Lecture 7. Visual realism
Part 2: Texture mapping programming

Yong-Jin Liu
liuyongjin@tsinghua.edu.cn
Texture mapping

- Easy to understand; but difficult to master
- In the 70s: complex algorithm based on texture mapping
- Completely programmable hardware
\( f(x) = p \)
Developable surface:

- Concept in differential geometry
- Measure on surface: distance between two points
- If a surface can be developed into a plane without distortion, then the surface is developable surface

Cylinder, cone, plane,

Tangent plane of a space curve
Developable surface

- Without distortion means: without tension/compression and shearing, but can undergo bending.
Most surface is undevelopable

- Folding into a plane will lead to distortion somewhere inevitably
Most surface is undevelopable

Different optimization criteria

- Length distortion minimization
- Area distortion minimization
- Angle distortion minimization
- Etc.

Cartography

Draftsmanship
子午线投影（Central Meridian）

Near equator is better; near polar is worse
Albers Projection

Two standard parallels define the map layout.
( selected by mapmaker )

Areas equal to globe.
Deformation of shapes increases away from those parallels.

Area preserving is good; but shape distortion is large
Central meridian selected by mapmaker touches cylinder if the cylinder is tangent.

Can show whole Earth, but the directions, distances, and areas are reasonable accurate only within 15 degrees of the central meridian.

No stright rhumb lines.
space-oblique Mercator projection
Directions from center are good; but shape and
Length changes largely
球面投影（立体投影）
（Stereographic Projection）

Angle preserving; but not area and length
Steps of texture mapping in OpenGL

1. Create texture object, and specify a texture for it.
2. Define how the texture maps to each pixel.
3. Enable texture mapping function.
4. Draw the 3D scene, providing texture and geometry object coordinates.
Detailed steps:

- Get unused texture object handle using `glGenTextures();`
- Set the state parameters of texture object;
- Specify texture picture using `glTexImage2D()` or `glutBuild2DMipmaps();`
- Binding texture object using `glBindTexture();`
- Enable texture mapping;
- Render geometry object: send it to OpenGL, specify texture coordinate for each vertex.
Steps of texture mapping:

1. Get unused texture object handle using glGenTextures()

```c
GLuint texID;
glGenTextures(1, &texID);
```

```c
void glGenTextures(GLsizei n, GLuint* textures);
void glDeleteTextures(GLsizei n, const GLuint* textures);
GLboolean glIsTexture(GLuint texture);
```
Steps of texture mapping:

2. Set the state parameters of texture object

```c
void glTexParameter[if](GLenum target,
                        GLenum pname,
                        TYPE param);
```

target: either `GL_TEXTURE_1D` or `GL_TEXTURE_2D`
<table>
<thead>
<tr>
<th>pname</th>
<th>param</th>
</tr>
</thead>
<tbody>
<tr>
<td>GL_DEPTH_TEXTURE_MODE</td>
<td>Format depth comparison result to gray scale, intensity or alpha texel: GL_LUMINANCE, GL_INTENSITY, GL_ALPHA</td>
</tr>
<tr>
<td>GL_GENERATE_MIPMAP</td>
<td>GL_TRUE, GL_FALSE, enable or disable the generating of mipmap</td>
</tr>
<tr>
<td>GL_TEXTURE_COMPARE_FUNC</td>
<td>GL_LEQUAL, GL_GEQUAL, control depth comparison</td>
</tr>
<tr>
<td>pname</td>
<td>param</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>GL_TEXTURE_COMPARE_MODE</td>
<td>Enable or disable depth comparison: GL_COMPARE_R_TO_TEXTURE, GL_NONE</td>
</tr>
<tr>
<td>GL_TEXTURE_MAG_FILTER</td>
<td>Set the texture magnification filter method: GL_NEAREST, GL_LINEAR</td>
</tr>
<tr>
<td>GL_TEXTURE_MIN_FILTER</td>
<td>Set the texture minimization filter method: GL_NEAREST, GL_LINEAR,</td>
</tr>
<tr>
<td></td>
<td>GL_NEAREST_MIPMAP_NEAREST, ...</td>
</tr>
<tr>
<td><strong>pname</strong></td>
<td><strong>param</strong></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------------------------------------------------------------</td>
</tr>
<tr>
<td>GL_TEXTURE_WRAP_S</td>
<td>Set the surrounding mode of texture coordinates beyond 0.0~1.0: GL_CLAMP, GL_REPEAT, GL_CLAMP_TO_EDGE</td>
</tr>
<tr>
<td>GL_TEXTURE_WRAP_T</td>
<td></td>
</tr>
<tr>
<td>GL_TEXTURE_WRAP_R</td>
<td></td>
</tr>
</tbody>
</table>
GL_REPEAT
GL_CLAMP_TO_EDGE
Steps of texture mapping:

3. Specify texture picture using `glTexImage2D()`

```c
void glTexImage2D(GLenum target, GLint level,
                 GLint internalformat, GLsizei width,
                 GLsizei height, GLint border,
                 GLenum format, GLenum type,
                 const GLvoid* data);

glTexImage2D(GL_TEXTURE_2D, 0, 3, 640, 480, 0,
             GL_RGB, GL_UNSIGNED_BYTE, my_Image);
```
- How to read picture?
- Read the pixel data into a weight $\times$ height array named data.
- data points to the left bottom of the pixel matrix, and process the pixel by the order of left to right and bottom to up, first row then column.
Steps of texture mapping:

4. Binding texture object using `glBindTexture()`

```c
void glBindTexture(GLenum target, GLuint texture);
```

```c
GLuint my_texture_ID;

glBindTexture(GL_TEXTURE_2D, my_texture_ID);

GLuint my_texture_ID;

glGenTextures(1, &my_texture_ID);
```
3D model represented by mesh
15_texture_mapping
Example analysis

- Geometry
  - Cylinder
  - Torus
  - Sphere
- Texture
radius = 1.0;  length = 2.0;  slices = 32

Slices = \frac{2\pi}{\theta}
Z=0 plane

minRadius

majRadius

X plane

Y plane

minApprox

maxApprox

z plane

X plane

Y plane
Central Meridian (selected by mapmaker)

- Great distortion at high latitudes
- Examples of two rhumb lines (direction true between any two points)
- Equator touches cylinder if cylinder is tangent
- Reasonably true shapes and distances within 15 degrees of Equator
25_projected_shadows
Projective geometry
Texture mapping
its called experimenting..., at 75.0fps
Initial coordinate \((x, y)\)

\[ z = x + iy \]

Coordinate after transformation \((u, v)\)

\[ w = u + iv \]

Conformal mapping: Angle preserving transformation

\[ w = k(1 + z + e^z) \]
Texture + geometry transformation

MyRipple
Fisheye transformation

\[ \theta' = \theta; \]

\[ r' = \begin{cases} 
  m \times \frac{r}{1 - r} & \text{if } r < 1 \\
  1 + m \times \frac{r}{1 - r} & \text{if } r \geq 1
\end{cases} \]
It's called experimenting..., at 70.1 fps
it's called experimenting..., at 75.5 fps
its called experimenting..., at 75.0fps
void Terrain::Render()
{
    double plane_eqn[4] = { 0, -1, 0, origin.y + dimen.y*waterHeight};
    glClipPlane( GL_CLIP_PLANE0, plane_eqn );
    glEnable( GL_CLIP_PLANE0 );

    // vertical-flip everything
    glTranslatef(0, 2*waterHeight*dimen.y, 0);
    glScalef(1, -1, 1);

    // reverse culling order
    glCullFace( GL_BACK );

    RenderTerrain();
    RenderSkyBox();

    // restore correct culling
    glCullFace( GL_FRONT );

    // re-flip vertically
    glScalef(1, -1, 1);
    glTranslatef(0, -2*waterHeight*dimen.y, 0);

    // disable extra clipping-plane
    glDisable( GL_CLIP_PLANE0 );

    // render water
    RenderWater();

    // render the terrain
    RenderTerrain();

    // render the skybox (will set its own textures)
    RenderSkyBox();
}
Selective Course Project

- Try to program the following texture mapping model: