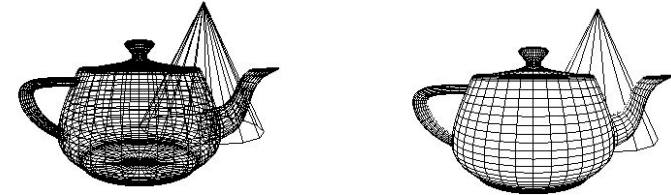


# Viewing

321190  
2007년 봄학기  
5/8/2007  
박경신

## Hidden Surface

- 은면(Hidden surfaces)은 occlusion depth cue를 제공한다.
- 컴퓨터 그래픽스에서, 가려짐 (occlusion)이란 용어는 뷰포트로부터 가까운 물체가 뷰포트에서 멀리있는 물체를 가리는 것을 말한다.
- 그래픽스 파이프라인에서 occlusion culling으로 셰이딩(shading)과 래스터화(rasterization)하기 전에 은면 제거(hidden surface removal)을 한다.



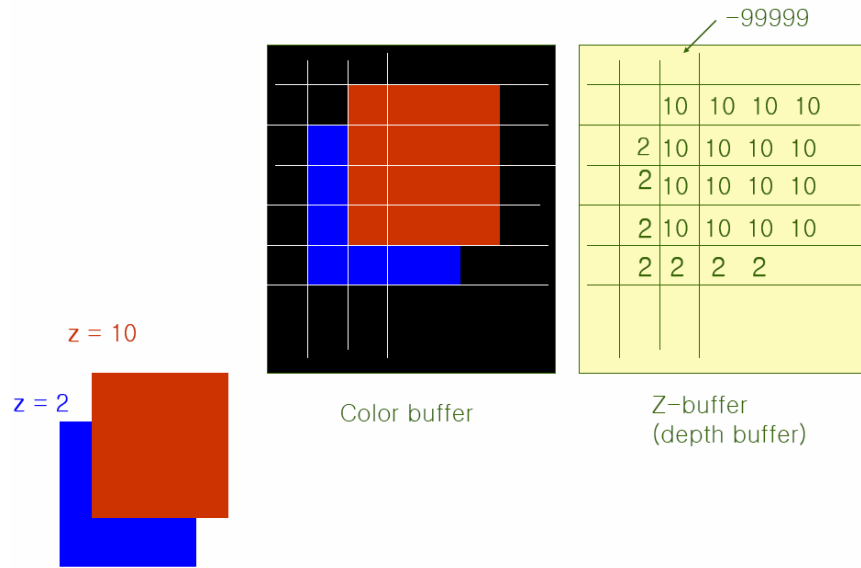
## Hidden Surface Removal

- 은면 제거 (Hidden Surface Removal) 알고리즘
  - 객체 공간기법 - 객체나 객체 부분들을 서로 비교하여 전체적으로 어느 면과 선이 보이지 않는 것인지 결정
    - 깊이 정렬 알고리즘 (Depth-sorting algorithm) - 폴리곤의 각 면을 깊이에 따라 정렬한 뒤, 면 것부터 투영하여 그린다. Painter's algorithm라고도 불린다.
    - Binary Space Partitioning (BSP) tree - BSP tree를 사용하여 관측 방향에 따라 front, back을 구분하여 공간을 계속적으로 분할한다.
  - 이미지 공간 기법 - 투영 과정의 일부분으로 동작하여, 각 투영선 위의 객체 화소 위치에서 점 단위로 가시성이 결정
    - Z-buffer (depth buffer) - 가장 일반적으로 사용되는 이미지 공간 기법으로, 물체의 가시성을 화소 단위로 조사하여 z (깊이) 값이 가장 작은 평면의 값을 그린다. Z값을 저장하는 깊이버퍼 (z-buffer)가 필요하다.
    - Ray-casting - 시점에서 투영면의 각 화소를 통해 빛 (ray)를 투사하고, 이 빛과 처음으로 만나는 객체를 선택하여 해당 픽셀을 그린다. 임의의 곡면과 같은 표면에서 효과적인 은면 제거 방법이다.

## Hidden Surface Removal

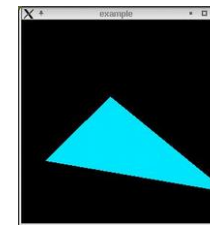
- 은면 제거가 적용된 다양한 방법
  - 깊이 정보 테스트
    - glEnable(GL\_DEPTH\_TEST);
  - 표면/이면 제거
    - glEnable(GL\_CULL\_FACE);
    - glCullFace(GL\_FRONT);
    - glCullFace(GL\_BACK);

## Z-buffer

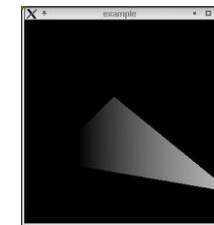


## Z-buffer

- 폴리곤 렌더링이란 결국 픽셀로 채워지는 것을 의미한다.
- 컬러 버퍼 (Color buffer)는 그리고자 하는 픽셀당 RGB 색 정보를 가진다.
- 깊이 버퍼 (Z-buffer, depth buffer)는 그리고자 하는 픽셀당 깊이 정보 (depth value)를 가진다.



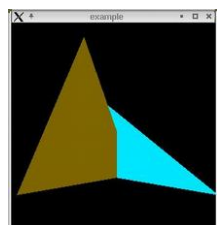
Color buffer



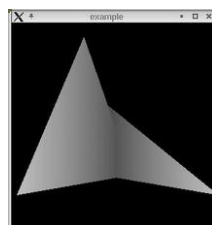
Depth buffer

## Z-buffer Algorithm

- Z-buffer algorithm은 새로운 픽셀을 그릴 때마다, 새로운 깊이 정보를 깊이 버퍼 (z-buffer) 안에 있는 깊이(depth) 정보와 비교한다.
- 폴리곤 (Polygons)은 어떠한 방향에서도 그려질 수 있으며 교차할 수도 있다.



Color buffer



Depth buffer

## OpenGL Z-buffering

- OpenGL에서 z-buffer를 사용하려면 먼저 깊이 버퍼를 초기화하고, 깊이 정보 테스트를 활성화한다.
 

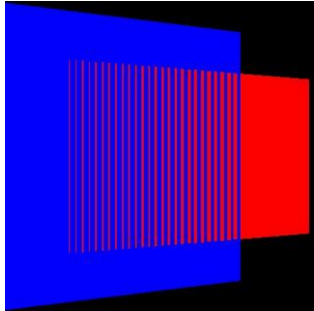
```
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB|GLUT_DEPTH);
glEnable(GL_DEPTH_TEST);
```
- 매 프레임마다 깊이 버퍼를 지운다.
 

```
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
```
- 정육면체와 같은 객체의 경우 관측자로부터 멀어지는 방향을 향하는 모든 면을 제거하고자 할 때 사용한다.
 

```
glEnable(GL_CULL);
```

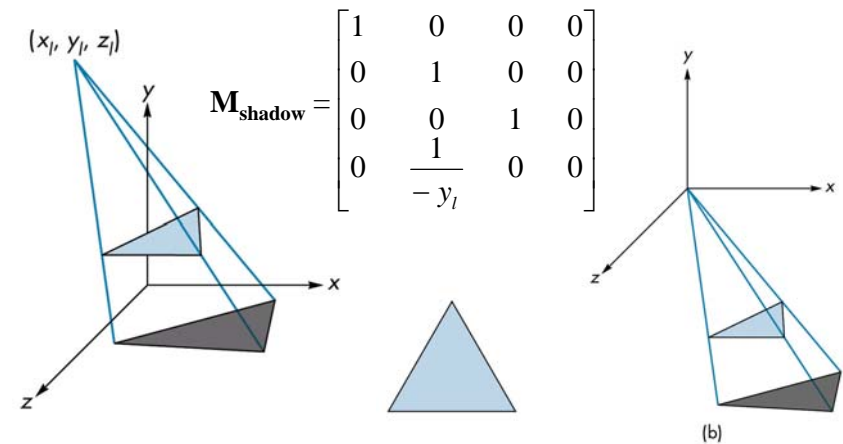
## Depth Fighting

- Z-buffer의 깊이값은 한정된 해상도를 갖고 있다.
- 깊이 버퍼에서 아주 가까운 깊이값(depth value)을 가지는 폴리곤의 중첩(overlap)은 "depth-fighting"을 만든다.
- 폴리곤이 그려질 때 부동 소수점 반올림 에러 (floating point round-off errors)때문에 생기는 현상으로, 폴리곤 임의의 부분이 서로 렌더링하려는 현상이다.



## Projections and Shadows

- 투영행렬을 사용한 간단한 그림자 생성

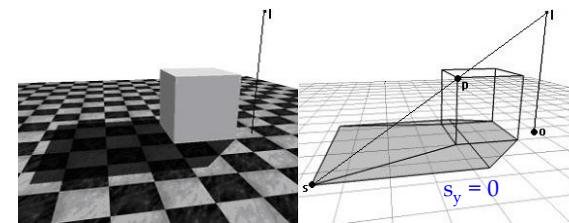


## Projections and Shadows

```

GLfloat m[16]           // shadow projection matrix
m[0] = m[5] = m[10] = 1.0;
m[7] = -1.0/y_l;
glColor3fv(polygon_color);
glBegin(GL_POLYGON);
.. // draw polygon normally
glEnd();
glMatrixMode(GL_MODELVIEW);
glPushMatrix();
glTranslatef(x_l, y_l, z_l); // translate back
glMultMatrix(m);           // project shadow matrix
glTranslatef(-x_l, -y_l, -z_l); // move light to the origin
glColor3fv(shadow_color);
glBegin(GL_POLYGON);
.. // draw the polygon again
glEnd();
glPopMatrix();
    
```

## Planar Shadow [J. Blinn, 88]



$$\frac{s_x - l_x}{p_x - l_x} = \frac{l_y - 0}{l_y - p_y}$$

$$s_x = \frac{l_y(p_x - l_x)}{l_y - p_y} + l_x$$

$$t = \frac{l_y}{l_y - p_y}$$

$$\begin{bmatrix} s_x \\ 0 \\ s_z \\ 1 \end{bmatrix} = \begin{bmatrix} l_y & -l_x & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -l_z & l_y & 0 \\ 0 & -1 & 0 & l_y \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

$$\mathbf{s} = \mathbf{l} + t(\mathbf{p} - \mathbf{l})$$

$$\mathbf{n} \cdot \mathbf{s} + d = 0$$

$$t = \frac{l_y}{l_y - p_y}$$

## Planar Shadow [J. Blinn, 88]

$$s = l + \frac{l_y}{l_y - p_y}(p - l) \quad s_x = l_x + \frac{l_y}{l_y - p_y}(p_x - l_x) \quad s_z = l_z + \frac{l_y}{l_y - p_y}(p_z - l_z)$$

$$s_x = l_x + \frac{l_y}{l_y - p_y}(p_x - l_x) = \frac{l_x(l_y - p_y) + l_y(p_x - l_x)}{l_y - p_y} = \frac{l_x(l_y - p_y) + l_y(p_x - l_x)}{l_y - p_y}$$

$$s_y = 0 = \frac{l_y p_x - l_x p_y}{l_y - p_y} = \frac{-l_x p_y + l_y p_x}{l_y - p_y}$$

$$s_z = l_z + \frac{l_y}{l_y - p_y}(p_z - l_z)$$

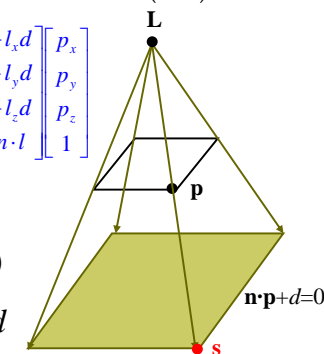
$$\begin{bmatrix} s_x \\ 0 \\ s_z \\ 1 \end{bmatrix} = \begin{bmatrix} l_y & -l_x & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -l_z & l_y & 0 \\ 0 & -1 & 0 & l_y \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

$$\begin{aligned} s_x &= l_y p_x - l_x p_y + 0 p_z \\ s_y &= 0 \\ s_z &= 0 p_x - l_z p_y + l_y p_z \\ w &= 0 p_x - p_y + 0 p_z + l_y \end{aligned}$$

## Projection Shadow

$$\begin{bmatrix} s_x \\ s_y \\ s_z \\ 1 \end{bmatrix} = \begin{bmatrix} n \cdot l + d - l_x n_x & -l_x n_y & -l_x n_z & -l_x d \\ -l_y n_x & n \cdot l + d - l_y n_y & -l_y n_z & -l_y d \\ -l_z n_x & -l_z n_y & n \cdot l + d - l_z n_z & -l_z d \\ -n_x & -n_y & -n_z & n \cdot l \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

점광원 (점 L)



$$s = l + t(p - l)$$

$$\mathbf{n} \cdot \mathbf{s} + d = 0$$

$$\mathbf{n} \cdot (l + t(p - l)) + d = 0$$

$$\mathbf{n} \cdot l + t(\mathbf{n} \cdot (p - l)) = -d$$

$$t(\mathbf{n} \cdot p - \mathbf{n} \cdot l) = -d - \mathbf{n} \cdot l$$

$$t = \frac{-d - \mathbf{n} \cdot l}{\mathbf{n} \cdot p - \mathbf{n} \cdot l} \quad \therefore \mathbf{s} = l + \left[ \frac{\mathbf{n} \cdot l + d}{\mathbf{n} \cdot l - \mathbf{n} \cdot p} \right] (\mathbf{p} - l)$$

## Projection Shadow

$$s = l + \frac{n \cdot l + d}{n \cdot p - n \cdot l}(p - l) \quad s_x = l_x + \frac{n \cdot l + d}{n \cdot l - n \cdot p}(p_x - l_x)$$

$$s_x = l_x + \frac{n \cdot l + d}{n \cdot p - n \cdot l}(p_x - l_x) = \frac{l_x(n \cdot l - n \cdot p) + (n \cdot l + d)p_x - (n \cdot l + d)l_x}{n \cdot l - n \cdot p}$$

$$s_y = l_y + \frac{n \cdot l + d}{n \cdot p - n \cdot l}(p_y - l_y) = \frac{l_x n \cdot l - l_x n_x p_x - l_x n_y p_y - l_x n_z p_z + (n \cdot l + d)p_x - l_x n \cdot l - l_x d}{-n_x p_x - n_y p_y - n_z p_z + n \cdot l}$$

$$s_z = l_z + \frac{n \cdot l + d}{n \cdot p - n \cdot l}(p_z - l_z) = \frac{(n \cdot l + d - l_x n_x)p_x - l_x n_y p_y - l_x n_z p_z - l_x d}{-n_x p_x - n_y p_y - n_z p_z + n \cdot l}$$

$$\begin{bmatrix} s_x \\ s_y \\ s_z \\ 1 \end{bmatrix} = \begin{bmatrix} n \cdot l + d - l_x n_x & -l_x n_y & -l_x n_z & -l_x d \\ -l_y n_x & n \cdot l + d - l_y n_y & -l_y n_z & -l_y d \\ -l_z n_x & -l_z n_y & n \cdot l + d - l_z n_z & -l_z d \\ -n_x & -n_y & -n_z & n \cdot l \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

## Projection Shadow Matrix

$$\begin{bmatrix} s_x \\ s_y \\ s_z \\ 1 \end{bmatrix} = \begin{bmatrix} n \cdot l - l_x n_x & -l_x n_y & -l_x n_z & -l_x n_w \\ -l_y n_x & n \cdot l - l_y n_y & -l_y n_z & -l_y n_w \\ -l_z n_x & -l_z n_y & n \cdot l - l_z n_z & -l_z n_w \\ -l_w n_x & -l_w n_y & -l_w n_z & n \cdot l - l_w n_w \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

투영평면  $n = [n_x, n_y, n_z, n_w]$

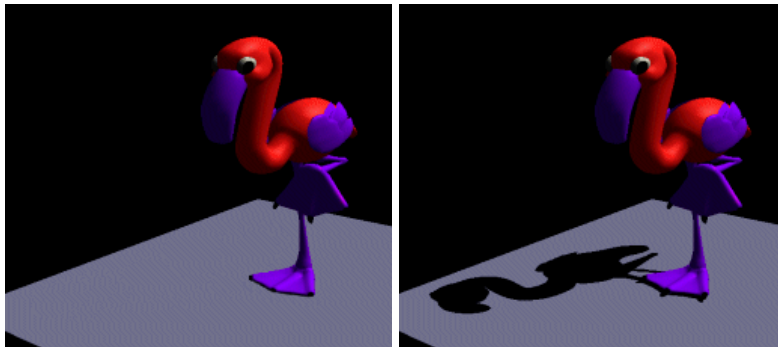
광원  $l = [l_x, l_y, l_z, l_w]$  where  $l =$ 평행광원이면  $l_w = 0$ ,  $l =$ 점광원이면  $l_w = 1$

## Projection Shadow Matrix

```
// create a shadow matrix that will project the desired shadow
void shadowmatrix(GLfloat shadowMat[16], GLfloat plane[4], GLfloat lightpos[4])
{
    GLfloat dot; // dot product of light position and ground plane normal
    dot = plane[0] * lightpos[0] + plane[1] * lightpos[1] + plane[2] * lightpos[2]
        + plane[3] * lightpos[3];
    shadowMat[0] = dot - lightpos[0] * plane[0];
    shadowMat[4] = 0.f - lightpos[0] * plane[1];
    shadowMat[8] = 0.f - lightpos[0] * plane[2];
    shadowMat[12] = 0.f - lightpos[0] * plane[3];
    shadowMat[1] = 0.f - lightpos[1] * plane[0];
    shadowMat[5] = dot - lightpos[1] * plane[1];
    shadowMat[9] = 0.f - lightpos[1] * plane[2];
    shadowMat[13] = 0.f - lightpos[1] * plane[3];
    shadowMat[2] = 0.f - lightpos[2] * plane[0];
    shadowMat[6] = 0.f - lightpos[2] * plane[1];
    shadowMat[10] = dot - lightpos[2] * plane[2];
    shadowMat[14] = 0.f - lightpos[2] * plane[3];
    shadowMat[3] = 0.f - lightpos[3] * plane[0];
    shadowMat[7] = 0.f - lightpos[3] * plane[1];
    shadowMat[11] = 0.f - lightpos[3] * plane[2];
    shadowMat[15] = dot - lightpos[3] * plane[3];
}
```

```
void renderShadow(void) {
    renderOccluders(); // 1. Step - Render all object that cast shadows
    glClear(GL_STENCIL_BUFFER_BIT); // clear stencil buffer
    glEnable(GL_STENCIL_TEST); // 2. Step - Render all receivers
    glStencilOp(GL_KEEP, GL_KEEP, GL_REPLACE);
    while (pReceiver) {
        glStencilFunc(GL_ALWAYS, pReceiver->getId(), ~0);
        render(pReceiver);
        pReceiver = pReceiver->getNext();
    }
    glDisable(GL_TEXTURE_2D); // 3. For every receiver plane (pReceiver)
    glDisable(GL_DEPTH_TEST);
    glEnable(GL_BLEND);
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
    glColor4f(0.0f, 0.0f, 0.0f, 0.5f); // shadow color
    glStencilOp(GL_KEEP, GL_KEEP, GL_ZERO);
    while (pReceiver) {
        glStencilFunc(GL_EQUAL, pReceiver->getId(), ~0);
        glPushMatrix();
        glmMultMatrixf((GLfloat *) pReceiver->getShadowMatrix());
        renderOccludersFast(); // texturing, use constant shading only (GL_FLAT) etc
        glPopMatrix();
        pReceiver = pReceiver->getNext();
    }
    glDisable(GL_BLEND);
    glDisable(GL_STENCIL_TEST);
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_LIGHTING);
    glEnable(GL_TEXTURE_2D);
}
```

## Shadow



Render without shadow

Render with shadow

## References

- [http://en.wikipedia.org/wiki/Hidden\\_surface\\_removal](http://en.wikipedia.org/wiki/Hidden_surface_removal)
- [http://en.wikipedia.org/wiki/Binary\\_space\\_partitioning](http://en.wikipedia.org/wiki/Binary_space_partitioning)
- [http://en.wikipedia.org/wiki/Painter%27s\\_algorithm](http://en.wikipedia.org/wiki/Painter%27s_algorithm)
- <http://en.wikipedia.org/wiki/Z-buffering>
- <http://en.wikipedia.org/wiki/Z-fighting>
- <http://www.shadowstechniques.com/blinn.html>
- <http://www.devmaster.net/articles/shadowprojection/>