# Photorealistic Rendering
# Lesson X

Joel Isaacson Bar-Ilan
Computer Science

22nd January 2006

## 1 Shadows and Textures

### 1.1 Shadows

Shadows are an important addition to 3-d visualization. Shadows add important visual clues to scenes.

Shadows vary as a function of the lighting environment. If the light sources are large or close to the obscuring object then the shadow has soft edges. If the light source is small or at a large distance from the object and shadow then the edge is hard.

### 1.2 Shadows on the ground plane

There is a relatively simple method of generating shadows that appear in the ground plane. This method can't handle shadows that are cause by one object shadowing another. In this method we project polygons that make up the object on the ground plane. A simple linear transformation will generate the shadow of a polygon on the ground plane. We will assume that the light source is at infinity at a direction $\mathbf{L} = (x_l, y_l, z_l)$ as shown in figure **??**. Here we have $\mathbf{P} = (x_p, y_p, z_p)$ and $\mathbf{S} = (x_s, y_s, z_s)$.

The vector $\mathbf{S} - \mathbf{P}$ is CO linear with $\mathbf{L}$, therefore writ en

$$\mathbf{S} = \mathbf{P} - \alpha\mathbf{L} \tag{1}$$

Assume that $z_s = 0$, we have:

$$0 = z_p - \alpha z_1 \tag{2}$$

$$\alpha = z_p/z_1 \tag{3}$$

and:

$$x_s = x_p - (z_p/z_1)x_1 \tag{4}$$

$$y_s = y_p - (z_p/z_1)y_1 \tag{5}$$

These equations can be as a homogeneous transformation:

$$[x^s, y^s, 0, 1] = [x_p, y_p, z_p, 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -x_l/z_l & -y_l/z_l & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

This ground plane shadow algorithm is sufficient in many cases the handle the need for shadowing in image generation.

### 1.3 Shadow algorithms

In general any one of the $n$ objects in a scene can block the lighting any other object. There are a number of algorithms that have been developed to treat shadowing.

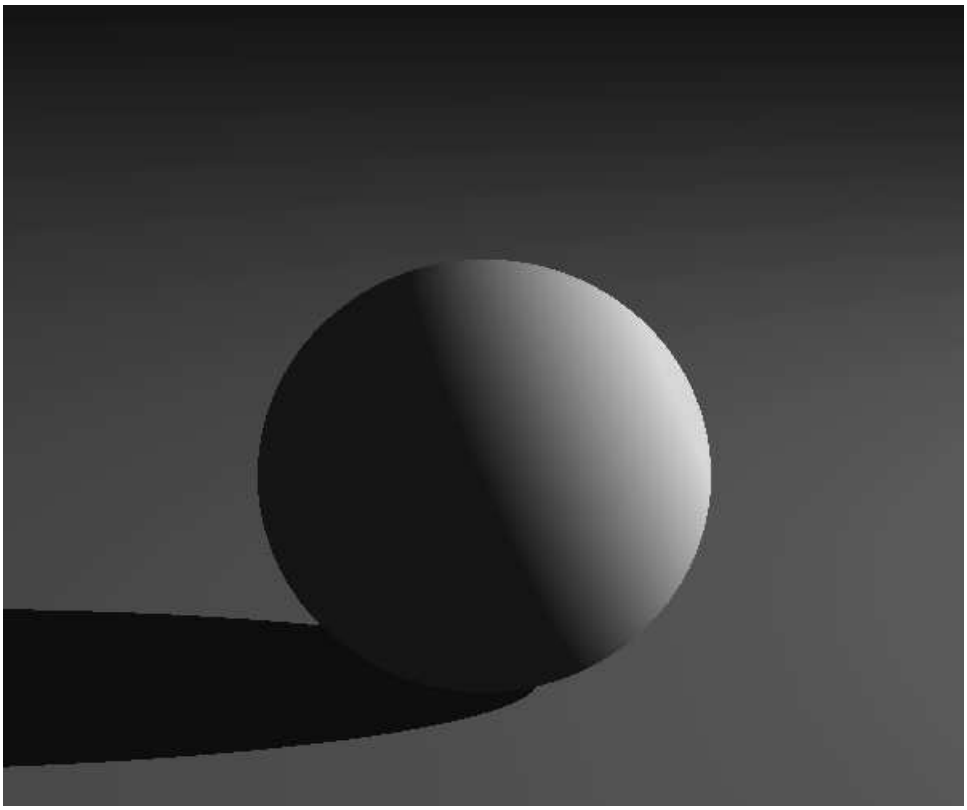

Figure 1: A shadowed sphere on a plane

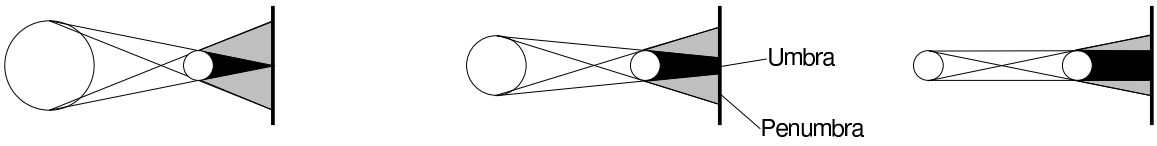


Figure 2: The umbra-penumbra of the shadowed area

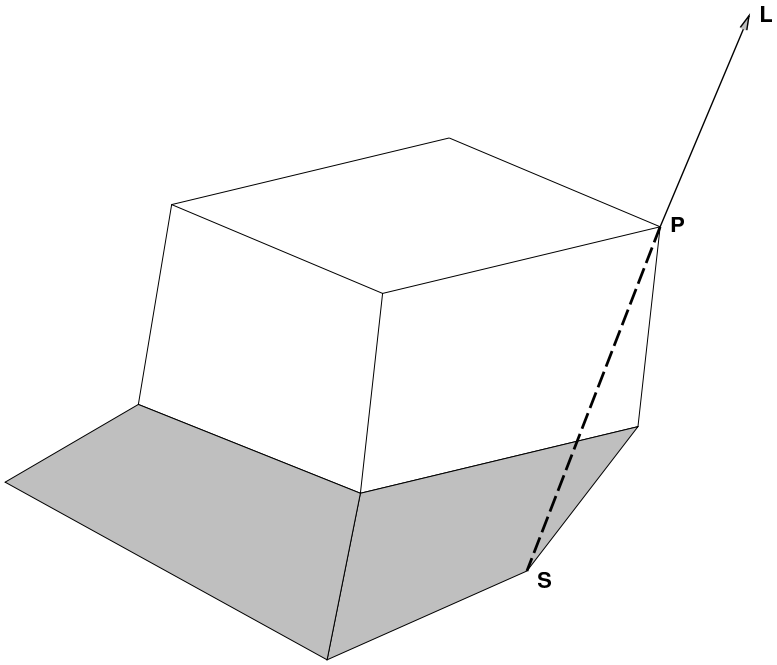


Figure 3: Ground plane projection for a single object

### 1.3.1 Scan line algorithms

Adding shadows to a scan line algorithm requires a preprocessing step. We build up a data structure that links all polygon that may shadow a given polygon. This can be done by projecting all polygons onto a sphere centered at the light source. This is done by translating the origin of the coordinates to the light source and expressing the $(x, y, z)$ of the vertices as $(\theta, \psi, r)$. We then can check overlap in the $(\theta, \psi)$ coordinates. The polygons that overlap and are farther from the light source than the other polygons are candidates for partial shadowing. The standard scan line algorithm is used. We have to deal differently with polygons that have shadows on them. The pixel intensity is modulated by the shadow.

In Fig. 4 we see a scan line that passes through two polygons and a shaded area. Along the scan line we pass from left to right and pass through the following segments:

1. Polygon A is visible.
2. Polygon B is visible.
3. Polygon B is shadowed by polygon A.
4. Polygon B is visible.

### 1.3.2 Shadow volumes

This algorithm creates a shadow volume object which is the object swept out by the shadow of an object in the view volume (Fig 5).

There are several possible rendering algorithms that can be use these shadow object to render a scene. The simplest is a scheme that sorts the polygons by the distance from the view point. If the visible part of the polygon is within a shadow object then it is rendered as a shaded surface. A count of shadow polygons encountered is keep for each shadow polygon entered, front facing shadowpolygons are given a value of +1 and back-facing shadow polygons are given a value of -1. If the this value is positive at the polygon to be shaded then this polygon is shaded. If this value is zero then the polygon is lighted.

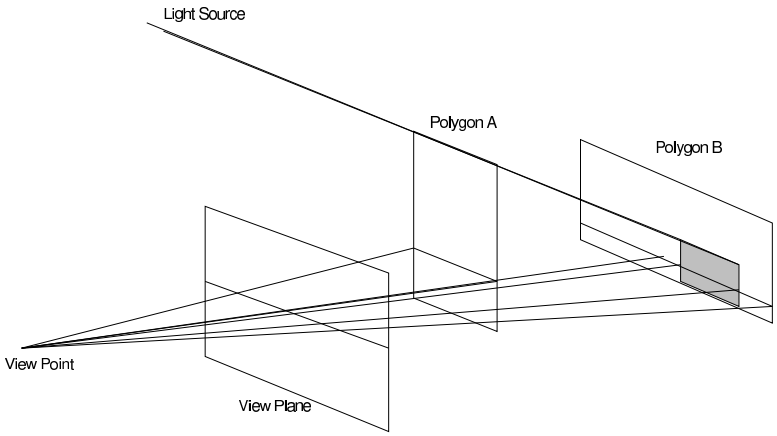


Figure 4: One polygon shading another polygon

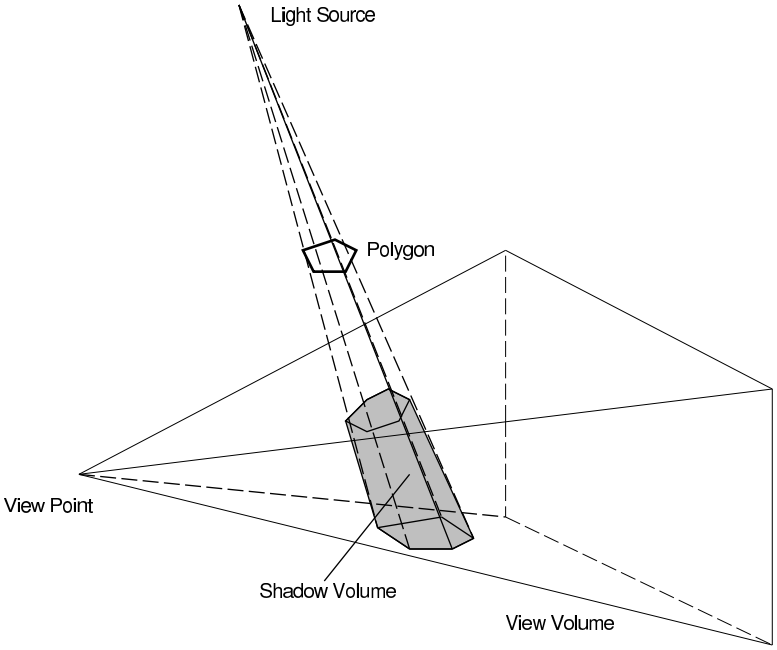


Figure 5: The shadow volume generated by a polygon in a view volume

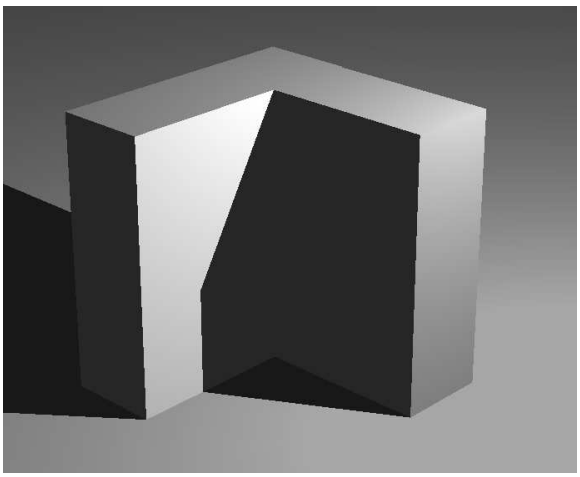

Figure 6: A shaded scene

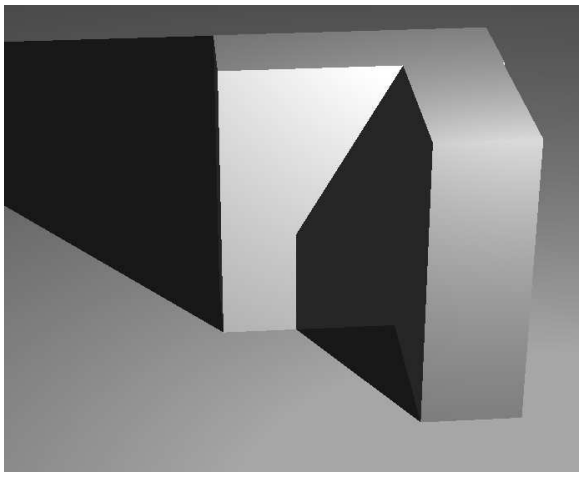

Figure 7: A different view of the same scene

#### 1.3.3 Two pass Z-buffer shading (shadow map)

This method is based on the fact that if the view point and light source are at the same point then only lighted areas are seen. We then can perform a rendering of the scene with the view point set at the light source. The Z-buffer will then contain the location of the lighted polygons. We then can save this Z-buffer. This will be called the shadow map. We then perform a rendering from the actual view point. When we render a point on a surface we can consult the shadow map. If the z-value of this point in the shadow map corresponds to the point on the surface then we render it as lighted otherwise we render it as shaded.

#### 1.3.4 Shadow polygon data base

The previous approach can be used to create a data structure of lighted and shaded polygons. The set of polygons that create the scene is called $\mathbf{P}$. We then perform a hidden surface algorithm with the view point at the light source. This will give us a set of lighted polygons, $\mathbf{L}$. The set of shadowed polygons is just $\mathbf{U} = \mathbf{P} - \mathbf{L}$. We then create a scene that is made up of $\mathbf{L}$ with the original reflectance parameters and $\mathbf{U}$ with reflectance parameters set to make the polygons seem to be shaded. This data structure (and the previous shadow map technique) can be reused to render multiple viewpoints of the same scene.

### 1.4 Texture Mapping

Texture mapping is a relatively easy way of getting effects that are more interesting than the "shiny plastic" effects produced by the simple Phong model. The basic idea is to modulate the reflectance parameters of the surface of the object being rendered. The texture used is typically a photograph of a natural object or some synthetically generated image.

#### 1.4.1 Object attributes modified by texture

Many different attributes can be modulated over the surface of an object. In general the most common thing to modulate is the diffuse reflection coefficients (color). The modulation of the specular reflection coefficients is also common and is usually called environmental mapping.

#### 1.4.2 The 2-d to 3-d mapping functions

Mapping a two dimensional texture onto a three dimensional object is both non-trivial and in general impossible. Just think of the hundreds of years cartographers have been struggling with mapping the earth's three dimensional surface onto a two dimensional map without complete success. If the surface is a parametric surface $(S(u, v)$ we can in some

cases directly map the texture ($T(x, y)$ using a simple mapping of $x$ to $u$ and $y$ to $v$. We of course might have problems of matching the pattern on the edges. This is similar to the problem of matching wallpaper when applied to a wall.

### 1.4.3 Two-part mapping

This technique performs the mapping in two stages. The first stage is a mapping of the 2-d texture to a simple 3-d intermediate surface that can be thought of as encompassing the view volume. This transformation is called the **S** mapping. The second mapping is a mapping of this 3-d intermediate surface to the surface of the object. This transformation is called the **O** mapping.

### 1.4.4 The S transform

The simple object that the texture is mapped to is usually a sphere, cylinder or box. For a sphere on the the many cartographic projections (Mercador etc.) can be used. For a cylinder you can just wrap the pattern around the cylinder as if you are wrapping a tube as a gift. A cube can also simply wrapped with a pattern.

### 1.4.5 The O transform

The general idea is that the intermediate object, created with transform **S**, is scaled up to encompass the viewing volume. Very loosely you then shrinkwrap (transform **O**) the enclosing intermediate object until the texture sticks to the surface of the object to be texture mapped. There are many different ways to map texture values of the object into the intermediate object.

1. The intersection of the reflected ray and the intermediate surface gives the mapping. This is commonly known as environment mapping. This can be used to give the effect of reflections of an object surroundings onto the object. The effect of much more computationally intensive rendering called *ray-tracing* can be sometimes obtained for a fraction of the computational cost.
2. The intersection of the surface normal of the object with the intermediate surface.
3. The projection of the line going through the object centroid and the object surface with the intermediate surface gives the texture value.
4. The intersection of the intermediate surface normal and the object's surface.

These four cases are shown in Fig. 8

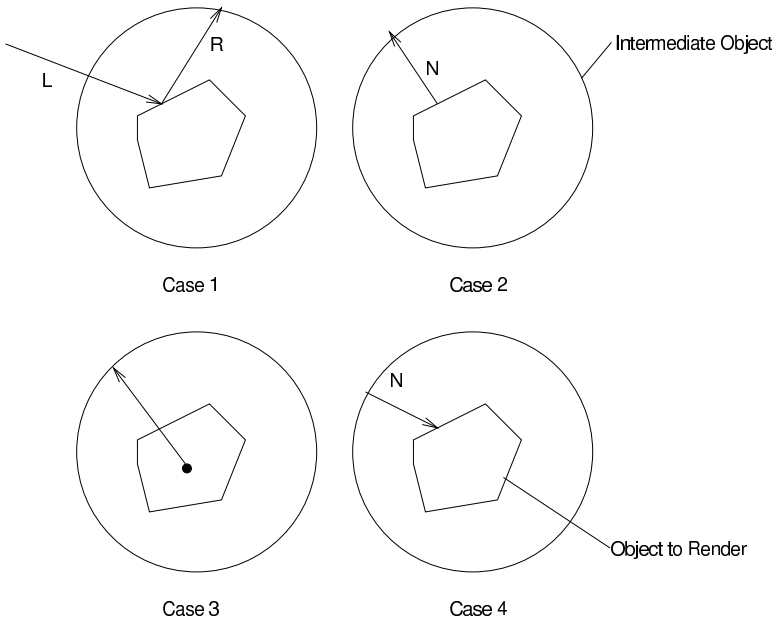


Figure 8: The four **O** mappings.