CMSC 427: Chapter 8
Real-time Shadows and Reflection

Reading: Covered only briefly in Shirley’s book. Resources can be found on the web (e.g., OpenGL tutorials or Wikipedia).
Overview:
- Shadows: Real-time shadows, light-maps, shadow Z-buffer, shadow volumes.
- Real-time reflection in OpenGL.

Elements of Realistic Rendering

Realism in Rendering:
- Perspective projection
- Illumination and shading
- Texture mapping and surface detail
- Shadows, Reflection, and other effects
- Hidden surface removal
- Color
**Why Shadows?**

Shadows give us important **visual cues** about 3D object placement and motion.
- See the film clip from Univ. of Minn. Changing the shadow position creates an entirely different impression of object position. [http://gandalf.psych.umn.edu/~kersten/kersten-lab/demos/shadows.html](http://gandalf.psych.umn.edu/~kersten/kersten-lab/demos/shadows.html)

Shadows also add **realism**.

OpenGL **does not** implement shadows, so you need to **generate them explicitly**.

---

**Hard and Soft Shadows**

- **point light source**
- **area light source**
- **umbra**
- **penumbra**
- **occluder**

Images courtesy, Eric Haines and Tomas Moeller
Overview

• Shadow painting
• Light Maps
• Shadow Z-buffer
• Shadow Volumes
• Real-Time Reflection

Real-time Shadow Rendering

Painting Shadow Polygons on the Ground: (Blinn 88)
- Given point \( p = (x_p, y_p, z_p) \) and a light source \( L \), we want to compute its shadow point \( s = (x_s, y_s, z_s) \), on the plane \( z = 0 \).

Simplifying Assumptions:
- Point light sources, so only hard shadows.
- Shadows are cast onto planar surfaces.
- Light source at infinity in the direction \( v_L = (x_L, y_L, z_L) \).

Derivation:
- Shadow point \( s \) lies on the ray defined by \( p \) and \(-v_L\), that is, \( s = p - \alpha v_L \), for some \( \alpha > 0 \).
- Since the shadow lies on the plane \( z = 0 \), we have

\[
\begin{align*}
z_s &= 0 \Rightarrow \alpha = z_p / z_L, \\
s &= (x_s, y_s, z_s) = \left(x_p - \frac{z_p}{z_L} x_L, y_p - \frac{z_p}{z_L} y_L, 0\right)
\end{align*}
\]
Real-time Shadow Rendering

Shadow Projection Matrix:

\[
\begin{bmatrix}
    x_s \\
    y_s \\
    z_s \\
    1
\end{bmatrix}
= \begin{bmatrix}
    1 & 0 & -x_L/z_L & 0 \\
    0 & 1 & -y_L/z_L & 0 \\
    0 & 0 & 0 & 0 \\
    0 & 0 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
    x_p \\
    y_p \\
    z_p \\
    1
\end{bmatrix}
\]

Painting Shadows on the Ground:

Step 1: Draw objects with using the standard color.
Step 2: Render objects again, but using the above shadow transformation and draw (or blend with) a darker shadow color.

Example:

```c
void drawGround()
{
    glPushMatrix();
    glMultMatrixf(shadowMatrix);
    drawObjects(usingShadowColor);
    glPopMatrix();
}
```

Real-time Shadow Rendering

Local Light Sources: If the light source is not at infinity but at a local point \( L = (x_L, y_L, z_L) \), we can use the following projective transformation (Blinn 88):

\[
\begin{bmatrix}
    x_s \\
    y_s \\
    z_s \\
    1
\end{bmatrix}
= \begin{bmatrix}
    z_L & 0 & -x_L & 0 \\
    0 & z_L & -y_L & 0 \\
    0 & 0 & 0 & 0 \\
    0 & 0 & -1 & z_L
\end{bmatrix}
\begin{bmatrix}
    x_p \\
    y_p \\
    z_p \\
    1
\end{bmatrix}
\]

Note: Unlike the case for light source at infinity, this is a projective transformation, and so needs to be applied in `GL_PROJECTION` mode.
Real-time Shadow Rendering: Issues

Multiple Light Sources:
- For each light source:
  - Apply the shadow transformation for this light source,
  - Disable lighting and draw object in the shadow color.

z-Buffer conflicts:
- Shadows are drawn directly on the ground polygon, so ties in the z-buffer cause problems.
- Simple geometry: Disable depth test when drawing shadows.
- Complex geometry: Shadow polygons are offset a tiny amount:
  ```
  glEnable ( GL_POLYGON_OFFSET_FILL );
  glPolygonOffset ( 1.0f, 1.0f );
  // display the shadow polygon here
  glDisable ( GL_POLYGON_OFFSET_FILL );
  ```

How to avoid shadows being drawn outside the ground polygon?
- Use OpenGL’s stencil buffer to restrict drawing area.
- Fill in the stencil pixels where drawing is allowed (say with 1).
- Draw shadow but filter to areas where stencil buffer value is 1.
Aside: Stencil Buffer

Stencil Buffer:
- A pixel buffer which does not store colors, but rather values that affect whether pixels are rendered.
- Used for masking operations.
- When enabled, each pixel is applied to a stencil test.
  - If it fails, it is discarded.
  - If it passes, it is continues to depth-testing for possible rendering.
- The test depends on a value in the stencil buffer, and a reference value. (Independent of color and depth information.)
- If the test passes, a stencil operation is then performed.

Usage:
- Specify stencil test and/or reference value (glStencilFunc).
- Specify stencil operation (glStencilOp).
- Enable stenciling (glEnable(GL_STENCIL_TEST)).
- Draw objects.

```c
void glStencilFunc ( GLenum func, GLint ref, GLuint mask )

ArgumentNullException: The parameter is null.

ArgumentException: The parameter is invalid.

void glStencilOp ( GLenum sf, GLenum df, GLenum dp )

ArgumentNullException: The parameter is null.

ArgumentException: The parameter is invalid.
```

Chapter 8, Slide 11
Copyright © D. M. Mount and A. Varshney

Aside: Stencil Buffer

```c
void glStencilFunc ( GLenum func, GLint ref, GLuint mask )

(func): The test function. GL_ALWAYS, GL_NEVER, GL_LESS,
GL_LESS_EQUAL, GL_GREATER, GL_GREATER_EQUAL,
GL_EQUAL, GL_NOTEQUAL. (Default: GL_EQUAL).

(ref): Reference value for the stencil test. (Default: 0).

(mask): Mask that is "and"-ed with both the reference value and the
stored stencil value when the test is done. The initial value is all 1's.

void glStencilOp ( GLenum sf, GLenum df, GLenum dp )

(sf): Action applied to stencil buffer when the stencil test fails.
GL_KEEP, GL_ZERO, GL_REPLACE, GL_INCR, GL_INCR_WRAP,
GL_DECR, GL_DECR_WRAP, and GL_INVERT. (Default: GL_KEEP.)

(df): Action when the stencil test passes, but the depth test fails.
(Defaults: GL_KEEP.)

(dp): Action when both the stencil test and the depth test pass.
(Defaults: GL_KEEP.)

Chapter 8, Slide 12
Copyright © D. M. Mount and A. Varshney
```
Aside: Stencil Buffer

Examples:

Draw an object, set the stencil buffer to 1's for each pixel in which the object is drawn.

```c
glEnable ( GL_STENCIL_TEST ) // enable stenciling
glDisable ( GL_DEPTH_TEST ) // use these if object is not to be drawn to the screen
glColorMask ( 0, 0, 0, 0 ) // …drawn to the screen
glStencilFunc ( GL_ALWAYS, 1, 1 ) // set reference value to 1
glStencilOp ( GL_KEEP, GL_KEEP, GL_REPLACE ) // store 1 for each pixel
drawObject ( ) // draw object to stencil buffer
glEnable ( GL_DEPTH_TEST ) // restore regular drawing
glColorMask ( 1, 1, 1, 1 )
```

Draw an object, but show only those pixels where the stencil buffer value is 1.

```c
glStencilFunc ( GL_EQUAL, 1, 1 ) // draw only if stencil value = 1
glStencilOp ( GL_KEEP, GL_KEEP, GL_KEEP ) // no change to stencil
drawObject( ) // draw the object
```

Overview

- Shadow painting
- Light Maps
- Shadow Z-buffer
- Shadow Volumes
- Real-Time Reflection
Light Maps

**Light Maps:** Another way to generate shadows.
- Fast and easy to implement.
- Capable of generating complex light/shadow patterns.
- Best for static lights and static scenes and viewer independent lighting.

**Idea:**
- Compute the view-independent lighting of a scene (offline).
- Store it as a 2-d texture map.
- Paint the lighting onto your surfaces through texture mapping.

**Discussion:**
- Light maps are reasonably effective even when used at low resolutions (since they do not usually need high-frequency detail).
- For greatest efficiency, cluster similarly lit polygonal patches together (Zhukov et al. 1998).

Light Map: Example

A texture-mapped scene without light map:

Images courtesy, 3D Games by Watt and Policarpo
Light Map: Example

Light map before filtering (smoothing):

Images courtesy, 3D Games by Watt and Policarpo

Chapter B, Slide 17
Copyright © D. M. Mount and A. Varshney

Light Map: Example

Light map after linear filtering:

Note: Aliasing due to low light-map resolution. Could be avoided by using a higher resolution light map.

Images courtesy, 3D Games by Watt and Policarpo

Chapter B, Slide 18
Copyright © D. M. Mount and A. Varshney
In OpenGL, this can be done in two passes, one for colored textures and one for light map.

Images courtesy, 3D Games by Watt and Policarpo

Chapter 8, Slide 19
Copyright © D. M. Mount and A. Varshney
Shadow-Augmented Light Maps

**Adding Shadows to Light Maps:**
- If light sources and scene objects are static then the shadows will be static.
- Precompute the shadows as a part of the light map.

Discussion: Combine light-maps (for static parts of scene) with real-time shadows (for dynamic elements).

--

Overview

- Shadow painting
- Light Maps
  - Shadow Z-buffer
- Shadow Volumes
- Real-Time Reflection
The Shadow Z-Buffer

Z-Buffer Shadow Algorithm (Williams ’78):  
**Intuition:** Imagine that the viewer is positioned at the light source. The surface points that a viewer sees are the lit points. Thus, shadows and hidden-surface removal are related.

**Two-Step Process:**
- **Step 1:** Render the scene from the light’s point of view and store the results (just the depth information) in a shadow z-buffer (which is stored, e.g., as an environment cube).
- **Step 2:** Render the scene from the user’s viewpoint and for each pixel that overwrites a previously written pixel, determine its visibility to the light source by consulting the shadow z-buffer.

How is Step 2 implemented? (Light visibility test)
- Transform the pixel’s screen space coordinates (which holds depth information) into the light source’s coordinate frame.
- Index into the shadow z-buffer to see whether the rendered point’s depth is greater than the depth for the corresponding pixel in the shadow z-buffer.
- If the depth is greater \(\Rightarrow\) point is in shadow and use the shadow color, otherwise render normally.
The Shadow Z-Buffer

Low resolution shadow map
(Aliasing can be seen in some of the shadows.)

Higher resolution shadow map

Images from 3D Computer Graphics by Watt

The Shadow Z-Buffer

Shadow z-buffer from user's viewpoint (darker ⇒ farther from light source)

Shadow environment Z-buffer from light source's viewpoint (stored as an environment cube)

Images from 3D Computer Graphics by Watt
Overview

- Shadow painting
- Light Maps
- Shadow Z-buffer
- Shadow Volumes
- Real-Time Reflection

Shadow Volumes

**Shadow Volume:** (Crow 77, Heidmann 91)
- One of the most popular methods for real-time rendering of shadows.
- More accurate and more dynamic than light maps, but a careful implementation is required to get the very best results.
- Based on a clever use of the stencil buffer and depth buffer to identify areas of the scene that are visible to the light sources.
- Once identified, these areas are then rendered with full lighting.

**What is a shadow volume?**
- Given a point light source L and an object O, it is the region of 3-D space occluded from the light source by O.
- For each object of our scene, we want to know which portion lies outside the shadow volume.
**Shadow Volumes**

**Basic Shadow Volume Algorithm:**
- Render the scene as if it were completely in shadow (e.g., using only ambient light).
- For each light source:
  - Using the depth information for the scene, construct a mask in the stencil buffer that has holes only where the visible surface is not in shadow.
  - Render the scene again as if it were completely lit, using the stencil buffer to mask the shadowed areas. (Use additive blending to add colors to the scene.)

**Heidmann's Trick:**
- We render the faces of the shadow volumes to the stencil buffer. (They do not appear in the color or depth buffer.)
- Each pixel of the stencil buffer maintains a counter.
  - If we are moving from light into shadow, we increment the counter.
  - If we are moving from shadow into light, we decrement the counter.
  - If the final counter value is 0, then this pixel is in the light!
- We use back-face/front-face culling capability to determine whether we are coming from light to shadow or shadow to light.
Shadow Volume

**OpenGL Implementation:**
- Disable writes to the depth and color buffers.
- Enable **back-face culling**
  
  ```c
  glEnable(GL_CULL_FACE); glCullFace(GL_BACK);
  ```
- Set the stencil operation to **increment** on depth pass (only count shadows in front of the object).
  ```c
  glStencilOp(GL_INCR, GL_KEEP, GL_KEEP);
  ```
- Render the **shadow volumes** (because of culling, only their front faces are rendered).
- Enable **front-face culling**.
  ```c
  glCullFace(GL_FRONT);
  ```
- Set the stencil operation to **decrement** on depth pass.
  ```c
  glStencilOp(GL_DECR, GL_KEEP, GL_KEEP);
  ```
- Render the **shadow volumes** (only their back faces are rendered).
- Now that the stencil is set up, **draw your objects**.

---

**Shadow Volumes - Example**

1. Enable **back-face culling** and render the **shadow volumes** to stencil buffer, **increment** if depth test succeeds. (+)
2. Enable **front-face culling** and render the **shadow volumes** to stencil buffer, **decrement** if depth test succeeds. (-)
3. Pixels not rendered because **depth test fails** are shown as "x".
4. Note that + and - values are in **stencil buffer** (not on the object surfaces).
5. Areas where stencil buffer value is 0 are **illuminated**.
Real-time Reflection in OpenGL

**Reflection:**
- We consider the following question: How to generate the illusion of a reflective flat surface (e.g., a calm lake) in OpenGL?
- OpenGL does not support reflection, so we must "fake" this.

**Source:**
The following method is very accurate, and works even if part of the scene lies below the reflective surface.

It is based on a tutorial, which can be found at:

http://nehe.gamedev.net/data/lessons/lesson.asp?lesson=26
Reflection: Overview of the Method

**Basic Approach:**

- **Step 1:** Initializations (e.g., enable stenciling, clear stencil buffer).
- **Step 2:** Draw the **reflective region** into the **stencil buffer** (not to the graphics window). And activate stencil-buffer testing.
- **Step 3:** Reflect the scene **under the reflective surface** and draw it. (Since the stencil buffer is active, this reflection will appear only where the reflective surface appears in the image.)
- **Step 4:** Disable stenciling. Blend in the lake surface color.
- **Step 5:** Render the scene as usual (above the lake).

---

**Summary:**

- Real-Time Shadows
- Light-maps
- Shadow Z-buffer
- Shadow volumes
- Real-time reflection in OpenGL