Project 3. Paint A Teapot with Different Textures

Due 11:59:59pm, December 2nd

You can work in a team with no more than two members.

If a team consists of a graduate student and an undergraduate student, graduate-students' requirements are applied in the grading.

Please make sure to turn in a 'readme' file to include the team members' names.

Topics

Texture mapping in OpenGL

Environment mapping

Using Texture Objects

1. specify textures in texture objects
2. set texture filter
3. set texture function
4. set texture wrap mode
5. set optional perspective correction hint
6. bind texture object
7. enable texturing (**`glEnable(GL_TEXTURE_2D)`**)

Texture Parameters

OpenGL has a variety of parameters that determine how texture is applied

- Wrapping parameters determine what happens if s and t are outside the (0,1) range
- Filter modes allow us to use area averaging instead of point samples
- Mipmapping allows us to use textures at multiple resolutions
- Environment parameters determine how texture mapping interacts with shading

Setting Texture Parameters

```
glTexParameteri(GLenum target, GLenum pname,Type param );
```

Can be f, fv, iv, etc.

- `target:` type of texture, e.g. `GL_TEXTURE_2D`
- `pname:` the symbolic name of a single-valued texture parameter, e.g., `GL_TEXTURE_WRAP_S` and `GL_TEXTURE_MIN_FILTER`
  - Need to run this function for every parameter
- `param:` the value of parameter

OpenGL manual for glTexParameter

https://www.opengl.org/sdk/docs/man4/html/glTexParameter.xhtml

Wrapping Mode

Valid range of texture coordinates: $0 \leq s, t \leq 1$

***Clamping mode*: if $s, t > 1$ use 1, if $s, t < 0$ use 0**

- Avoid wrapping artifacts for nonrepeated texture patterns

***Wrapping mode*: use s, t *modulo* 1**

- Preferred for repeated texture patterns, e.g., checkerboard

```
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_S,GL_CLAMP)

glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_T,GL_REPEAT)
```

`GL_TEXTURE_WRAP_S(T)`: set texture coordinate *s (t)*

Parameters to choose: `GL_CLAMP`, `GL_CLAMP_TO_EDGE`, `GL_CLAMP_TO_BORDER`, `GL_MIRRORED_REPEAT`, `GL_REPEAT`, or `GL_MIRROR_CLAMP_TO_EDGE`.

Wrapping Mode

***Clamping mode*: if $s, t > 1$ use 1, if $s, t < 0$ use 0**

***Wrapping mode*: use s, t *modulo* 1**

Example: repeated texture patterns

Wrapping Mode

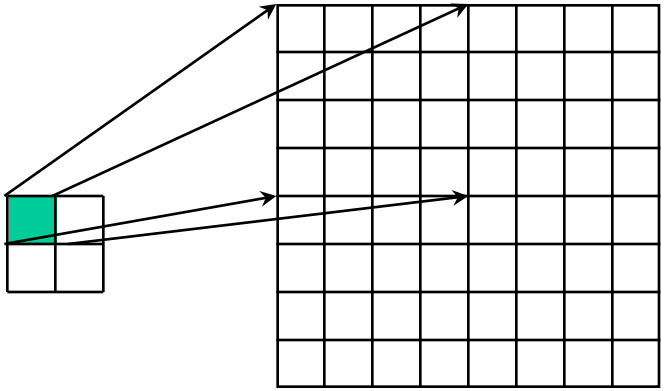
Example: nonrepeated texture

https://open.gl/textures

Magnification and Minification

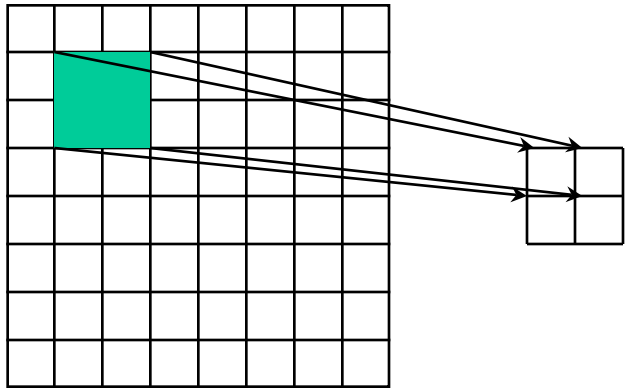
***Minification**:* more than one texel can cover a pixel
***Magnification**:* more than one pixel can cover a texel

Can use point sampling (nearest texel) or linear filtering (2 x 2 filter) to obtain texture values



Texture Polygon

Magnification



Texture Polygon

Minification

Magnification and Minification: Filter Modes

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXURE_MAG_FILTER,
                GL_NEAREST);

glTexParameteri(GL_TEXTURE_2D, GL_TEXURE_MIN_FILTER,
                GL_LINEAR);
```

GL_NEAREST is faster, but causes jitter edge
GL_LINEAR is the default mode.

Note that linear filtering requires a border of an extra texel for filtering at edges (border = 1)

Example: Texture Object

```
GLuint textures[1];
glGenTextures( 1, textures );

glBindTexture( GL_TEXTURE_2D, textures[0] );
glTexImage2D( GL_TEXTURE_2D, 0, GL_RGB, TextureSize,
   TextureSize, 0, GL_RGB, GL_UNSIGNED_BYTE, image );
glTexParameterf( GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,
    GL_REPEAT );
glTexParameterf( GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,
     GL_REPEAT );
glTexParameterf( GL_TEXTURE_2D,
      GL_TEXTURE_MAG_FILTER, GL_NEAREST );
glTexParameterf( GL_TEXTURE_2D,
      GL_TEXTURE_MIN_FILTER, GL_NEAREST );
glActiveTexture( GL_TEXTURE0 );
```

Mipmapped Textures

An alternative way to deal with minification

Mipmapping sets up multiple levels of prefiltered texture maps of decreasing resolutions, for example,

- Level 0: 64x64,
- Level 1: 32x32,
- Level 2: 16x16,
- Level 3: 8x8,
- Level 4: 4x4,
- Level 5: 2x2,
- Level 6: 1x1

Mipmapped Textures

Choose appropriate size automatically by OpenGL

`glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_NEAREST_MIPMAP_NEAREST);`

Or `GL_LINEAR_MIPMAP_NEAREST`

Or declare mipmap level during texture definition, e,g,

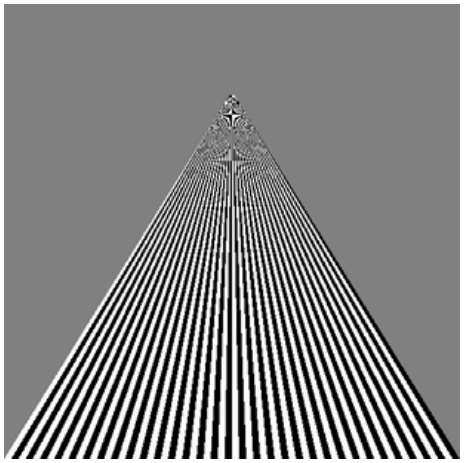
`glTexImage2D( GL_TEXTURE_2D, level, … )`

Example

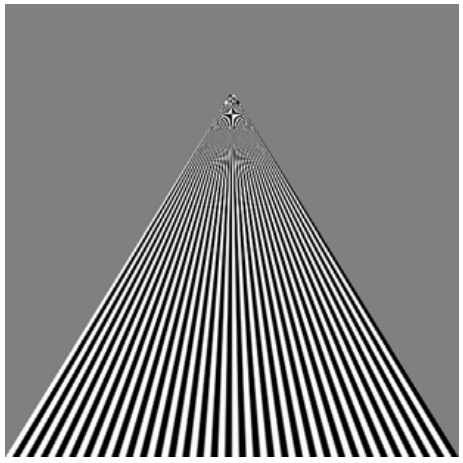
Original texture pattern: black and white stripes
Object: quadrilateral

Significant aliasing effects

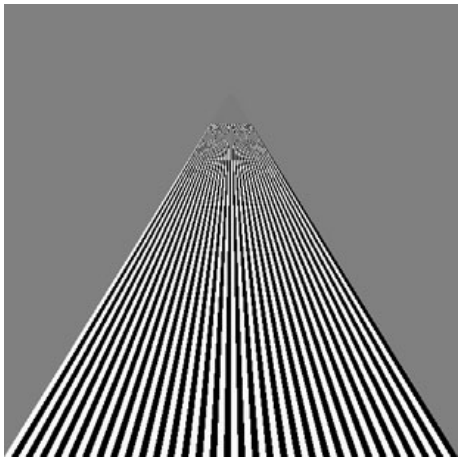
point sampling (nearest)



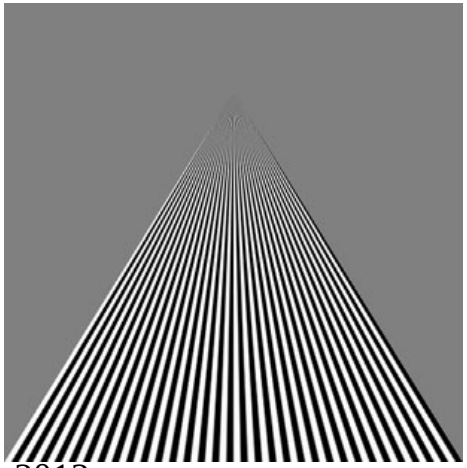
linear filtering



mipmapped Point sampling



mipmapped Linear filtering

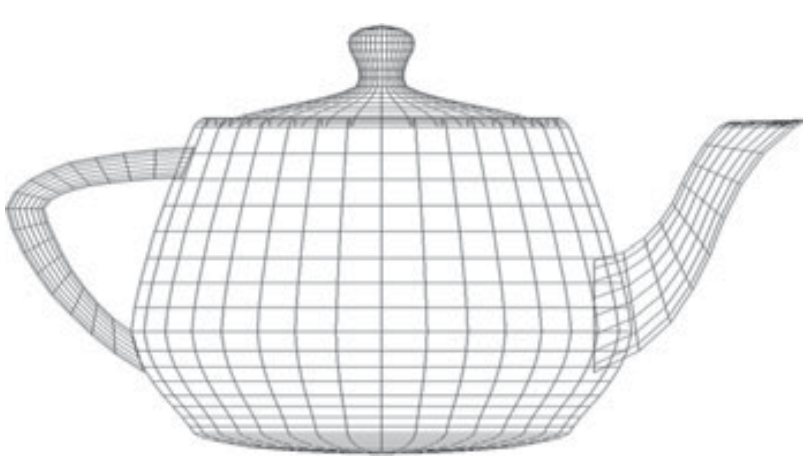


Assign Texture Coordinates

Easy for equal-sized rectangular polygons

Challenging for curved objects

- Polygons vary in size
  - –Larger polygons for flatter region
  - –Smaller polygons for higher curvature
- Resulting varying size of texture patterns



Assign Texture Coordinates

Represent texture coordinates as a linear combination of the vertex coordinates

$$s = a_s x + b_s y + c_s z + d_s w$$

$$t = a_t x + b_t y + c_t z + d_t w$$

Generate texture coordinates in terms of the distance from a plane in either eye or object frame

Object frame

Eye frame

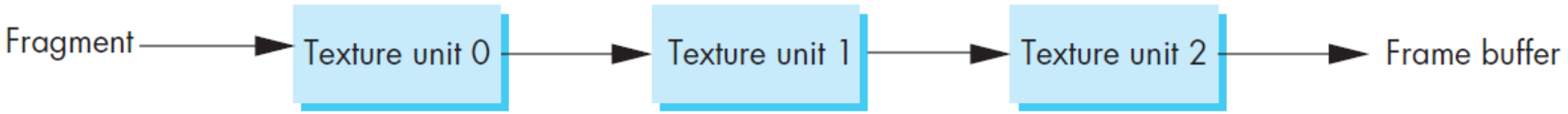
Multitexturing

Apply a sequence of textures through cascaded texture units.

Activate each texture object in turn and determine how to apply multiple textures.

```
glActiveTexture( GL_TEXTURE0 );
glBindTexture( GL_TEXTURE_2D, textures[0] );
glActiveTexture( GL_TEXTURE1 );
glBindTexture( GL_TEXTURE_2D, textures[1] );
```

Each texture unit can have its own texture coordinates.



Apply Texture in Fragment Shader

Control how textures are applied in fragment shader

```
vec4 tex0 = texture2D(Tex0, TexCoord0);

vec4 tex1 = texture2D(Tex1, TexCoord1);
```

Modulate: `gl_FragColor = tex0 * tex1;`

Blend with a factor or based on the alpha:

```
gl_FragColor = mix(tex0, tex1, BlendFactor);
```

Can be alpha0

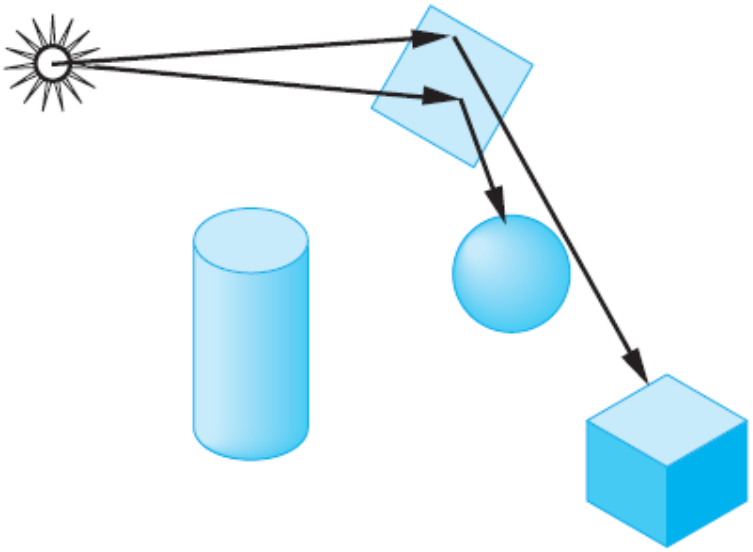
Add: `gl_FragColor = tex0+tex1;`

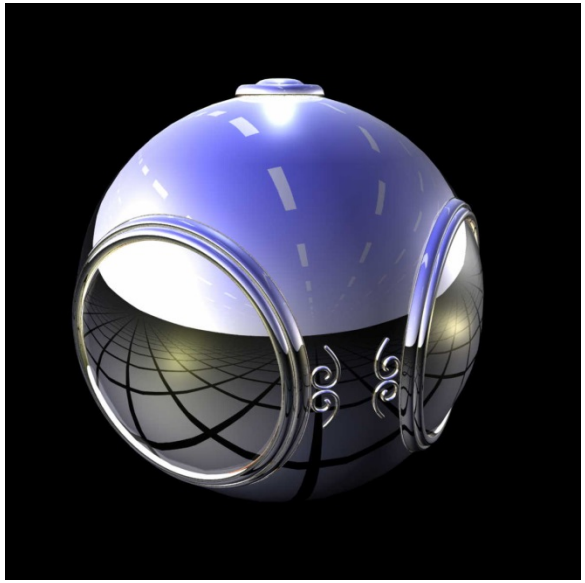
Mapping Methods

- Texture mapping
- Environmental (reflection) mapping
  - Variant of texture mapping
- Bump mapping
  - Solves flatness problem of texture mapping

Environment/Reflection Map

For reflective surfaces, model the specular reflections which mirror the environment



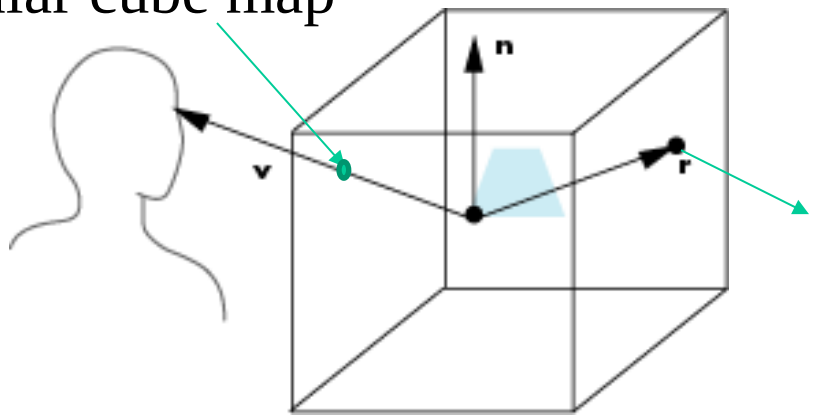


Environment/Reflection Map

Texture coordinates of a cube map are 3D

Instead of using the view vector to determine the texture, ***environment map*** uses reflection vector to locate texture in ***cube map***

corresponding texel
for regular cube map



corresponding texel
for reflection map

Cube Maps

We can form a cube map texture by defining six 2D texture maps that are square and correspond to the sides of a box

Supported by OpenGL, e.g.,

```
glEnable(GL_TEXTURE_CUBE_MAP);

glBindTexture(GL_TEXTURE_CUBE_MAP, texObj[0]);

glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X ,0,3,1,1,0,GL_RGB,
GL_UNSIGNED_BYTE, image1);
```

The other 5 are GL_TEXTURE_CUBE_MAP_NEGATIVE_X,

GL_TEXTURE_CUBE_MAP_POSITIVE_Y, GL_TEXTURE_CUBE_MAP_NEGATIVE_Y, GL_TEXTURE_CUBE_MAP_POSITIVE_Z, GL_TEXTURE_CUBE_MAP_NEGATIVE_Z

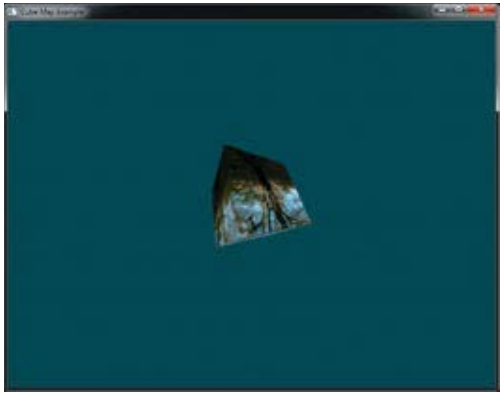
Cube Maps

Supported in GLSL through cubemap sampler

```
vec4 texColor = textureCube(mycube, texcoord);
```

Note: texture coordinates must be 3D

Outside



Closeup

From the center

Skybox example

Shreiner et al: OpenGL Programming Guide (Version 4.3)

Angel's Example of Reflection Map

// Define the texture object as a cube map

```
GLuint tex[1];

GLubyte red[3] = {255, 0, 0}; GLubyte green[3] = {0, 255, 0};

GLubyte blue[3] = {0, 0, 255}; GLubyte cyan[3] = {0, 255, 255};

GLubyte magenta[3] = {255, 0, 255}; GLubyte yellow[3] = {255, 255, 0};

glActiveTexture(GL_TEXTURE1);

glGenTextures(1, tex);

glBindTexture(GL_TEXTURE_CUBE_MAP, tex);
```

Angel's Example of Reflection Map

//Set up the cube map, assuming the environment has been mapped to the cube

```
glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X,0,GL_RGB,1,1,0,GL_RGB,

GL_UNSIGNED_BYTE, red);

glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_X,0,GL_RGB,1,1,0,GL_RGB,
GL_UNSIGNED_BYTE, green);

glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_Y,0,GL_RGB,1,1,0,GL_RGB,

GL_UNSIGNED_BYTE, blue);

glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_Y,0,GL_RGB,1,1,0,GL_RGB,
GL_UNSIGNED_BYTE, cyan);

glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_Z,0,GL_RGB,1,1,0,GL_RGB,

GL_UNSIGNED_BYTE, magenta);

glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_Z,0,GL_RGB,1,1,0,GL_RGB,
GL_UNSIGNED_BYTE, yellow);

glTexParameteri(GL_TEXTURE_CUBE_MAP,GL_TEXTURE_MIN_FILTER,GL_NEAREST);
```

Angel's Example of Reflection Map

```
// set up a sampler for fragment shader
GLuint texMapLocation;
texMapLocation = glGetUniformLocation(program, "texMap");
glUniform1i(texMapLocation, 1); // corresponding to unit 1
```

Angel's Example of Reflection Map

```
//Calculate the normal of each side of the cube
point4 normals[N];
vec4 normal;
void quad(int a, int b, int c, int d){
  static int i =0;
  normal = normalize(cross(vertices[b]-vertices[a],vertices[c]-vertices[b]));
  normals[i] = normal;
  points[i] = vertices[a];
  i++;
…
}
```

Angel's Example of Reflection Map

```
// send the normal to the vertex array
glBindBuffer(GL_ARRAY_BUFFER, buffer);
glBufferData(GL_ARRAY_BUFFER, sizeof(points) + sizeof(normals),
NULL, GL_STATIC_DRAW);
glBufferSubData(GL_ARRAY_BUFFER, 0, sizeof(points), points);
glBufferSubData(GL_ARRAY_BUFFER, sizeof(points),sizeof(normals), normals);

// link to the shader
loc2 = glGetAttribLocation(program, "Normal");
glEnableVertexAttribArray(loc2);
```

Angel's Example of Reflection Map: Vertex Shader

```
uniform mat4 Projection, ModelView;
in vec4 vPosition;
in vec4 normal;
out vec3 R; //reflection vector

void main(void)
{
   gl_Position = Projection*ModelView*vPosition;
   vec3 N = normalize(ModelView*normal);
   vec4 eyePos = ModelView*vPosition;
   R = reflect(-eyePos.xyz, N);
}
```

Angel's Example of Reflection Map: Fragment Shader

```
in vec3 R;
uniform samplerCube texMap;

void main(void)
{
    gl_FragColor = texture(texMap, R);
}
```

Seamless Cube Map

Edges at adjacent faces may cause problem

`glEnable(GL_TEXTURE_CUBE_MAP_SEAMLESS)`

Shreiner et al: OpenGL Programming Guide (Version 4.3)