obscure Dull Violet */

```
DDV = [102/255, 51/255, 153/255]; /* Dark Dull Violet */
LDV = [153/255, 102/255, 204/255]; /* Light Dull Violet */
PDV = [204/255, 153/255, 255/255]; /* Pale Dull Violet */
DHV = [102/255, 0/255, 204/255]; /* Dark Hard Violet */
LHV = [153/255, 51/255, 255/255]; /* Light Hard Violet */
K = [ 0/255, 0/255, 0/255]; /* Black */
OG = [ 51/255, 51/255, 51/255]; /* Obscure Gray */
DG = [102/255, 102/255, 102/255]; /* Dark Gray */
LG = [153/255, 153/255, 153/255]; /* Light Gray */
PG = [204/255, 204/255, 204/255]; /* Pale Gray */
W = [255/255, 255/255, 255/255]; /* White */

module visibonecolorstest()
{
/*
216 colors: 12 x 18 matrix
*/
colors =
[PPR, PPM, RRP, MMP, DRP, DMP, LRP, LMP, DPR, DPM, LPR, LPM,
MPR, MPM, OOY, OOR, YYO, RRO, DYO, DRO, LYO, LRO, DOY, DOR,
LOY, LOR, MOY, MOR, SSG, SSY, GGS, YYS, DGS, DYS, LGS, LYS,
DSG, DSY, LSG, LSY, MSG, MSY, TTC, TTG, CCT, GGT, DCT, DGT,
LCT, LGT, DTC, DTG, LTC, LTG, MTC, MTG, AAB, AAC, BBA, CCA,
DBA, DCA, LBA, LCA, DAB, DAC, LAB, LAC, MAB, MAC, VVM, VVB,
MMV, BBV, DMV, DBV, LMV, LBV, DVM, DVB, LVM, LVB, MVM, MVB,
OWM, ODM, DFM, DHM, M, DWM, DDM, MFM, LHM, MWM, LDM, LFM,
LWM, PDM, PWM, OWR, ODR, DFR, DHR, R, DWR, DDR, MFR, LHR,
MWR, LDR, LFR, LWR, PDR, PWR, OWY, ODY, DFY, DHY, Y, DWY,
DDY, MFY, LHY, MWY, LDY, LFY, LWY, PDY, PWY, OWG, ODG, DFG,
DHG, G, DWG, DDG, MFG, LHG, MWG, LDG, LFG, LWG, PDG, PWG,
OWC, ODC, DFC, DHC, C, DWC, DDC, MFC, LHC, MWC, LDC, LFC,
LWC, PDC, PWC, OWB, ODB, DFB, DHB, B, DWB, DDB, MFB, LHB,
MWB, LDB, LFB, LWB, PDB, PWB, ODP, DDP, LDP, PDP, DHP, LHP,
ODO, DDO, LDO, PDO, DHO, LHO, ODS, DDS, LDS, PDS, DHS, LHS,
ODT, DDT, LDT, PDT, DHT, LHT, ODA, DDA, LDA, PDA, DHA, LHA,
ODV, DDV, LDV, PDV, DHV, LHV, K, OG, DG, LG, PG, W];

$fn = 100;
for (i=[1:12])
{
for (j=[1:18])
{
color(colors[i*j]) translate([(i-1)*4,(j-1)*4,0]) sphere(2);
}
}
}
```

Çizimleri F5 (önizleme) modunda renkli olarak görmek için ayrıca OpenSCAD tarafından tanınan renk isimlerini de kullanabiliriz.

Örneğin aşağıdaki kod mavi renkte bir küp çizecektir.

```
color("Blue",0.5) cube(5);
```



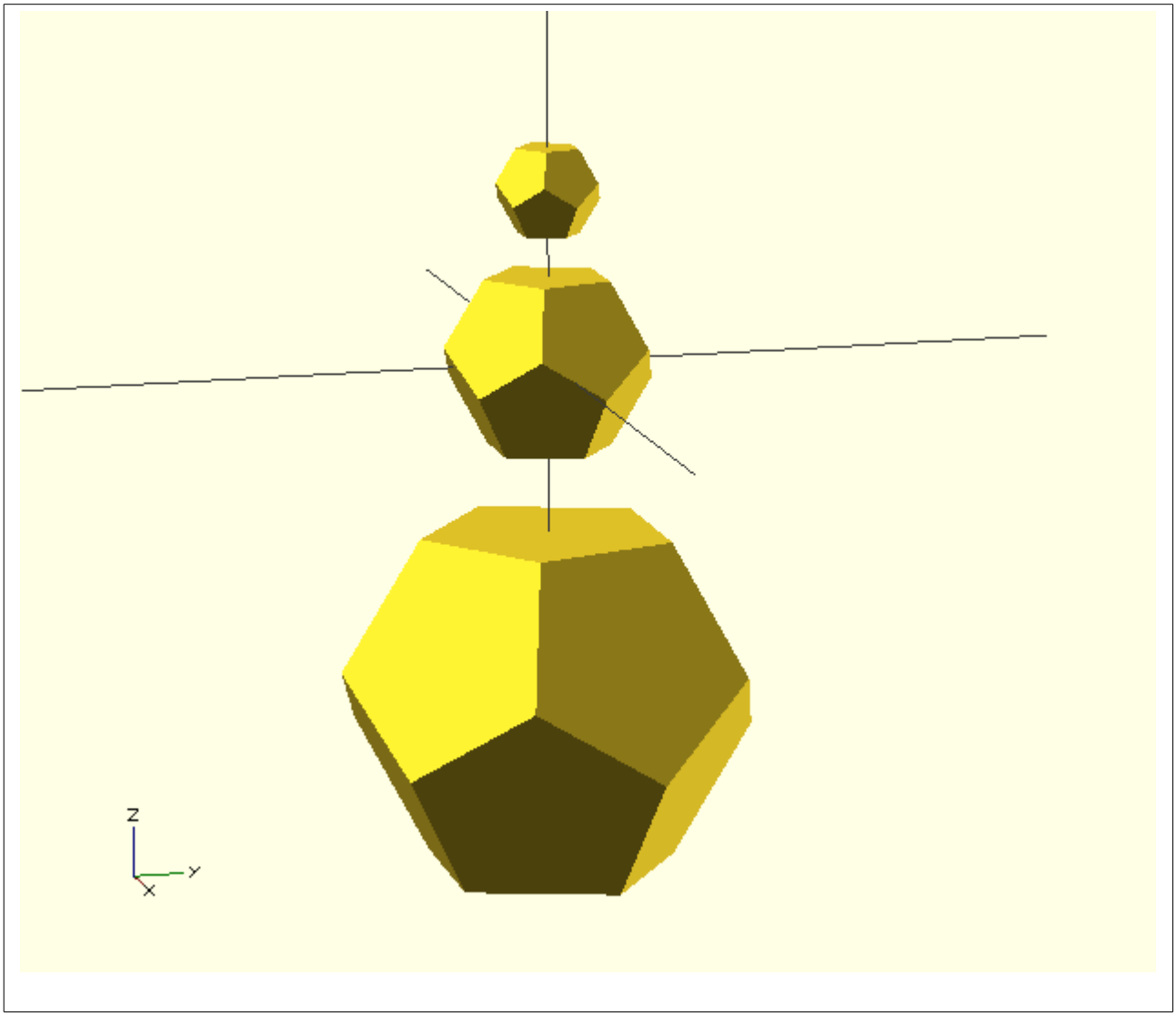
OpenSCAD tarafından tanınan renk isimleri aşağıda verilmiştir.

Purples	**Pembe**	**Mavi**	**Kırmızı**	**Yeşil**
Lavender	Pink	Aqua	IndianRed	GreenYellow
Thistle	LightPink	Cyan	LightCoral	Chartreuse
Plum	HotPink	LightCyan	Salmon	LawnGreen
Violet	DeepPink	PaleTurquoise	DarkSalmon	Lime
Orchid	MediumVioletRed	Aquamarine	LightSalmon	LimeGreen
Fuchsia	PaleVioletRed	Turquoise	Red	PaleGreen
Magenta		MediumTurquoise	Crimson	LightGreen
MediumOrchid		DarkTurquoise	FireBrick	MediumSpringGreen
MediumPurple		CadetBlue	DarkRed	SpringGreen
BlueViolet		SteelBlue		MediumSeaGreen
DarkViolet		LightSteelBlue		SeaGreen
DarkOrchid		PowderBlue		ForestGreen
DarkMagenta		LightBlue		Green
Purple		SkyBlue		DarkGreen
Indigo		LightSkyBlue		YellowGreen
DarkSlateBlue		DeepSkyBlue		OliveDrab
SlateBlue		DodgerBlue		Olive
MediumSlateBlue		CornflowerBlue		DarkOliveGreen
		RoyalBlue		MediumAquamarine
		Blue		DarkSeaGreen
		MediumBlue		LightSeaGreen
		DarkBlue		DarkCyan
		Navy		Teal
		MidnightBlue		

Turuncu	**Sarı**	**Kahverengi**	**Beyaz**	**Gri**
LightSalmon	Gold	Cornsilk	White	Gainsboro
Coral	Yellow	BlanchedAlmond	Snow	LightGrey
Tomato	LightYellow	Bisque	Honeydew	Silver
OrangeRed	LemonChiffon	NavajoWhite	MintCream	DarkGray
DarkOrange	LightGoldenrodYellow	Wheat	Azure	Gray
Orange	PapayaWhip	BurlyWood	AliceBlue	DimGray
	Moccasin	Tan	GhostWhite	LightSlateGray
	PeachPuff	RosyBrown	WhiteSmoke	SlateGray
	PaleGoldenrod	SandyBrown	Seashell	DarkSlateGray
	Khaki	Goldenrod	Beige	Black
	DarkKhaki	DarkGoldenrod	OldLace	
		Peru	FloralWhite	
		Chocolate	Ivory	
		SaddleBrown	AntiqueWhite	
		Sienna	Linen	
		Brown	LavenderBlush	
		Maroon	MistyRose	

ÖRNEKLER

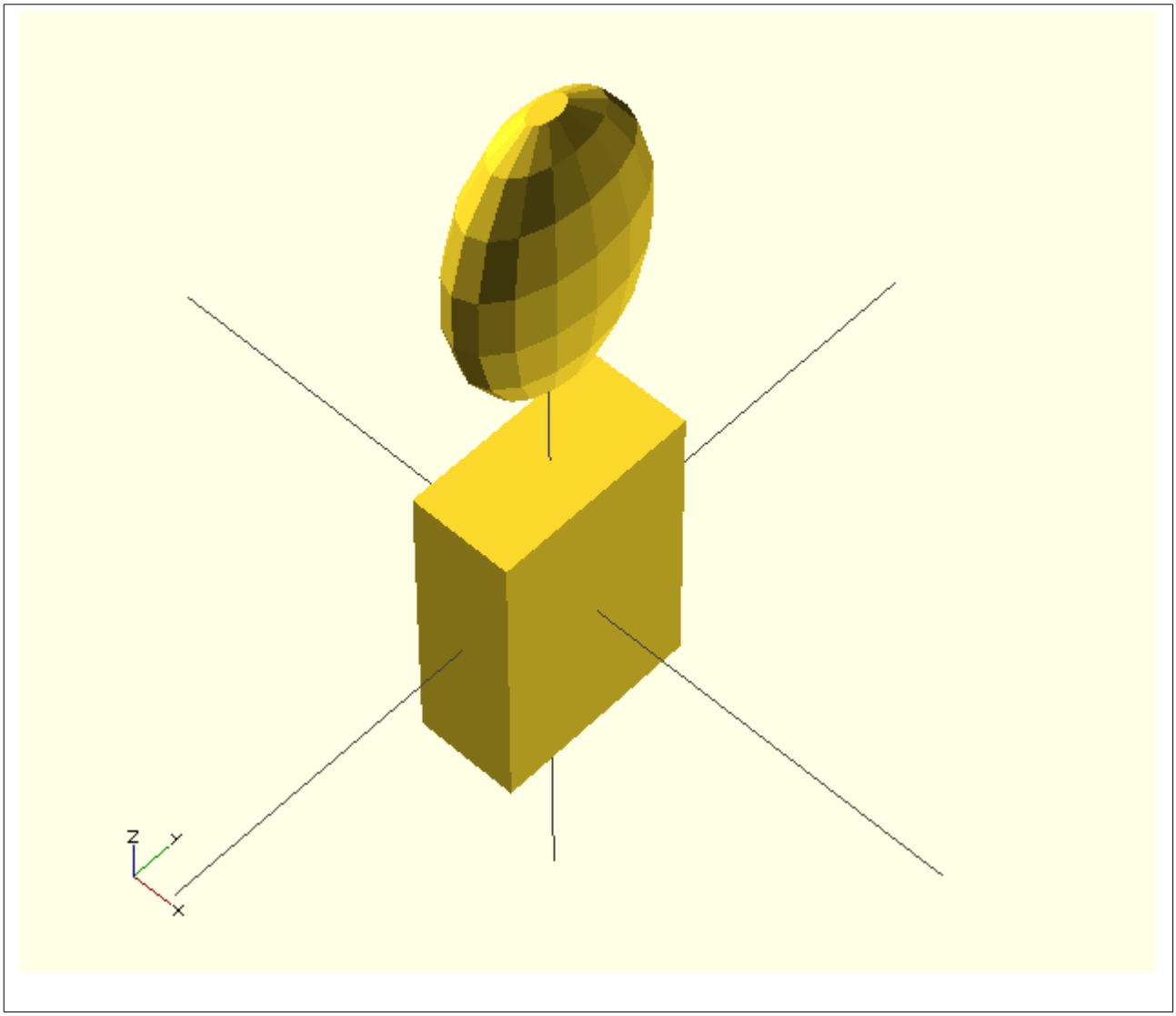
ÖRNEK 1




```
//create a dodecahedron by intersecting 6 boxes
module dodecahedron(height)
{
scale([height,height,height]) //scale by height parameter
{
intersection(){
//make a cube
cube([2,2,1], center = true);
intersection_for(i=[0:4]) //loop i from 0 to 4, and intersect results
{
//make a cube, rotate it 116.565 degrees around the X axis,
//then 72*i around the Z axis
rotate([0,0,72*i])
rotate([116.565,0,0])
cube([2,2,1], center = true);
}
```

```
}
}
}
//create 3 stacked dodecahedra
//call the module with a height of 1 and move up 2
translate([0,0,2])dodecahedron(1);
//call the module with a height of 2
dodecahedron(2);
//call the module with a height of 4 and move down 4
translate([0,0,-4])dodecahedron(4);
```

ÖRNEK 2




```
// Rather kludgy module for determining bounding box from intersecting projections
module BoundingBox()
{
intersection()
{
translate([0,0,0])
```

```
linear_extrude(height = 1000, center = true, convexity = 10, twist = 0)
projection(cut=false) intersection()
{
rotate([0,90,0])
linear_extrude(height = 1000, center = true, convexity = 10, twist = 0)
projection(cut=false)
rotate([0,-90,0])
child(0);

rotate([90,0,0])
linear_extrude(height = 1000, center = true, convexity = 10, twist = 0)
projection(cut=false)
rotate([-90,0,0])
child(0);
}
rotate([90,0,0])
linear_extrude(height = 1000, center = true, convexity = 10, twist = 0)
projection(cut=false)
rotate([-90,0,0])
intersection()
{
rotate([0,90,0])
linear_extrude(height = 1000, center = true, convexity = 10, twist = 0)
projection(cut=false)
rotate([0,-90,0])
child(0);

rotate([0,0,0])
linear_extrude(height = 1000, center = true, convexity = 10, twist = 0)
projection(cut=false)
rotate([0,0,0])
child(0);
}
}
}

// Test module on ellipsoid
translate([0,0,40]) scale([1,2,3]) sphere(r=5);
BoundingBox() scale([1,2,3]) sphere(r=5);
```