Computer Graphics 9 - Environment mapping and mirroring

Tom Thorne

Slides courtesy of Taku Komura
www.inf.ed.ac.uk/teaching/courses/cg
Overview

• **Environment Mapping**
  - Introduction
  - Sphere mapping
  - Cube mapping
  - Refractive mapping

• **Mirroring**
  - Introduction
  - Reflection first
  - Stencil buffer
  - Reflection last
Environment Mapping: Background

- Many objects are glossy or transparent
- Glossy objects reflect the external world
- The world is refracted through transparent objects
- Important to make the scene appear realistic
Example
Environment Mapping: Background

Precisely simulating such phenomena is computationally costly

- Requires ray tracing, which can be expensive

- Tracking the rays, finding out where they collide, and doing another lighting computation
Environment Mapping

- Simple yet powerful method to generate reflections
- Simulate reflections by using the reflection vector to index a texture map at "infinity".

The original environment map was a sphere [by Jim Blinn ’76]
Sphere maps

- A mapping between the reflection vector and a circular texture
- Contains the whole environment around a point in a single image
- Low resolution around edges
Sphere maps: overview

- Compute the reflection vector at the surface of the object
- Find the corresponding texture coordinates on the sphere map
- Use the texture to colour the surface of the object
Indexing sphere maps

• Calculate the reflection vector $R$ based on direction to eye $I$

$$R = 2(N \cdot I)N - I$$
Indexing the sphere map

• Consider the mapping between reflection vectors on the sphere and the normal vector

• Assume that \( v \) is fixed at \((0,0,1)\)

• An un-normalised normal vector \( n \) is then:

\[
\mathbf{n} = \mathbf{r} + \mathbf{v} = (r_x, r_y, r_z + 1)
\]
Indexing the sphere map

\[ \overline{n} = \left( \frac{r_x}{m}, \frac{r_y}{m}, \frac{r_z + 1}{m} \right) \]

\[ m = \sqrt{r_x^2 + r_y^2 + (r_z + 1)^2} \]

• Assume the sphere is of unit radius and centred at the origin

• We can index the sphere map using the x and y components of the normalised normal vector
Generating sphere maps

- Take a photograph of a shiny sphere
- Mapping a cubic environment map onto a sphere
- For synthetic scenes, use ray tracing
Issues with sphere mapping

- Cannot change the viewpoint (requires recomputing the sphere map)
- Highly non-uniform sampling
- Highly non-linear mapping
- Linear interpolation of texture coordinates picks up the wrong texture pixels
- Do per-pixel sampling or use high resolution polygons
Cube Mapping

- The map resides on the surfaces of a cube around the object
- Align the faces of the cube with the coordinate axes
Procedure

**During rasterisation, for every pixel,**

1. Calculate the reflection vector \( R \) using the camera (incident) vector and the normal vector of the object \( N \)

2. Select the face of the environment map and the pixel on the face according to \( R \)

3. Colour the pixel with the colour of the environment map
   - Look up the environment map just using \( R \)
Procedure

**During rasterisation, for every pixel,**

1. Calculate the reflection vector $R$ using the camera (incident) vector and the normal vector of the object $N$

2. Select the face of the environment map and the pixel on the face according to $R$

3. Colour the pixel with the colour of the environment map
   - Look up the environment map just using $R$
Procedure

During rasterisation, for every pixel,

1. Calculate the reflection vector $R$ using the camera (incident) vector and the normal vector of the object $N$

2. Select the face of the environment map and the pixel on the face according to $R$

3. Colour the pixel with the colour of the environment map
   - Look up the environment map just using $R$
Indexing Cubic Maps

- Assume you have $R$ and the cube’s faces are aligned with the coordinate axes

- How do you decide which face to use?

- The reflection vector coordinate with the largest magnitude

- $R=(0.3, 0.2, 0.8) \rightarrow$ facing in $+z$ direction
Indexing Cubic Maps

• How do you decide which texture coordinates to use?

• Divide by the coordinate with the largest magnitude

• Now have a value in the range \([-1,1]\)

• Remapped to a value between 0 and 1.
Cubic Mapping: How to make one?

• Draw with a computer

• Take 6 photos of a real environment with a camera in the object’s position: much easier
Made from the Forum Images
Pros and cons

- Advantages of cube mapping?
- Problems with sphere mapping?
Refractive environment mapping

- When simulating effects mapping the refracted environment onto translucent materials such as ice or glass, we must use Refractive Environment Mapping.
Snell’s law

- Light travels at different speeds in different media

<table>
<thead>
<tr>
<th>Material</th>
<th>Index of Refraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vacuum</td>
<td>1.0</td>
</tr>
<tr>
<td>Air</td>
<td>1.0003</td>
</tr>
<tr>
<td>Water</td>
<td>1.3333</td>
</tr>
<tr>
<td>Glass</td>
<td>1.5</td>
</tr>
<tr>
<td>Plastic</td>
<td>1.5</td>
</tr>
<tr>
<td>Diamond</td>
<td>2.417</td>
</tr>
</tbody>
</table>

Snell’s Law: 
\[ n_1 \sin \theta_1 = n_2 \sin \theta_2 \]
Snell’s Law

- When light passes through a boundary between two materials of different density (air and water, for example), the light’s direction changes.

- The direction follows Snell’s Law

- We can do environment mapping using the refracted vector T

\[ \eta_1 \sin \theta_I = \eta_2 \sin \theta_T \]

<table>
<thead>
<tr>
<th>Material</th>
<th>Index of Refraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vacuum</td>
<td>1.0</td>
</tr>
<tr>
<td>Air</td>
<td>1.0003</td>
</tr>
<tr>
<td>Water</td>
<td>1.3333</td>
</tr>
<tr>
<td>Glass</td>
<td>1.5</td>
</tr>
<tr>
<td>Plastic</td>
<td>1.5</td>
</tr>
<tr>
<td>Diamond</td>
<td>2.417</td>
</tr>
</tbody>
</table>
Snell’s law

- Incoming vector $I$
- Refracted vector $T$

\[
T = rI + (w - k)n
\]

\[
r = \frac{n_1}{n_2}
\]

\[
w = -(I \cdot n)r
\]

\[
k = \sqrt{1 + (w - r)(w + r)}
\]
Refractive environment mapping

- Use the refraction vector after the first hit as the index to the environment map

- Costly to compute the second refraction vector
Summary

• Environment mapping is a quick way to simulate the effects of reflecting the surrounding world on the surface of a glossy object.

• Practical approaches are cube mapping and sphere mapping.

• Can also be applied for simulating refraction.
Overview

• Environment Mapping
  • Introduction
  • Sphere mapping
  • Cube mapping
  • Refractive mapping

• Mirroring
  • Introduction
  • Reflection first
  • Stencil buffer
  • Reflection last
Flat Mirrors: Background

• Basic idea: Drawing a scene with mirrors

• Mirrors reflect the world

• A scene with a mirror can be drawn by rendering the world twice:
  • Draw original scene
  • Draw reflected scene
Flat Mirrors: Background

• Simply rendering the scene twice can result in problems

• Unless the mirrored world is hidden by the real world, the flipped world may appear outside of the mirror!

• We can avoid such problems using a “stencil buffer”
Reflecting objects

- If the mirror passes through the origin, and is aligned with a coordinate axis, then just negate appropriate coordinate.

- For example, if a reflection plane has a normal $n=(0,1,0)$ and passes the origin, the reflected vertices can be obtained by scaling matrix $S(1,-1,1)$.
Reflecting objects

- What if the mirror is not on a plane that passes the origin?

- How do we compute the mirrored world?

- First, we need to compute the location of objects relative to the mirror
Recap:
Transformations between different coordinate systems

- We can interpret that the transformation matrix is converting the location of vertices between different coordinate systems
  
  - $v_g = M v_l$
  - $v_l = M^{-1} v_g$
Reflecting objects

- To know the positions of objects with respect to the mirror coordinate
- We multiply by a transformation matrix from the world to the mirror coordinates

\[ x' = R(n)^{-1}T(-p)x \]
Reflecting objects

- For finding out the flipped location in the mirror coordinate, we multiply by the mirroring matrix

\[ x'' = S(1,1,-1) \ x' \]
Reflecting objects

- Now we want to know where the flipped points are with respect to the world origin.

- We can multiply $x''$ by the transformation matrix to move from the origin to the mirror to know where it is with respect to $O$.

$$x''' = T(p)R(n) \cdot x''$$
Reflecting objects

- Combined:

\[ x' = R(n)^{-1}T(-p) \ x \]
\[ x'' = S(1,1,-1) \ x' \]
\[ x''' = T(p)R(n) \ x'' \]
\[ x'''' = T(p)R(n)S(1,1,-1)R(n)^{-1}T(-p) \ x \]

\[ M_{\text{mirror}} = T(p)R(n)S(1,1,-1)R(n)^{-1}T(-p) \]
Reflecting objects

- Need to avoid drawing objects behind the mirror in front of it
- Specify a clipping plane parallel to the mirror
Drawing the mirrored world

- Draw the mirrored world first, then the real world
  - Only using the depth (Z) buffer
  - Does not work in some cases
- Draw the real-world first, and then the mirrored world
  - Requires using a stencil buffer
Z-buffer

- One method of hidden surface removal

- Basic Z-buffer idea: For every input polygon
  - For every pixel in the polygon interior, calculate its corresponding z value.
  - Compare the depth value with the closest value from a different polygon (largest z) so far
  - Paint the pixel (filling in the color buffer) with the color of the polygon if it is closer
Z buffer example

Correct Final image

z = -0.8
Z = -0.5
Z = -0.3

eye

Top View

z
**Z buffer example**

Step 1: Initialize the depth buffer

<table>
<thead>
<tr>
<th>-1.0</th>
<th>-1.0</th>
<th>-1.0</th>
<th>-1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1.0</td>
<td>-1.0</td>
<td>-1.0</td>
<td>-1.0</td>
</tr>
<tr>
<td>-1.0</td>
<td>-1.0</td>
<td>-1.0</td>
<td>-1.0</td>
</tr>
<tr>
<td>-1.0</td>
<td>-1.0</td>
<td>-1.0</td>
<td>-1.0</td>
</tr>
<tr>
<td>-1.0</td>
<td>-1.0</td>
<td>-1.0</td>
<td>-1.0</td>
</tr>
</tbody>
</table>
Z buffer example

Step 2: Draw the blue polygon (assuming the program draws blue polygon first – the order does not affect the final result any way).
Z buffer example

Step 3: Draw the yellow polygon

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-1.0</td>
<td>-1.0</td>
<td>-1.0</td>
<td>-1.0</td>
</tr>
<tr>
<td>-1.0</td>
<td>-0.3</td>
<td>-0.3</td>
<td>-1.0</td>
</tr>
<tr>
<td>-0.5</td>
<td>-0.3</td>
<td>-0.3</td>
<td>-1.0</td>
</tr>
<tr>
<td>-0.5</td>
<td>-0.5</td>
<td>-1.0</td>
<td>-1.0</td>
</tr>
</tbody>
</table>

If the depth value is larger than that in the z-buffer, the pixel is coloured and value in the z-buffer is updated.
Step 4: Draw the red polygon

If the depth value is larger than that in the z-buffer, the pixel is coloured and value in the z-buffer is updated.
Rendering Reflected Scene First

- First pass: Render the reflected scene without mirror, depth test on

- Second pass:
  - Disable the colour buffer, and render the mirror polygon (setting the Z-buffer values but not drawing pixel colours over reflected scene)
  - Now the Z buffer of the mirror region is set to the mirror’s surface

- Third Pass:
  - Enable the colour buffer again
  - Render the original scene, without the mirror
  - Depth buffer stops us from writing over things in mirror
Rendering the reflected scene first

- The reflected area outside the mirror region is overwritten by the objects in the front
- Can’t draw multiple mirrors or reflections of mirrors in mirrors (recursive reflections)
Using a stencil buffer

- The stencil buffer can help to prevent drawing outside of the mirror region
Using a stencil buffer

- The stencil buffer acts like a paint stencil - it lets some fragments through but not others
- It stores multi-bit values
- You specify two things:
  - The test that controls which fragments get through
  - The operations to perform on the buffer when the test passes or fails
Example

Mirror
Procedure

- First pass:
  - Render the scene without the mirror

- For each mirror:
  - Second pass:
    - Clear the stencil, disable the write to the colour buffer, render the mirror, setting the stencil to 1 if the depth test passes
  - Third pass:
    - Clear the depth buffer with the stencil active, passing things inside the mirror only
    - Reflect the world and draw using the stencil test. Only things seen in the mirror will be drawn
    - Combine it with the scene made during the first pass
Multiple mirrors

• Can manage multiple mirrors

• Render normal view, then do other passes for each mirror

• A recursive formulation exists for mirrors that see other mirrors

  • After rendering the reflected area inside the mirror surface, render the mirrors inside the mirror surface, and so on
References

• Akenine-Möller, Chapter 8.4 (Environment mapping)

• Akenine-Möller, Chapter 9.3.1 (Planar reflections)

• http://threejs.org/examples/#webgl_materials_cubemap

• http://www.pauldebevec.com/ReflectionMapping/