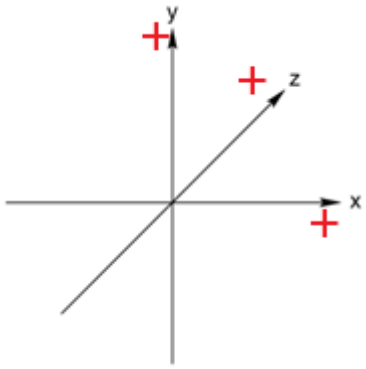
Coordinate system



Pov-Ray follows a left-handed coordinate system. If one stretches the thumb, index finger, and middle finger of left hand, each towards the x,y, and z directions as the picture 1 (left), the tips of each finger point to the positive direction of the axes in a 3D coordinate system of Pov-Ray image.

Background

A variety of choices for colors can be found at the top menu of the POV-Ray software under the 'Insert / Colors' tab. Prior to the color selection, POV-Ray version must be specified and "color.inc" must be included. The Adelson checkerboard uses 'Gray50' as the image below:

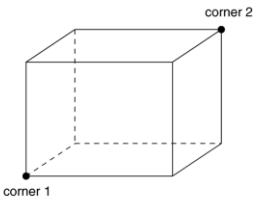
```
#version 2.5
#include "colors.inc"

Background { color CHOICE}
```

Basic objects

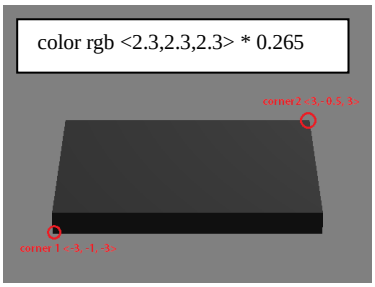
1. Board

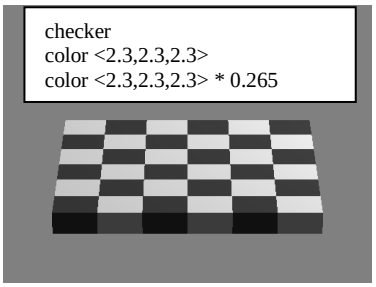
```
box {
        <Corner_1>, <Corner_2>
        [OBJECT_MODIFIERS...]
    }
```



1.1 Color

The 'pigment { }' argument inside the "OBJECT_MODIFIERS" slot is used to fill an object with one or more specified choice of color or pattern. For example:



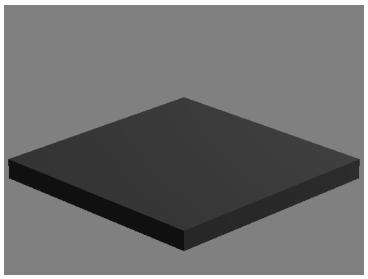


1.2 Rotation

An object can be rotated by placing the argument, ‘rotate AXIS * Θ ’, right after the argument specifying it. The Adelson checkerboard is rotated -45 degrees around the z-axis. For example:

Box { … } rotate x * -45

Box { … } rotate y * -45



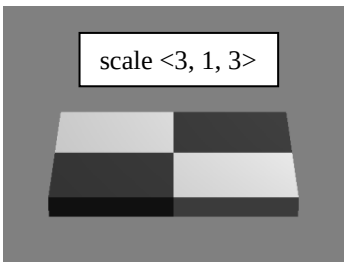
Box { … } rotate z * -45

1.3 Number of checks

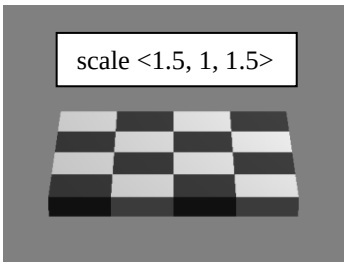
The number of checks on a checkerboard can be specified with the ‘scale { }’ argument inside the ‘pigment { }’ argument of the box object argument.

```
box{
        <-3, -0.6, -3>
        < 3, -0.3, 3>
        pigment {
                    checker
                    color <2.3, 2.3, 2.3>
                    color <2.3, 2.3, 2.3> * 0.265
                    scale <x,y,z>
                  }
    }
```

scale <3, 1, 3>

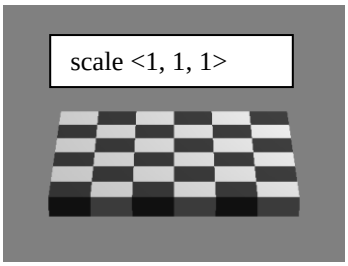


scale <1.5, 1, 1.5>

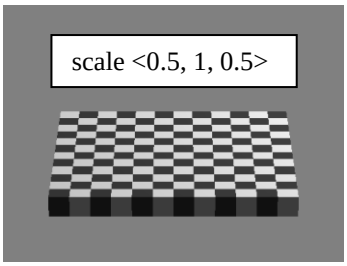


The number of checks on a board can be manipulated by varying the scales of x-axis and z-axis with a same value. More precisely, the scale vector determines the size of a single check.

scale <1, 1, 1>

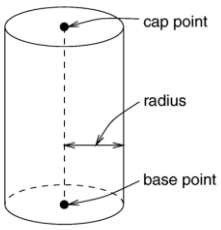


scale <0.5, 1, 0.5>



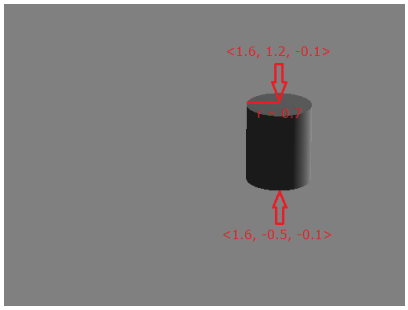
For example, the example boards on the left are constructed with a different number of checks with scalar multiple of 3, since they all have lengths of 6 in both x and z directions.

2. Cylinder




```
Cylinder {
        <Base_Point>, <Cap_Point>, Radius
        [ open ] [OBJECT MODIFIERS]
       }
```

A cylinder object needs three main specifications: base and cap points that determine the location on an image and the height, and a radius that determines the width. The cylinder in the original Adelson checkerboard script is shown below:

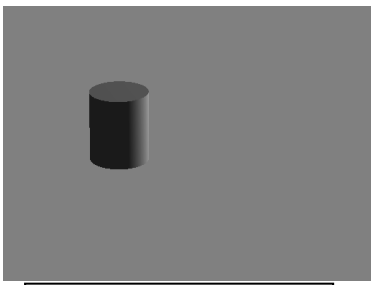


e.g.

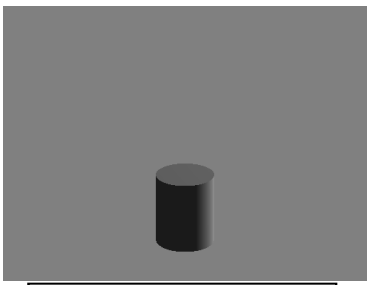
```
cylinder{
        <2.8, -0.5, 0.3>
        <2.8, 1.2, 0.3>
        .7
        pigment{    rgb <1.0, 1.0, 1.0> }
        translate <-1.2,0,-.4>
       }
```

An additional parameter, 'translate <x,y,z>', is used to move about the cylinder in accordance with the Pov-Ray coordinate system. The above script can be re-written as below:

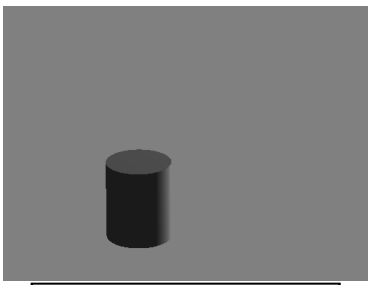
```
Cylinder {
        <0, 0, 0>
        <0, 1.7, 0>
        .7
        pigment{    rgb <1.0, 1.0, 1.0> }
        translate <1.6,-0.5,0.3>
        }
```



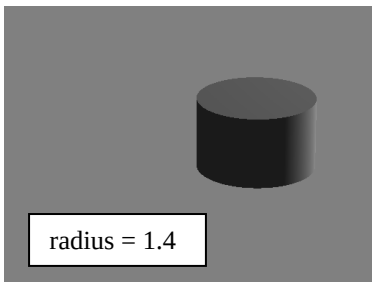
translate <-1.6,-0.5,0.3>



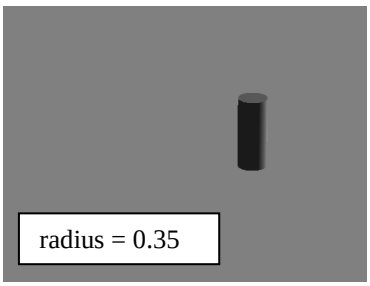
translate <0,-2.5,0>



translate <-1.0,0,-5.3>



radius = 1.4

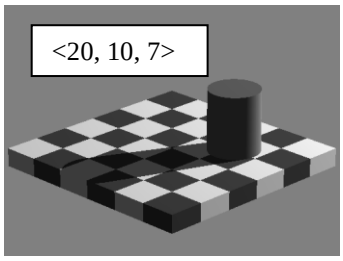


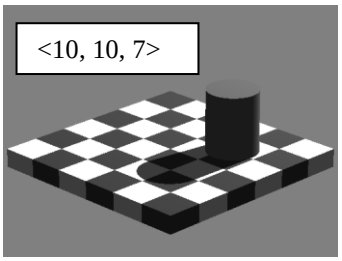
radius = 0.35

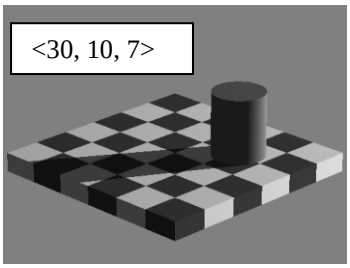
Light source

```
light_source {
                <Location>, COLOR
                [LIGHT_MODIFIERS...]
              }
```

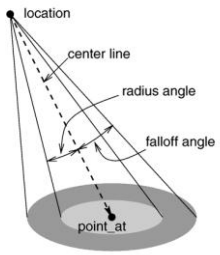
1) Point light
   The point light takes only two parameters: location and color. However, it only varies the intensity othe light, because the point light source illuminates every object on an image equally.





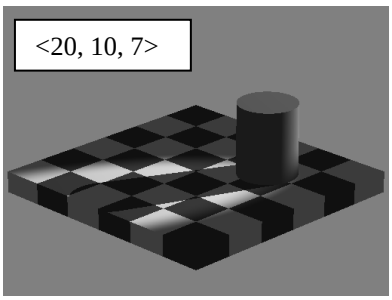


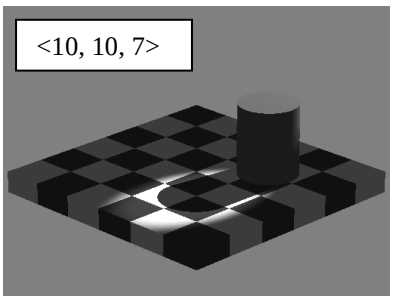
2) Spotlight
   The spotlight source radiates light from the source location (‘location’) in a cone-shaped fashion. There are four important parameters: point_at, radius, and falloff. When the spotlight hits an object, it produces a bright circle of light with specified center coordinate (‘point_at’) and radius angle (‘radius’) and a ring of dimmer light that surrounds it. The edges of the outer ring are defined by a falloff angle (‘falloff’).

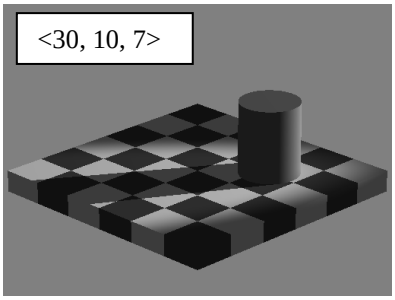



```
light_source {
                <Location>, COLOR
                spotlight
                point_at <x,y,z>
                radius X
                falloff X
              }
```

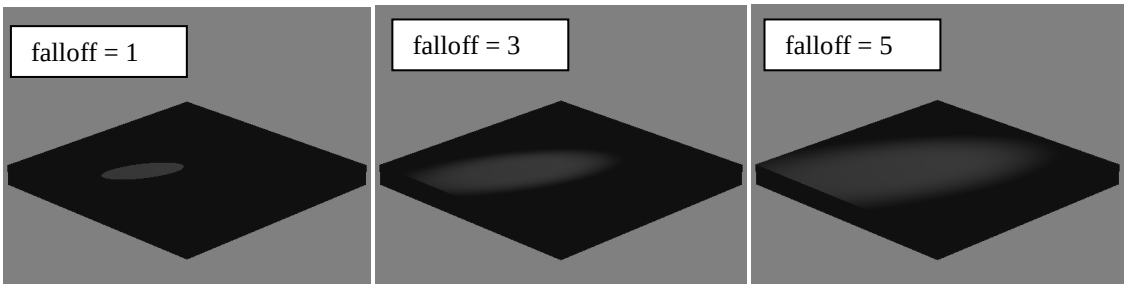
The three scenes below are generated with spotlights of radius = 3, falloff = 5, and tightness = 0, pointed at the center of the cylinder’s cap point, <1.6, 1.2, -0.1>. The coordinates on the top left of each image indicates the location of a light source.



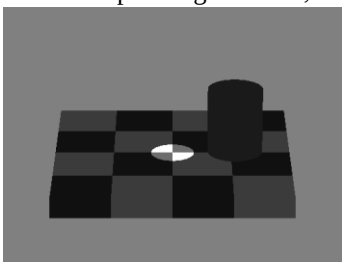




The three images below are generated with spotlights of radius =1 and varying falloffs. The light source is located at <20, 10, 7> and is pointed at <0, 0, 0>.

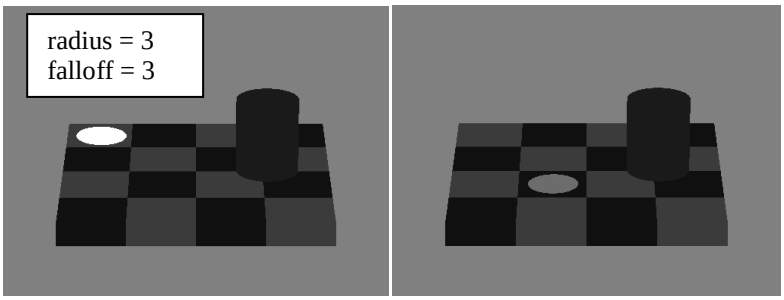


Unlike the point light source, the spotlight can be used to illuminate a specific region.



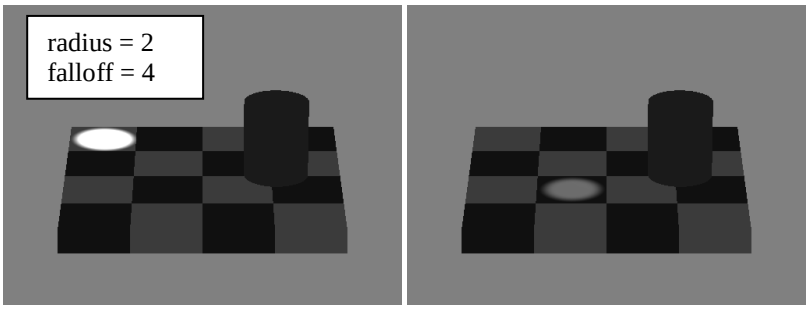
```
light_source {
                <0,10,0>, color White
                spotlight
                point_at <0,0,0>
                radius 3
                falloff 3
               }
```

The spotlight can be used to cast a light on a specified location (i.e. a single check) as shown in the two images below:



Note that light sources are located at <20,10,7> in both images above.

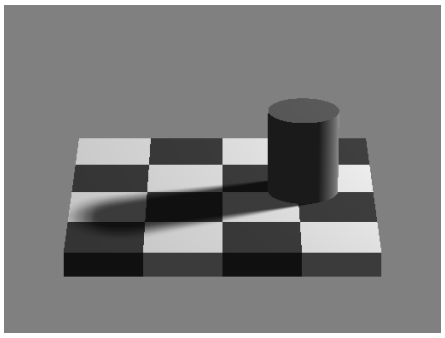
The 'falloff' command can be used to produce an effect of blurring edge as the image below:



3) Area light

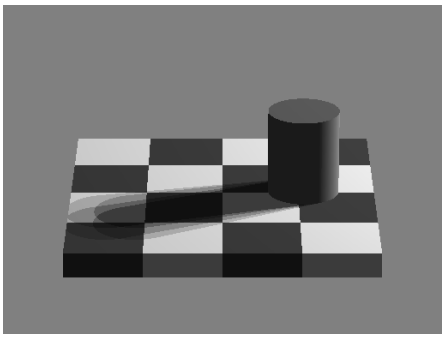
```
light_source {
                LOCATION_VECTOR, COLOR
                area_light
                AXIS_1_VECTOR, AXIS_2_VECTOR, Size_1, Size_2
                [ [LIGHT_MODIFIERS...]
               }
```

The area light is constructed by an array of multiple point light sources. It radiates from a rectangular-shaped grid source with a center (‘location_vector’) and lengths and directions of the edges (‘axis_1’ and ‘axis_2’ only in x and z axis). The number of point light sources on a grid is specified by the product of two integers (‘size_1’ and ‘size_2’). Since there are multiple light sources, the resulting illumination may create multiple shadows for an object.



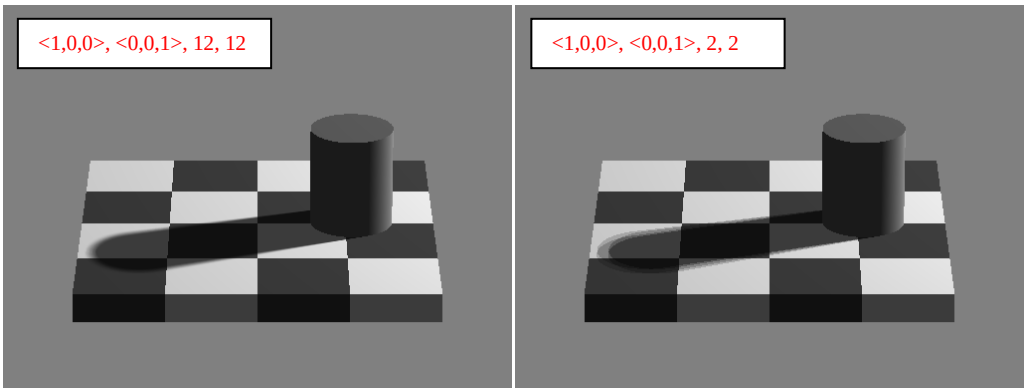
```
light_source {
        <20,10,7>, color White
        area_light
        <3,0,0>, <0,0,3>, 12, 12
      }
```

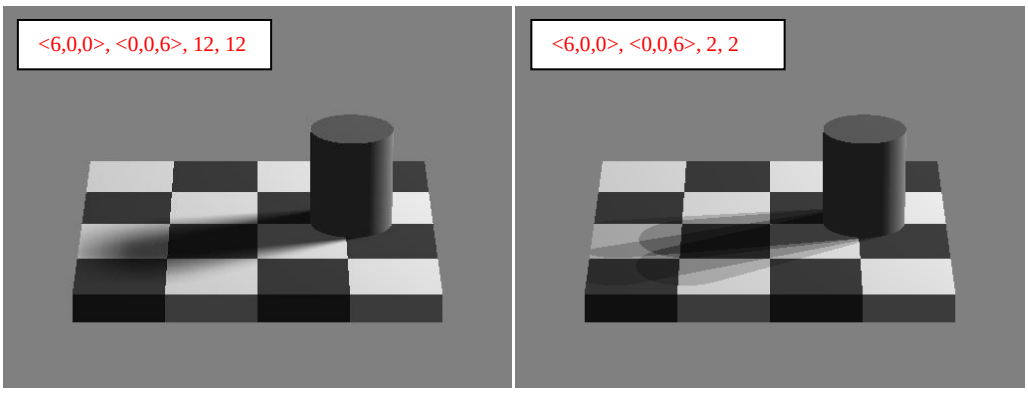
The area light is constructed with a total of 144 (12x12) point sources on a grid. The resulting shadow of the cylinder looks smooth, since there were a lot of shadows that overlay each other.



```
light_source {
        <20,10,7>, color White
        area_light
        <3,0,0>, <0,0,3>, 2, 2
      }
```

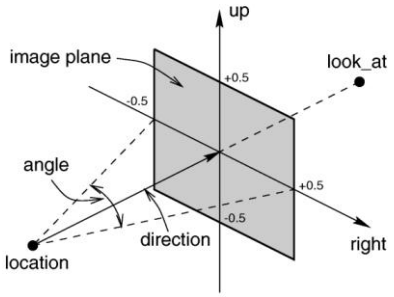
The area light is constructed with a total of 4 (2x2) point sources on a grid. All the four shadows of the cylinder are visible.





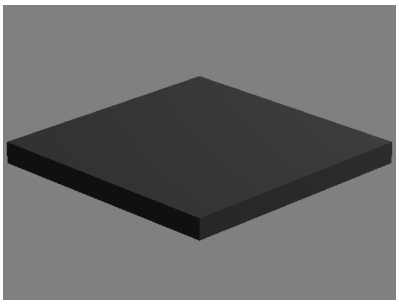
Camera

```
Camera {
        Location <x,y,z>
        Up <x,y,z>, Right <x,y,z>    (optional)
        Look_at <x,y,z>
        Angle Θ
      }
```

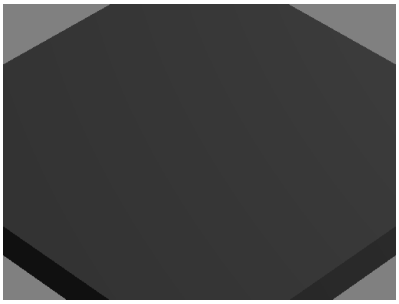


An image can be viewed at different angle by placing a POV-ray camera to a desired location. As shown above, only the image that falls inside a specified image plane is projected onto a screen. An image plane can be specified with the following four parameters: location vector that specifies the viewing position, up and right vectors that specify the size of an image plane, look_at vector that specifies the focus point on an image, and angle vector that specifies the viewing angle. However, a simple 3-dimensional checkerboard can be constructed with only three parameters: location, look_at, and angle, which can be treated as a zooming in/out function.

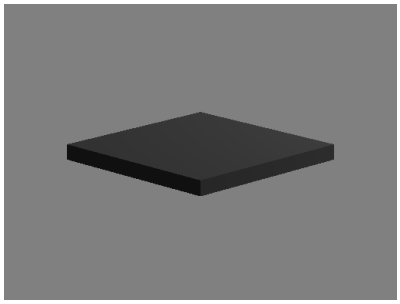
1. Varying location



```
Camera {
        location <0, 20, -50>
        look_at < 0, 0, 0>
        angle 9.2
      }
```

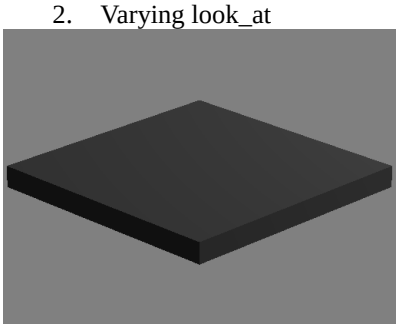


location <0, 20, -25>
look_at < 0, 0, 0>
angle 9.2

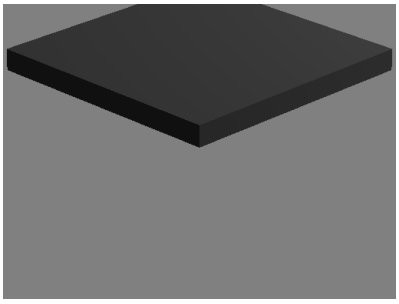


location <0, 20, -75>
look_at < 0, 0, 0>
angle 9.2

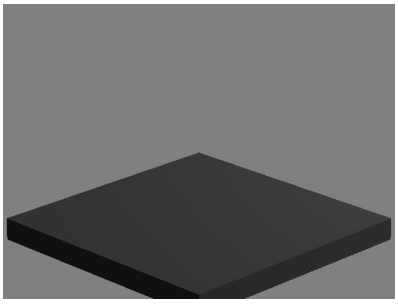
2. Varying look_at



```
Camera {
        location <0, 20, -50>
        look_at < 0, 0, 0>
        angle 9.2
      }
```

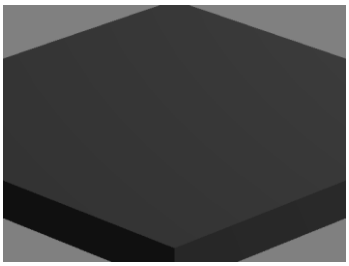


location <0, 20, -50>
look_at < 0, 0, -5>
angle 9.2

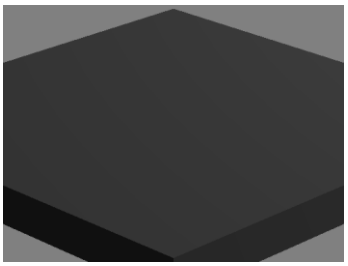


location <0, 20, -50>
look_at < 0, 0, 5>
angle 9.2

3. varying angle

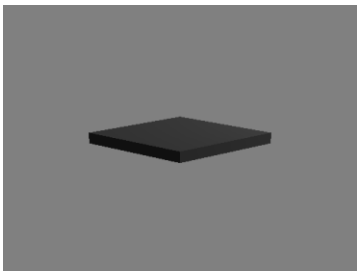


location <0, 20, -50>
look_at < 0, 0, 0>
angle 4.6

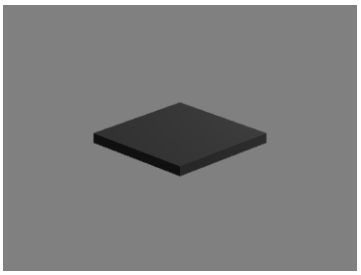


location <0, 10, -25>
look_at < 0, 0, 0>
angle 9.2

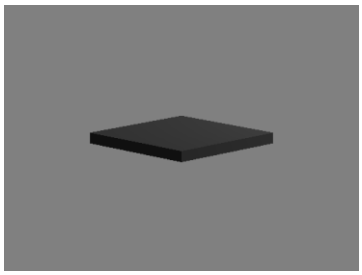
location <0, 20, -25>
look_at < 0, 0, 0>
angle 9.2



location <0, 20, -50>
look_at < 0, 0, 0>
angle 18.4



location <0, 40, -100>
look_at < 0, 0, 0>
angle 9.2



location <0, 20, -100>
look_at < 0, 0, 0>
angle 9.2