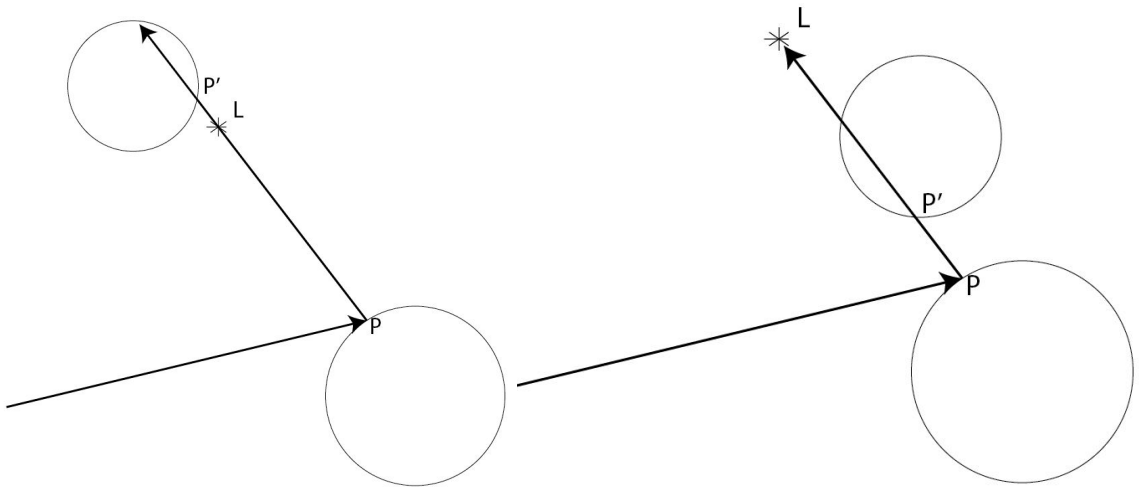
Introduction to Raytracing
Nicolas Bonneel

Lecture # 2

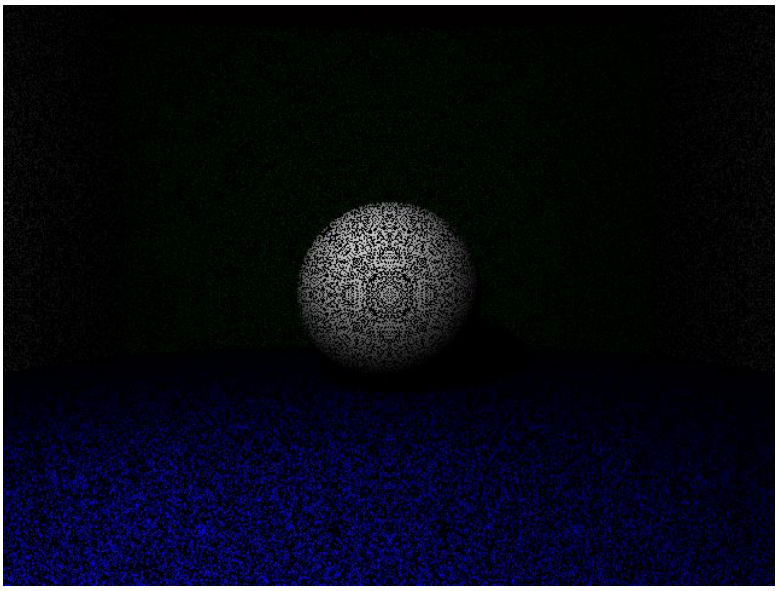
The objective of this lecture is to understand how to handle shadows, "pure specular" surfaces (i.e., perfect mirrors) and transparent surfaces. These effects have in common the need to generate secondary rays through the scene.

Cast Shadows

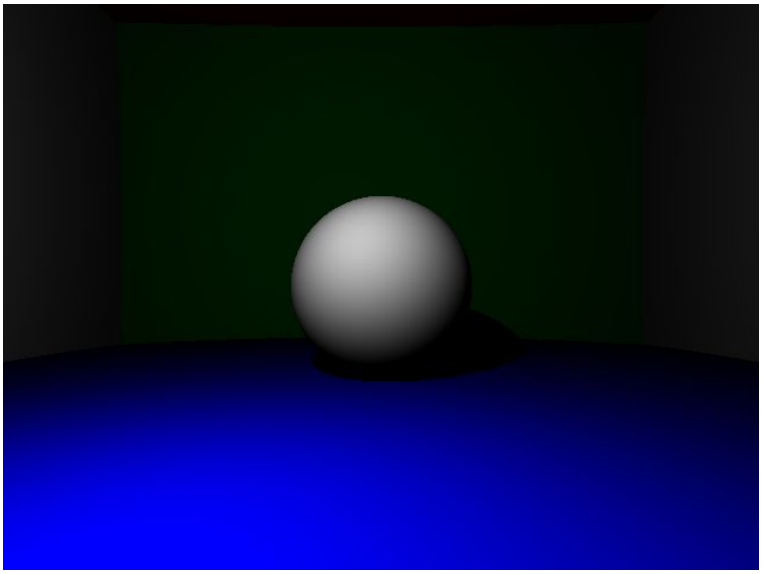
The principle is simple. When an intersection P is found, generate a secondary ray of light starting from P and going towards the light source L, computing again the intersections with the elements of the scene. If there is an intersection, the point of this intersection P' is determined. If the distance from P to P' is less than the distance from P to L, the point is shadowed and its color is black.
Hereafter, the configuration shown on the left will not produce a shadow: the distance from the intersection P to P' in the direction of the light is greater than the distance between P and L. In contrast, on the right, there will be a shadow and the pixel will be black.



In the scene above, this results in:



Visible noise is due to numerical inaccuracies. To solve them, simply launch the secondary ray from a point slightly off the surface, i.e., from the point P + epsilon * normal. We then obtain:



Specular surfaces

or "mirror" surfaces. The principle is similar, but now requires recursion.
Instead of sending a ray in the direction of the light L, we will simply reflect the incident ray around the normal of the object, and return the reflected color (the color of this new ray). It turns out that this new ray could reach a new mirror object, which itself could reflect a new object etc.

The algorithm to determine the color of a ray from the camera to the scene is then recursive, and is given by:

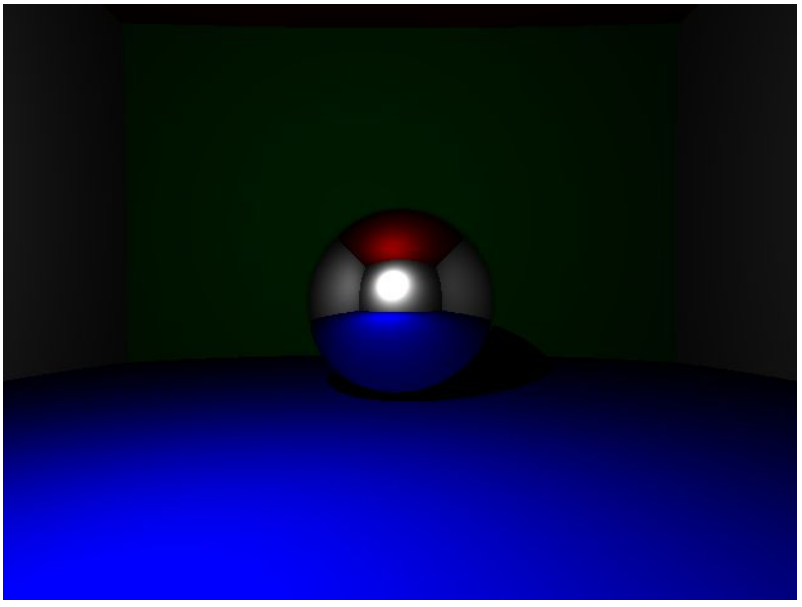
```
Color  getColor (Ray r, Integer bounce_number)
        Color result ← black
        If the surface is specular, and bounce_number> 0
                result ← getColor (reflect (r), bounce_number- 1)
        End if
        result ← result + diffuse part.
End Function getColor
```

where we initialize bounce_number with the maximum number of bounces (eg, 5) in order to avoid cases where two parallel surfaces make the same photon bounce endlessly.

The "reflect" function takes an incident ray, and changes its direction so that it reflects it around the normal. This is given by the formula $\vec{r} = \vec{i} - 2 < \vec{i}, \vec{n} > \vec{n}$ where $\vec{i}$ is the incident vector to the surface and $\vec{n}$ is the normal to the surface. You should be able to easily demonstrate this formula.

The result can be multiplied by a "specular color" if desired.

We then obtain the image (here, the diffuse color is black, and the specular color white):



Transparent surfaces

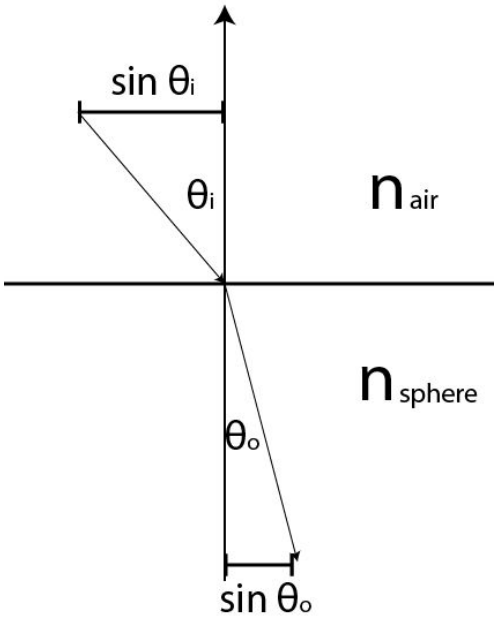
Handling transparent surfaces is similar, except the call to "reflect" which is replaced with a call to "refract".

The formula for refracting a ray is derived from Snell-Descartes's law:

$$n_{air} \sin\theta_i = n_{sphere} \sin\theta_o$$

where the n represent the refraction indices of the air and of the sphere, and i and o the incident and refracted angles.

We will find the formula for the refracted ray from the following figure:



$\vec{i} - <\vec{i}, \vec{n}>\vec{n}$ gives the tangential component of the incident ray (i.e., we removed the component along the normal n).
We multiply this component by $\frac{n_{air}}{n_{sphere}}$ to get the new tangential component of the refracted ray.
The refracted ray will have a tangential component: $\frac{n_{air}}{n_{sphere}}(\vec{i} - <\vec{i}, \vec{n}>\vec{n})$

We now seek its normal component. The refracted ray should have a unit norm, so the normal component is necessarily $-\sqrt{1-\sin^2\theta_o}\ \vec{n}$ According to Snell-Descartes formula, this corresponds to $-\sqrt{1-(\frac{n_{air}}{n_{sphere}}\sin\theta_i)^2}\ \vec{n}$, or $-\sqrt{1-\frac{n_{air}}{n_{sphere}}(1-<\vec{i},\vec{n}>^2)}\ \vec{n}$.

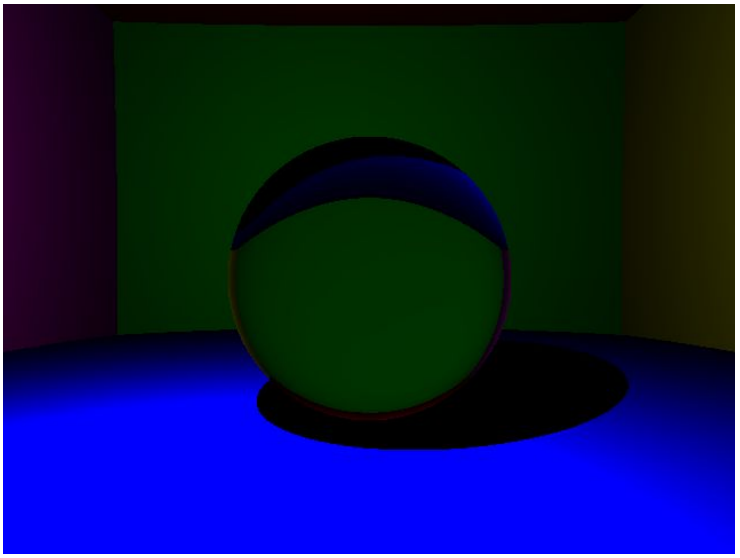
In short, the refracted ray is:

$$\vec{r} = \frac{n_{air}}{n_{sphere}}\vec{i} - (\frac{n_{air}}{n_{sphere}} <\vec{i},\ \vec{n}> + \sqrt{1-(\frac{n_{air}}{n_{sphere}})^2(1-<\vec{i},\vec{n}>^2)}\ )\ \vec{n}$$

If this value is complex, this means that there is no transmission, and reflection is total.

Note that this formula only applies if the normal is oriented toward the same side as the incident ray, i.e., $<\vec{i},\ \vec{n}> < 0$: the ray enters the sphere. Otherwise, the normal must be reversed and the roles of n_{air} and n_{sphere} are reversed too.

In the above scene (where I brought the camera closer to the sphere, colored side walls differently and increased the intensity of light), we get the result below.



Note the effect that "a lens reverses the image": the ground is reflected on the top of the sphere, and the walls seem reversed.

Additional Ref: Physically Based Rendering the book, Matt Pharr and Greg Humphrey (they got an Oscar for their work, useful for special effects in film), which is accompanied by an open source rendering engine.