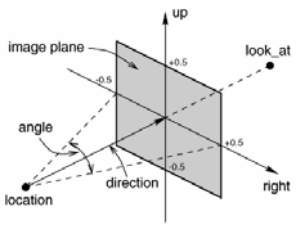
Camera

```
camera {
   sky <0,1,0>
   location  <2,  2.2, -3>
   direction <0,  0,  1>
   look_at   <0.7,  1.2,   0>
   up        <0,  3,   0>
   right   <4,  0,   0>
}
```



Simple Geometric Objects

- Sphere
- Box
- Cylinder
- Cone
- Plane
- Torus

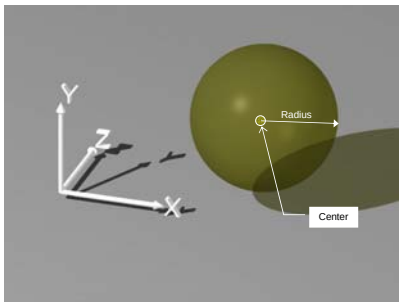
Sphere

```
sphere {
      <Center>, Radius
      //Surface properties …
   }
```

```
sphere { <4, 2, 1>, 1.5
         pigment {
                 red 0.9
                 green 0.9
                 blue 0.5
                 filter 0.7}
         finish {
                  phong 0.2
   }
}
```

Center

Radius



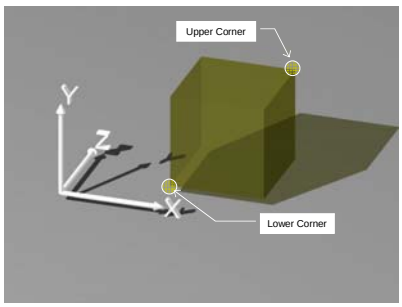
Box

```
box{
   <lower corner>, <upper corner>
   // surface properties …
   }
```

Upper corner

Lower corner

```
box { <2,0.2,1>, <4,2.2,3>
        pigment {
                red 0.9
                green 0.9
                blue 0.5
                filter 0.7}
        finish {
                 phong 0.2
    }
}
```

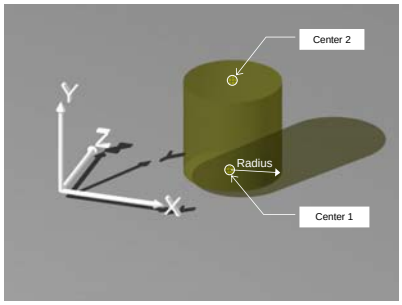


Cylinder

```
cylinder {
   <center1>,<center2>,<radius>
         //surface properties …
   }
```

```
cylinder {
        <3,0.2,2>,
        <3,2.2,2>,
        1
        pigment {
                red 0.9
                green 0.9
                blue 0.5
                filter 0.7}
        finish {
                phong 0.2
   }
}
```

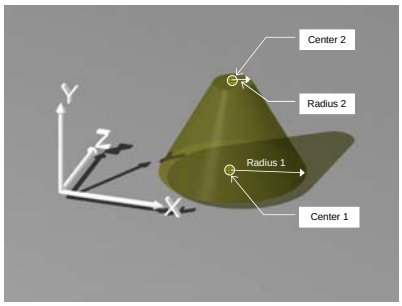
Center 1
Center 2
Radius



Cone

```
cone{ <center1>, <radius1>,
      <center2>, <radius2>
       // surface properties …
   }
```

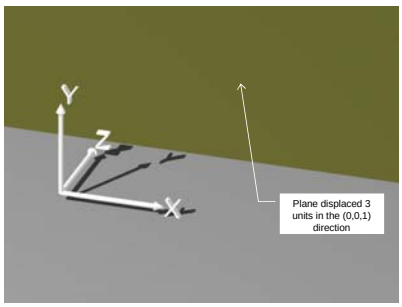
```
cone {
        <3,0.2,2>, 1.5,
        <3,2.2,2>, 0.4
        pigment {
                red 0.9
                green 0.9
                blue 0.5
                filter 0.7
        }
        finish {
                phong 0.2
        }
}
```



Plane

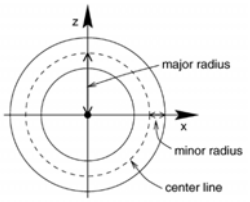
```
plane { <normal>, <distance>
        //surface properties …
   }
```

```
plane {
        <0,0,1>, 3
        pigment {
                red 0.9
                green 0.9
                blue 0.5
                filter 0.7
        }
        finish {
                phong 0.2
        }
}
```



Torus

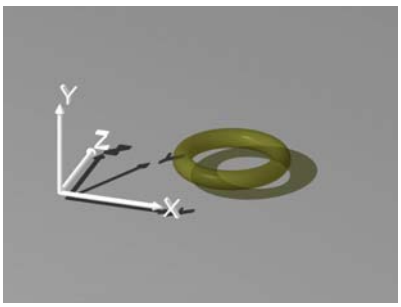
```
torus{
        Major radius,
        Minor radius,
   // surface properties …
   }
```



Torus

```
torus{
        Major radius,
        Minor radius,
   // surface properties …
   }
```

```
torus { 1, 0.2
        pigment {
                red 0.9
                green 0.9
                blue 0.5
                filter 0.7}
        finish {
                 phong 0.2
        }
        translate <3, 0.5, 2>
}
```



An example

```
#include "colors.inc"

camera{
    location <0, 5, -7>
    look_at   <0, 2, 0>
}

light_source{<-4, 10, -2.5>
      color red 1 green 1 blue 1
}

cone { <0,0,1>, 2.5
       <0,3,1>, 0.5
        pigment { color Blue }
        finish {diffuse 1 ambient .4}
}

sphere { <0,4,1>, 1.1
        pigment { color Blue }
        finish {diffuse 1 ambient .5}
}
```

Moving, Sizing, Orientation

- Translate
- Scale
- Rotation

Translate

- Allow objects to be moved
- Translations are always relative to the object location before the move

```
Translate <a,b,c>
  Translates the object a units in x, b units in y,
  and c units in z
Translate 3*x
  Translates the object 3 units along the x axis
```

Scale

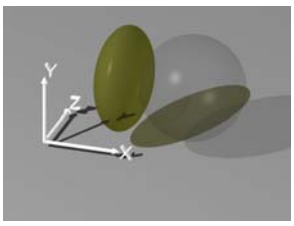
- Allow size of objects to be changed

```
Scale <x,y,z>
```

- Three terms of the vector specify the amount of scaling in each of the x, y and z directions
- Scaling occurs relative to the World's origin

Scale

```
sphere { <4, 2, 1>, 1.5
        pigment {
               red 0.9
               green 0.9
               blue 0.5
               filter 0.7}
        finish {
                phong 0.2
        }
        scale <0.5,1,1>
}
```



Scale

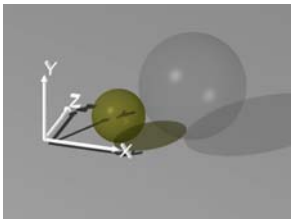
`Scale` n

- Evaluates as <n,n,n> so uniformly scale by n in every direction

```
sphere { <4, 2, 1>, 1.5
         pigment {
               red 0.9
               green 0.9
               blue 0.5
               filter 0.7}
         finish {
                  phong 0.2
        }
         scale 0.5
}
```

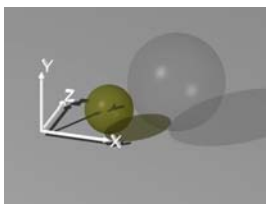
Scale 0.5

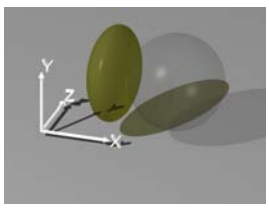
```
sphere { <4, 2, 1>, 1.5
         pigment {
               red 0.9
               green 0.9
               blue 0.5
               filter 0.7}
         finish {
                  phong 0.2
        }
         scale 0.5
}
```



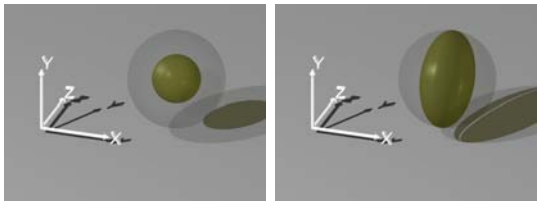
Problem

Sphere has changed location after being scaled!!!!



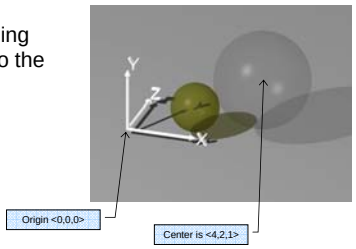


We want this



Analysis

- The sphere is defined at <4,2,1>
- Remember Scaling occurs relative to the origin

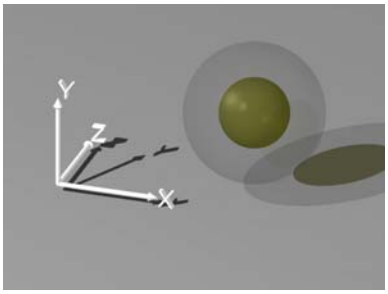


Fix

- Create the sphere at the origin
- Scale the sphere first and then move it to <4,2,1>

```
sphere { <0, 0, 0>, 1.5
         pigment {
                 red 0.9
                 green 0.9
                 blue 0.5
                 filter 0.7}
         finish {
                  phong 0.2
         }
         scale 0.5
         translate <4,2,1>
}
```

Voila!!



Rotate

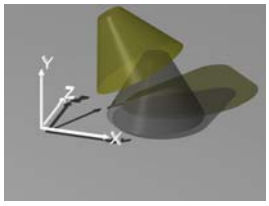
- Change the orientation of an object

`Rotate <x,y,z>`

- The three terms of the vector specify the number of degrees to rotate about each of the x-, y- and z-axes
- Also occurs relative to the World's origin

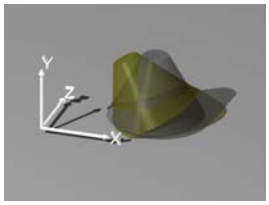
Rotate

```
cone{
        <3,0.2,2>, 1.5,
        <3,2.2,2>, 0.4
        pigment {
                red 0.9
                green 0.9
                blue 0.5
                filter 0.7}
        finish {
                 phong 0.2
        }
        rotate <0,0,30>
}
```



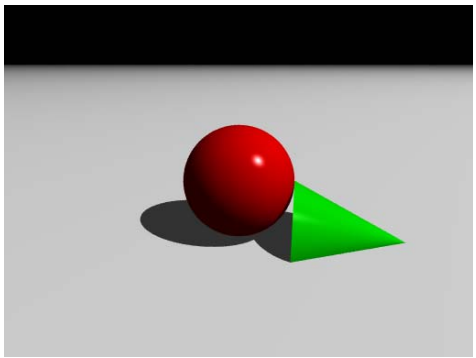
Rotate

```
cone{
        <0,0,0>, 1.5,
        <0,2,0>, 0.4
         pigment {
                red 0.9
                green 0.9
                blue 0.5
                filter 0.7}
         finish {
                 phong 0.2
        }
         rotate <0,0,30>
        translate <3, 0.2, 2>
}
```



An example

```
cone { <0,0,0>, 1, <0,2,0>, .001
       rotate <0, 0, -90>
       translate <1, 0, 0>
       pigment {color Green}
}
```



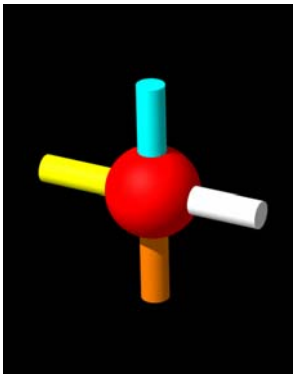
What happens if

- Rotate, then translate
- Translate, then rotate

In general

- Create object at the origin
- Scale
- Rotate
- Translate

An exercise



Text

- A text object creates 3-D text as an extruded block letter.
- Currently only TrueType fonts (ttf) and TrueType Collections (ttc) are supported in POV-Ray

Text

```
text { ttf "fontname.ttf/ttc" "String_of_Text“
       Thickness, <Offset>
       //Surface properties…
}
```

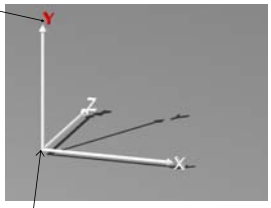
Text to be displayed

Spacing between letters

How thick (in depth) the text is

Text

```
text {
    ttf "arial.ttf" "Y" 0.3, 0
    pigment { Red }
    finish {ambient 0.7}
    scale 0.5
    translate 4.2*y
 }
```



Origin <0,0,0>

CSG

Constructive Solid Geometry

CSG

- What is it?
- Syntax in POV-Ray
- union
- intersection
- difference
- merge

CSG

- Allows you to combine multiple simple shapes into more complex ones
- CSG objects can be composed of primitives or other CSG objects to create more, and more complex shapes

Creating a shape using CSG

- Use the #declare statement to define the shape
- Use the object statement to display an instance of that shape

Example - Union

```
#declare pawnU = union {
       sphere{<.5, 1.7, .5>, .5}
       cone { <.5,.5, .5>.5, <.5,1.5,.5> 0.25  }
       box { <0,0,0>,<1,.5,1> }
}

object { pawnU
        pigment {
              red 0.9
              green 0.9
              blue 0.5
              filter 0.7}
        finish {
                phong 0.2   }
 }
```

"pawnU" is the name given to the union of the sphere, cone, and box

An instance of "pawnU" has been created with the object statement

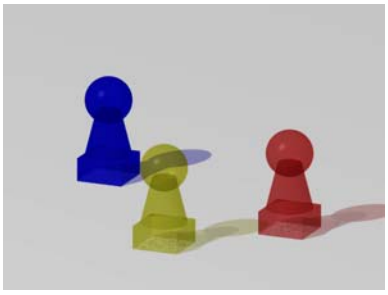
Union!

Multiple instances

```
object { pawnU
        pigment {red 0.9 green 0.9 blue 0.5 filter 0.9}
        finish { phong 0.2   }
        scale 0.5
}

object { pawnU
        pigment {red 0.9 green 0.5 blue 0.5 filter 0.9}
        finish { phong 0.2   }
        scale 0.5
        translate <1,0,1>
}

object { pawnU
        pigment {red 0.1 green 0.1 blue 0.9 filter 0.9}
        finish { phong 0.2   }
        scale 0.5
        translate <-1.5,0,2>
}
```

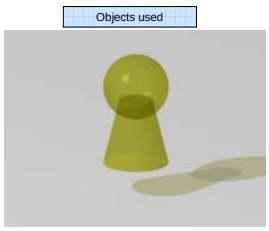
Multiple instances of "pawnU"

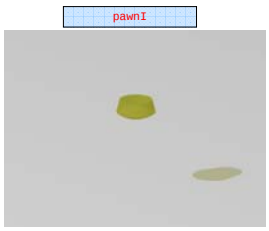
Multiple instances



Example - Intersection

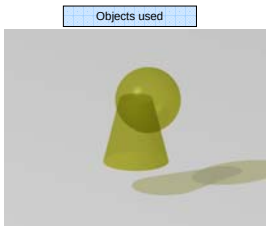
```
#declare pawnI = intersection {
        sphere{<.5, 1.7, .5>, .5}
        cone { <.5,.5, .5>.5, <.5,1.5,.5> 0.25  }
}

object { pawnI
         pigment {
               red 0.9
               green 0.9
               blue 0.5
               filter 0.7}
         finish {
                 phong 0.2   }
 }
```

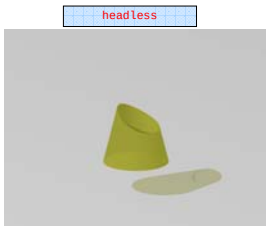




Example - difference

```
#declare headless = difference {
        cone { <.5,.5, .5>.5, <.5,1.5,.5> 0.25  }
       sphere{<.7, 1.5, .5>, .5}
}

object { headless
         pigment {
              red 0.9
              green 0.9
              blue 0.5
              filter 0.7}
         finish {
                 phong 0.2   }
 }
```

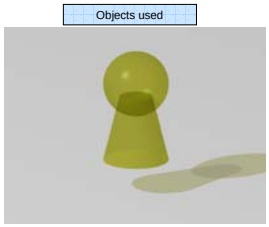




Example - merge

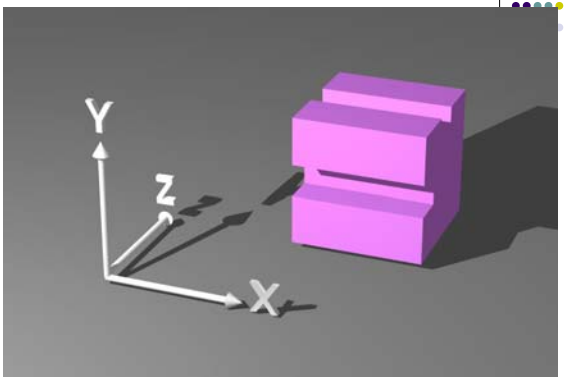
```
#declare mergedPawn = merge {
        cone { <.5,.5, .5>.5, <.5,1.5,.5> 0.25  }
       sphere{<.5, 1.7, .5>, .5}
}

object { mergedPawn
         pigment {
              red 0.9
              green 0.9
              blue 0.5
              filter 0.7}
         finish {
                 phong 0.2   }
 }
```

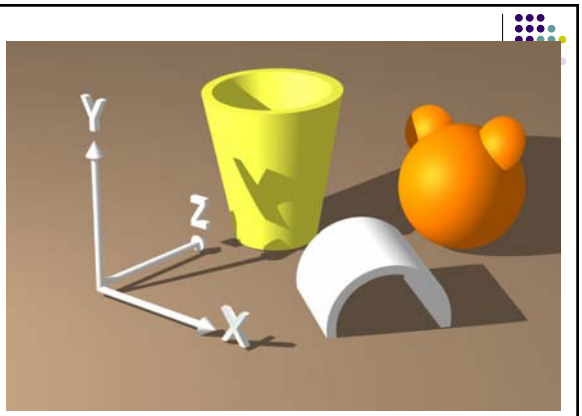
Objects used



mergedPawn

CSG-Other examples


Y
Z
X


Y
Z
X