# More on POV-Ray

## Surface Shading

- Once an object is defined, we need to deal with other issues such as
- Surface color
  - pigment
- Surface properties
  - finish

## Surface color

- Pigment statement
  - The color or pattern of colors for an object

## Surface properties

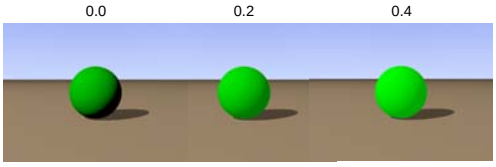
- How does light reflect?
  - Matte - Mirror
- What happens in shadows?
- What kind of highlights are visible?
- Transparency?

## Surface properties

- Finish statement

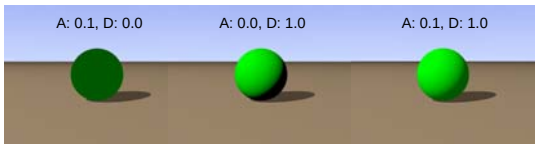
## ambient

- controls the amount of ambient light
- Default is 0.1 (if no ambient is specified)



## diffuse

- control how much of the light coming directly from any light sources is reflected via diffuse reflection
- Default is 0.6




## phong

- controls the amount of Phong highlighting on the object. It causes bright shiny spots on the object that are the color of the light source being reflected

## specular

- produces a highlight which is very similar to Phong highlighting but it uses slightly different model

## reflection

- When light does not diffuse and it *does* reflect at the same angle as it hits an object, it is called *specular reflection*

## roughness

- The size of the spot is defined by the value given the roughness keyword. Typical values range from 1.0 (very rough - large highlight) to 0.0005 (very smooth - small highlight). The default value, if roughness is not specified, is 0.05 (plastic).

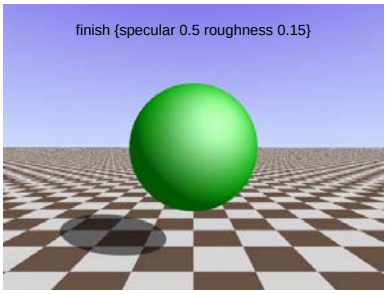
## Some typical surfaces

- Dull surface
  - Creates a large, soft highlight on the object's surface

```
finish {specular 0.5 roughness 0.15}
```

- Don't forget that ambient and diffuse default values are being used here too (0.1 and 0.6)

## Dull

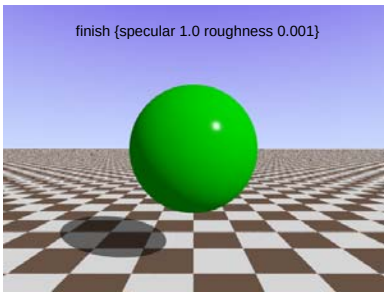



## Some typical surfaces

- Shiny surface
  - Shiny surface: creates a small, tight highlight on the object's surface

```
finish {specular 1 roughness 0.001}
```

## Shiny

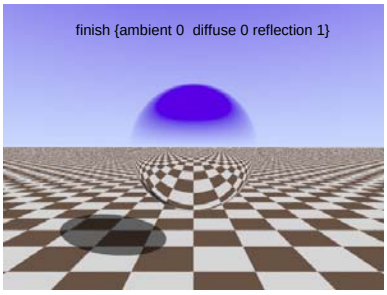


## Some typical surfaces

- Mirror surface
  - a perfectly mirrored finish with no highlights

```
finish {ambient 0 diffuse 0 reflection 1}
```

## Mirror

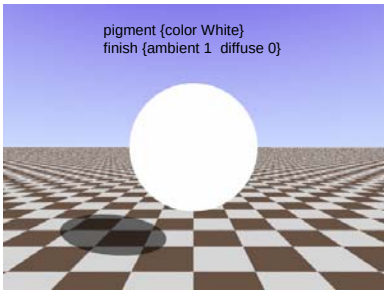



## Some typical surfaces

- Luminous surface
  - Luminous for shadowless skies and light_sources

```
finish {ambient 1 diffuse 0}
```

## Luminous

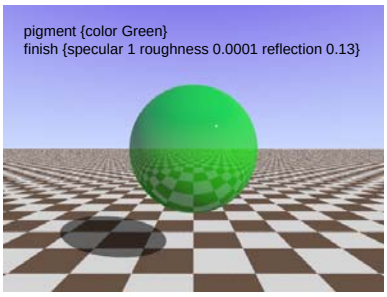



## Some typical surfaces

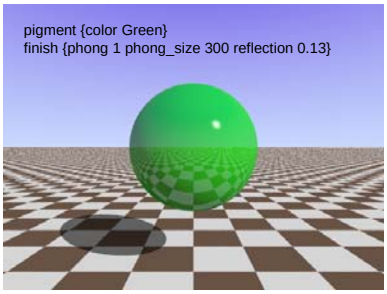
- Glossy surface
  - Very shiny with very tight highlights and a fair amount of reflection

```
finish {   specular 1
           roughness 0.0001
           reflection 0.13
}
```

## Glossy



## Phong Glossy




## Other examples

- Phong highlights
  - less "realistic" than specular, but useful for different effects
  - Worth to try:

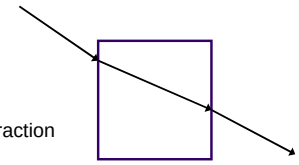
**Phong_Dull**

```
finish {phong 0.5  phong_size 1}
```

**Phong_Shiny**

```
finish {phong 1  phong_size 200}
```

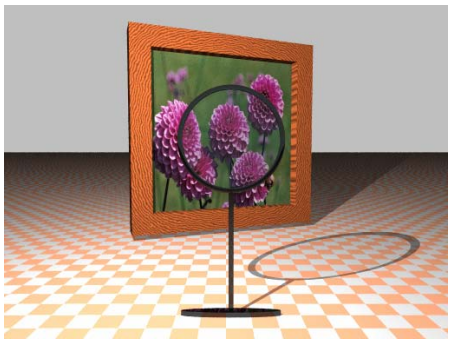
## Transparency



Index of refraction

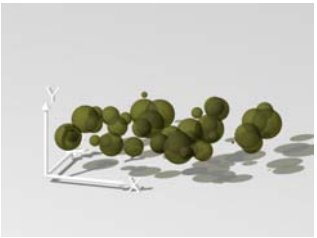
POV-Ray: interior {ior 1.3}
default (air): 1.0

## Experiment



## More about POV-Ray

- Generate 50 spheres, with different radii randomly located at a height of 1



## Conditional Directives

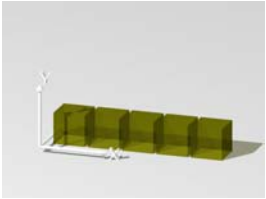
- #if...#else...#end

## Loops

```
#while(condition)
...
#end
```

## Loops

- Create five boxes along the x axis



## Version 1

```
#declare Count=0;
#while (Count < 5)
    box { <0.1,0,1>, <1.1,1,2>
          pigment { color rgb<0.9,0.9,0.5>
                    filter 0.5}
          translate x*1.1*Count
    }
    #declare Count=Count+1;
#end
```

Create an instance of object "b" translate it along x

Increment counter

## Version 2

```
#declare b = box { <0.1,0,1>, <1.1,1,2>
                   pigment {
                      color rgb<0.9,0.9,0.5>
                      filter 0.5}
                 }
#declare Count=0;
#while (Count < 5)
    object {b translate x*1.1*Count }
    #declare Count=Count+1;
#end
```

Declare a box

Create an instance of object "b" translate it along x

Increment counter

## Loops

- Create five boxes with changing color and transparency

## Loops

```
#declare b = box { <0.1,0,1>, <1.1,1,2> }
#declare f = 0.1;
#declare Count=0;
#while (Count < 5)
    object {b pigment {color rgb<f, 0.1, 0.5>
                       filter f}
            finish {ambient 0.6}
            translate x*1.2*Count }
 #declare Count=Count+1;
 #declare f = f + 0.15;
#end
```

The red component of the color changes

Transparency also changes

## Arrays

- Definition

```
#declare Array1D = array[10]
#declare Array2D = array[10][10]
```

- Index from 0 - 9

## Arrays

- Initializers

```
#include "colors.inc"
#declare arr = array[3] {Red,White,Blue}
#declare Digits = array[4][10] {
  {7,6,7,0,2,1,6,5,5,0},
  {1,2,3,4,5,6,7,8,9,0},
  {0,9,8,7,6,5,4,3,2,1},
  {1,1,2,2,3,3,4,4,5,5} }
```

## Arrays

- Include file
  - arrays.inc
  - Functions for handling arrays

## Randon numbers

- rand.inc
  - Functions for handling random numbers

  Seed

  Rand
  - What is it good for?

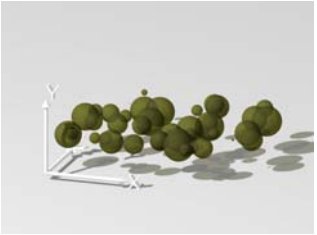
## Seed and Rand

- Before you can use the randomizer, you need to set a seed

```
#declare a = seed(1);
#declare px = rand(a);
```

- Px will be a random number between 0 and 1

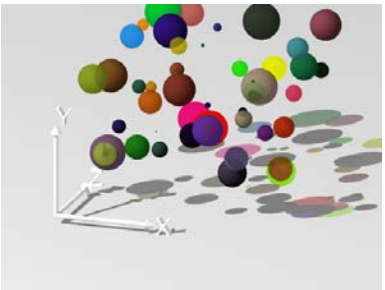
- Generate 50 spheres, with different radii randomly located at a height of 1



```
#declare Count=0;
#declare myColor = rgb<0.9,0.9,0.5>;
#declare rad = seed(0.2);
#declare a = seed(1);
#while (Count < 50)
    #declare r = rand(rad)*0.4;
    #declare px = rand(a);
    #declare pz = rand(a);
    sphere{ <0,0,0>, r
          pigment { color myColor filter 0.5}
          translate <px*5,1,pz*5>
    }
    #declare Count=Count+1;
#end
```

Spheres will have a radius between 0 and 0.4

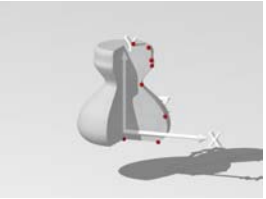
## Exercise

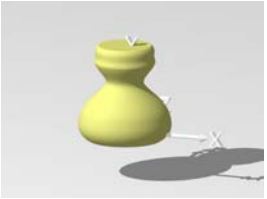



## sor

- Surface of revolution
- Generated by rotating the graph of a function about the y-axis.

## sor





## sor

```
#declare vase = sor{ 8,
   < 0.00, 0.00>,
   < 0.60, 0.00>,
   < 0.72, 0.44>,
   < 0.31, 0.93>,
   < 0.49, 1.26>,
   < 0.48, 1.35>,
   < 0.43, 1.56>,
   < 0.16, 1.60>
}

object { vase  pigment { color myColor }
               finish { ambient 0.5 diffuse 0.85 }
               scale 1.5
       }
```

# of points in the sor

Define the profile of the sor

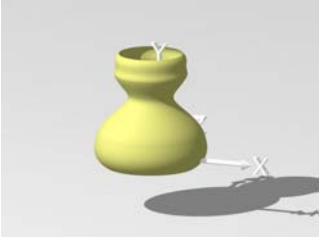
Create an instance of the sor

## Sor - open

```
#declare vase = sor{ 8,
   < 0.00, 0.00>,
   < 0.60, 0.00>,
   < 0.72, 0.44>,
   < 0.31, 0.93>,
   < 0.49, 1.26>,
   < 0.48, 1.35>,
   < 0.43, 1.56>,
   < 0.16, 1.60>
  open
}

object { vase  pigment { color myColor }
               finish { ambient 0.5 diffuse 0.85 }
               scale 1.5
       }
```

## Sor - open



## lathe

- Similar to sor, but it is different in the way the surface is generated (mathematically speaking)

## Lathe - `linear_spline`

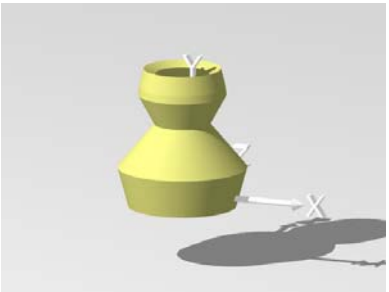
```
Create a lathe surface or revolution with the following
   points
#declare vase = lathe{ linear_spline 8,
   < 0.3, 0.00>,
   < 0.60, 0.00>,
   < 0.72, 0.44>,
   < 0.31, 0.93>,
   < 0.49, 1.26>,
   < 0.48, 1.35>,
   < 0.43, 1.56>,
   < 0.3, 1.50>
  }

object { vase  pigment { color myColor }
               finish { ambient 0.5 diffuse 0.85 }
               scale 1.5
       }
```

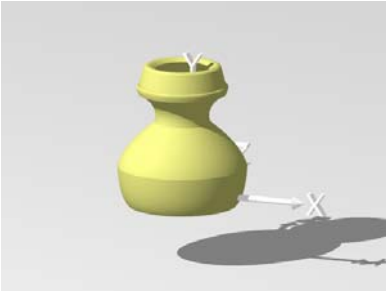
Total of point on the curve

Type of curve used linear_spline, quadratic_spline, cubic_spline

## Lathe - `linear_spline`



## Lathe - `quadratic_spline`




## Lathe - `cubic_spline`