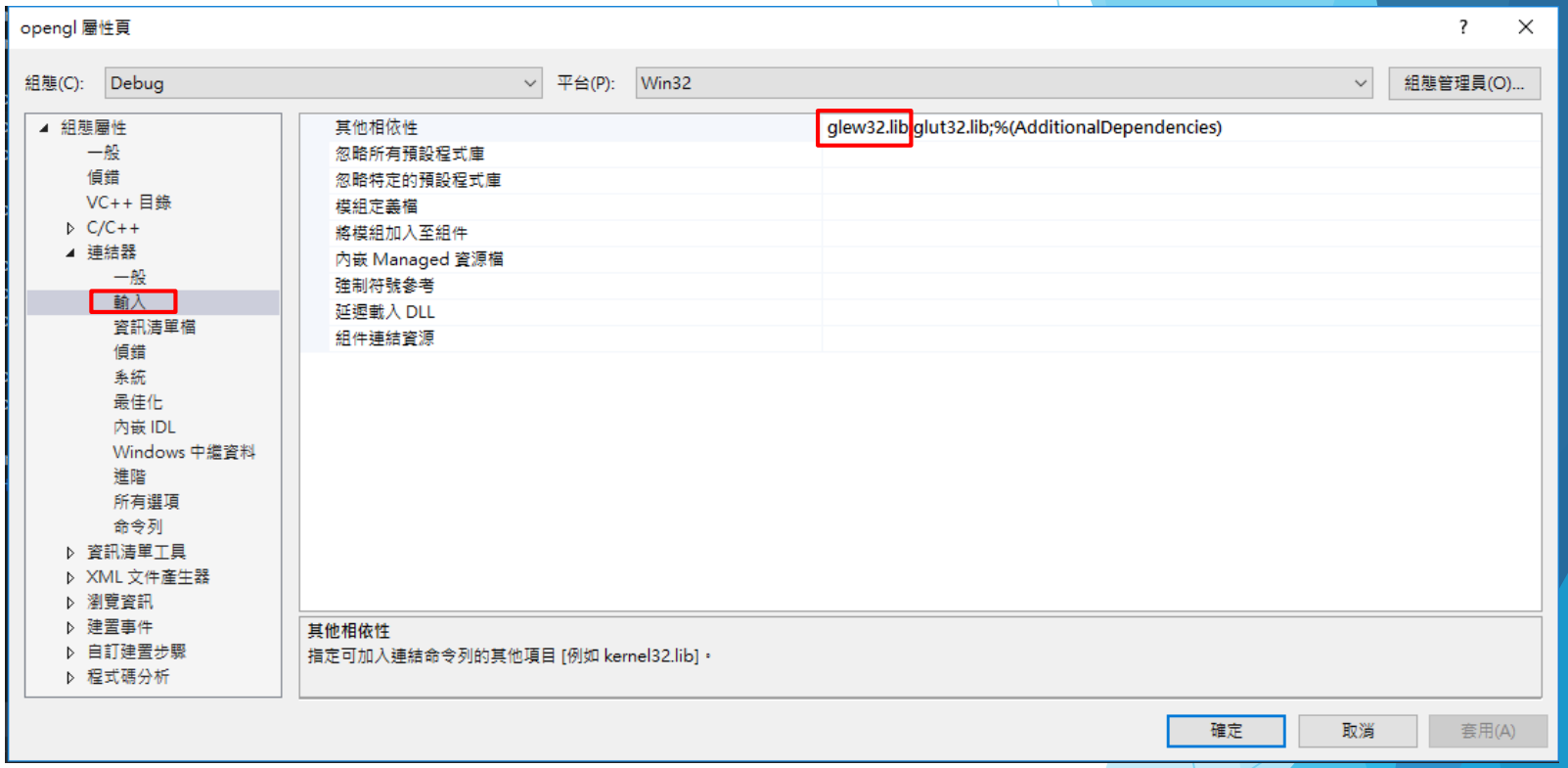
OpenGL - Texture

GLEW

- Unzip glew.zip
- Put “glew.h” in folder “include”
- Put “glew32.lib” in folder “lib”
- Put “glew32.dll” in folder “dll”

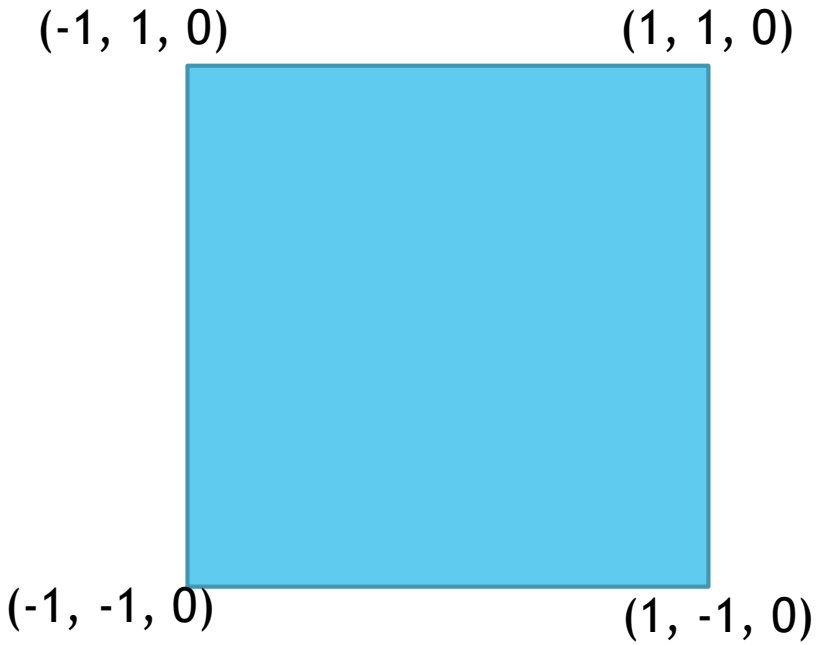


- include “glew.h” before include glut.h
- Call glewInit() after glutInit()

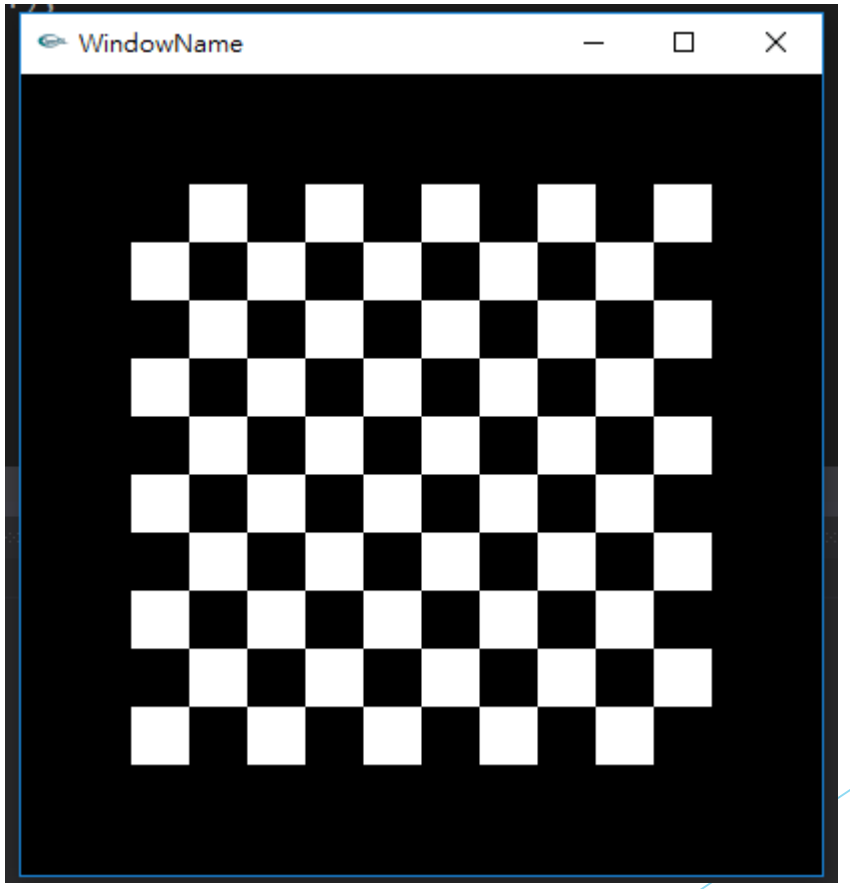
- Put textures in folder ”dll”

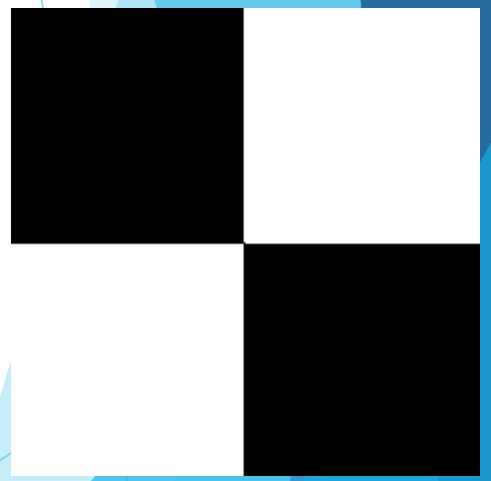
Example

- Scale(3, 3, 3)



(0, 0, 10)





Original texture

Example

- Read texture
- Generate texture
- Copy texture into GPU memory

```
132  unsigned char* readBMP(char* filename)
133  {
134      FILE* f = fopen(filename, "rb");
135      unsigned char info[54];
136      fread(info, sizeof(unsigned char), 54, f); // read the 54-byte header
137
138      // extract image height and width from header
139      image_width = *(int*)&info[18];
140      image_height = *(int*)&info[22];
141
142      int size = 3 * image_width * image_height;
143      unsigned char* data = new unsigned char[size];
144      fread(data, sizeof(unsigned char), size, f);
145      fclose(f);
146
147      for (int i = 0; i < size; i += 3){
148          unsigned char tmp = data[i];
149          data[i] = data[i + 2];
150          data[i + 2] = tmp;
151      }
152      return data;
153  }
```

```
157  void init_texture()
158  {
159      //enable 2D texture
160      glEnable(GL_TEXTURE_2D);
161
162      unsigned char *data = readBMP("./check_old.bmp");
163      glGenTextures(1, &texture_id);
164      glBindTexture(GL_TEXTURE_2D, texture_id);
165
166
167      //without mipmap
168      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
169      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
170      glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_REPLACE);
171
172      glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, image_width, image_height, 0, GL_RGB, GL_UNSIGNED_BYTE, data);
173      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
174      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
175
176      /* ... */
189
190      glBindTexture(GL_TEXTURE_2D, 0);
191      delete data;
192  }
193
```

Example

```
45  void display()
46  {
47      //ModelView Matrix
48      glMatrixMode(GL_MODELVIEW);
49      glLoadIdentity();
50      gluLookAt(0.0f, 0.0f, 10.0f, 0.0f, 0.0f, 0.0f, 0.0f, 1.0f, 0.0f);
51      //Projection Matrix
52      glMatrixMode(GL_PROJECTION);
53      glLoadIdentity();
54      gluPerspective(45, width / (GLfloat)height, 0.1, 1000);
55      //Viewport Matrix
56      glViewport(0, 0, width, height);
57
58      //clear
59      glClear(GL_COLOR_BUFFER_BIT);
60      glClear(GL_DEPTH_BUFFER_BIT);
61
62      glMatrixMode(GL_MODELVIEW);
63      lighting();
64
65      glBindTexture(GL_TEXTURE_2D, texture_id);
66      glPushMatrix();
67      glScalef(3, 3, 3);
68      glBegin(GL_QUADS);
69      glNormal3f(0, 0, 1);
70      glTexCoord2f(5.0f, 5.0f);
71      glVertex3f(1.0f, 1.0f, 0.0f);
72      glNormal3f(0, 0, 1);
73      glTexCoord2f(0.0f, 5.0f);
74      glVertex3f(-1.0f, 1.0f, 0.0f);
75      glNormal3f(0, 0, 1);
76      glTexCoord2f(0.0f, 0.0f);
77      glVertex3f(-1.0f, -1.0f, 0.0f);
78      glNormal3f(0, 0, 1);
79      glTexCoord2f(5.0f, 0.0f);
80      glVertex3f(1.0f, -1.0f, 0.0f);
81      glEnd();
82      glPopMatrix();
83      glBindTexture(GL_TEXTURE_2D, 0);
84
85      glutSwapBuffers();
```

Texture – enable texture

- glEnable(GLenum cap);
  - GL_TEXTURE_1D, GL_TEXTURE_2D, GL_TEXTURE_3D, GL_TEXTURE_CUBE_MAP
- glDisable(GLenum cap);

Texture – generate texture name

- void glGenTextures(GLsizei n, GLuint* textures);
  - Generate texture name(id)
  - n: Specifies the number of texture names to be generated.
  - textures: Specifies an array in which the generated texture names are stored.
- GLboolean glIsTextures(GLuint texture);
  - Determine if a name corresponds to a texture
  - textures: the name(id) of a texture
- void glDeleteTextures(GLsizei n, GLuint* textures)
  - Deletes n textures named by the elements of the array textures
  - n, textures: the same as glGenTextures

Texture – read texture into memory

- For example, read a BMP file

```
132  unsigned char* readBMP(char* filename)
133  {
134      FILE* f = fopen(filename, "rb");
135      unsigned char info[54];
136      fread(info, sizeof(unsigned char), 54, f); // read the 54-byte header
137
138      // extract image height and width from header
139      image_width = *(int*)&info[18];
140      image_height = *(int*)&info[22];
141
142      int size = 3 * image_width * image_height;
143      unsigned char* data = new unsigned char[size];
144      fread(data, sizeof(unsigned char), size, f);
145      fclose(f);
146
147      for (int i = 0; i < size; i += 3){
148          unsigned char tmp = data[i];
149          data[i] = data[i + 2];
150          data[i + 2] = tmp;
151      }
152      return data;
153  }
```

Read image width, height

Read image

Texture – bind texture

- void glBindTexture(GLenum target, GLuint texture);
  - Bind a named texture to a texturing target
  - Bind the texture before using or setting it
  - target: GL_TEXTURE_1D, GL_TEXTURE_2D, GL_TEXTURE_3D, or GL_TEXTURE_CUBE_MAP
  - texture: specifies the name(id) of a texture

- glBindTexture(target, 0);
  - Unbind texture objects if you don’t want to use them on next objects

Texture – copy texture into GPU

- void glTexImage2D(GLenum target,
  GLint level,
  GLint internalFormat,
  GLsizei width,
  GLsizei height,
  GLint border,
  GLenum format,
  GLenum type,
  const GLvoid* data);
- target:
  - GL_TEXTURE_2D,
  - GL_TEXTURE_CUBE_MAP_POSITIVE_X, 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.
- level:
  - Specifies the level-of-detail number.
  - Level 0 is the base image level. Level n is the nth mipmap reduction image.

Texture – copy texture into GPU

- internalFormat:
  - Specifies the number of color components in the texture. Must be 1, 2, 3, 4 or GL_RGB, GL_RGBA
  - For more details, see https://www.khronos.org/registry/OpenGL-Refpages/gl4/html/glTexImage2D.xhtml
- width:
  - Specifies the width of the texture image.
- height:
  - Specifies the height of the texture image, or the number of layers in a texture array,
- Border:
  - Specifies the width of the border. Must be 0.
- Format:
  - Specifies the format of the pixel data. GL_RED, GL_RG, GL_RGB, GL_BGR, GL_RGBA, GL_BGRA
- Type:
  - Specifies the data type of the pixel data. GL_UNSIGNED_BYTE, GL_BYTE,
- Data:
  - Specifies a pointer to the image data in memory.

Texture – texture environment parameters

- void glTexEnv{fi}[v](GLenum target, GLenum pname, TYPE param);
  - Indicate how the texels are combined with the original pixels
  - target: Specifies a texture environment. Use GL_TEXTURE_ENV
  - pname: Specifies the texture environment parameter. Use GL_TEXTURE_ENV_MODE
  - param: Specifies a single symbolic constant.
    - GL_ADD, GL_MODULATE, GL_DECAL, GL_BLEND, GL_REPLACE, GL_SUBTRACT, GL_COMBINE

format\param	GL_REPLACE	GL_MODULATE	GL_ADD	GL_DECAL	GL_BLEND
RGB	C = Ct	C = Cf * Ct	C = Cf + Ct	C = Ct	C = Cf(1 - Ct) + Cc * Ct
	A = Af	A = Af	A = Af	A = Af	A = Af
RGBA	C = Ct	C = Cf * Ct	C = Cf + Ct	C = Cf(1 - At) + Ct * At	C = Cf(1 - Ct) + Cc * Ct
	A = At	A = Af * At	A = Af * At	A = Af	A = At *Af

Ct = color of the texture

Cf = color of the frame buffer

Cc = color of the texture environment color

At = alpha of the texture

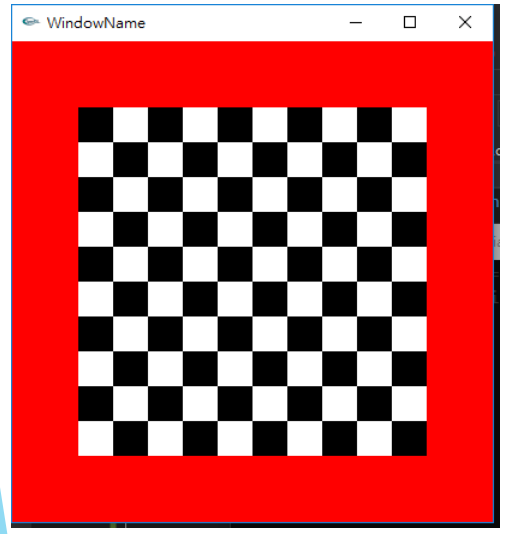
Af = alpha of the frame buffer

Texture – texture parameters

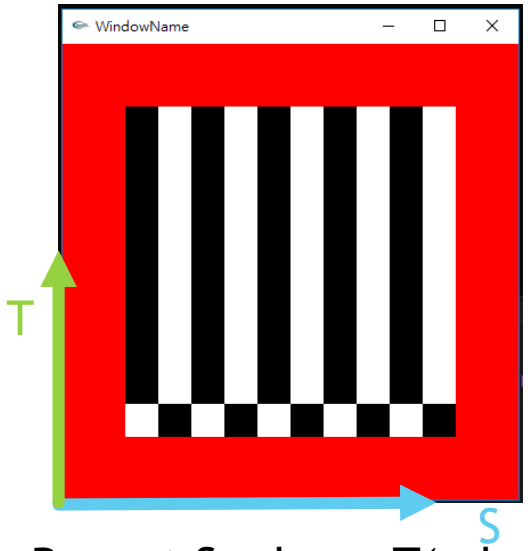
- void glTexParameter{fi}[v](GLenum target, GLenum pname, TYPE param);
- Set texture parameters
  - target: GL_TEXTURE_2D, GL_TEXTURE_CUBE_MAP
  - pname: Specifies the texture parameter. See the following pages
  - param: Specifies the value of pname. See the following pages

Texture – texture parameters

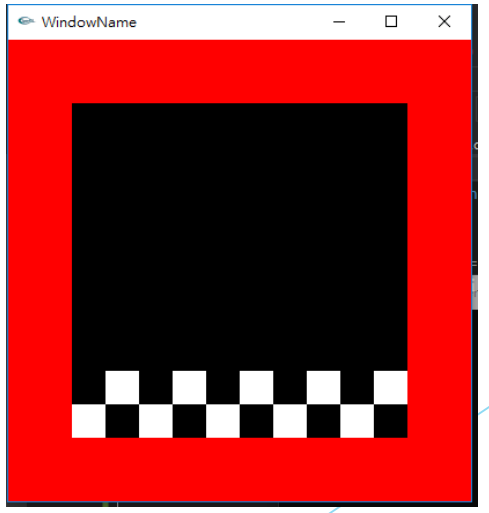
pname	param
GL_TEXTURE_WRAP_S GL_TEXTURE_WRAP_T GL_TEXTURE_WRAP_R	GL_REPEAT, GL_MIRRORED_REPEAT, GL_CLAMP_TO_EDGE, GL_MIRROR_CLAMP_TO_EDGE, GL_CLAMP_TO_BORDER.



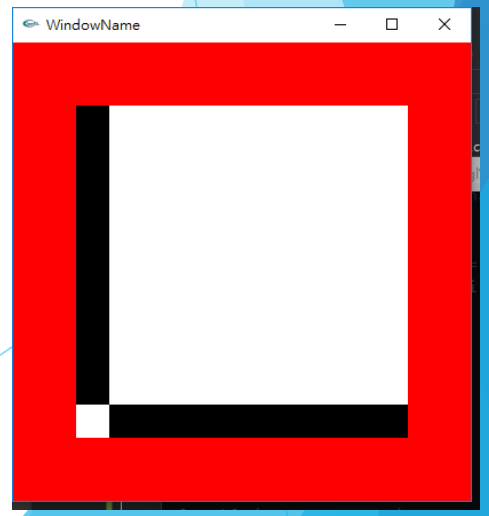
Repeat both S, T



Repeat S, clamp T(edge)



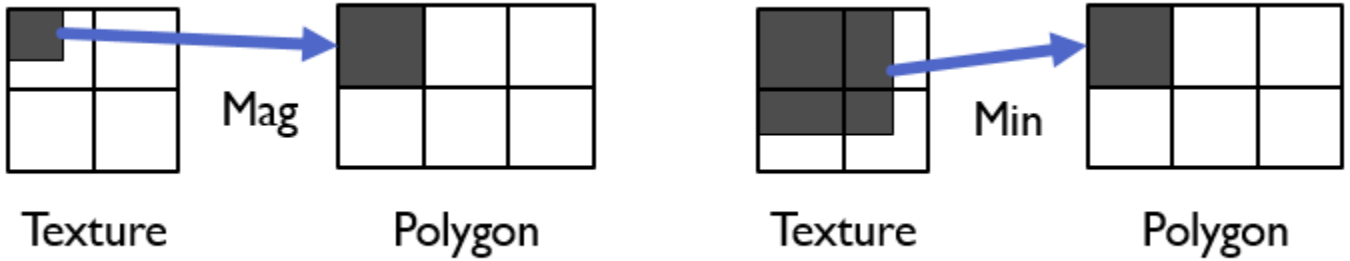
Repeat S, clamp T(border)

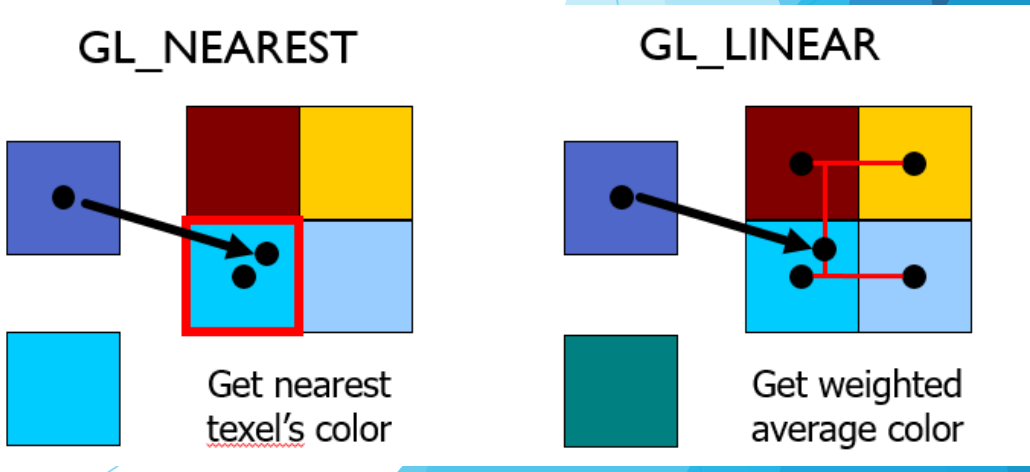


Clamp both S, T(edge)

Texture - texture parameters

pname	param
GL_TEXTURE_MIN_FILTER	GL_NEAREST, GL_LINEAR GL_NEAREST_MIPMAP_NEAREST GL_LINEAR_MIPMAP_NEAREST GL_NEAREST_MIPMAP_LINEAR GL_LINEAR_MIPMAP_LINEAR
GL_TEXTURE_MAG_FILTER	GL_NEAREST, GL_LINEAR

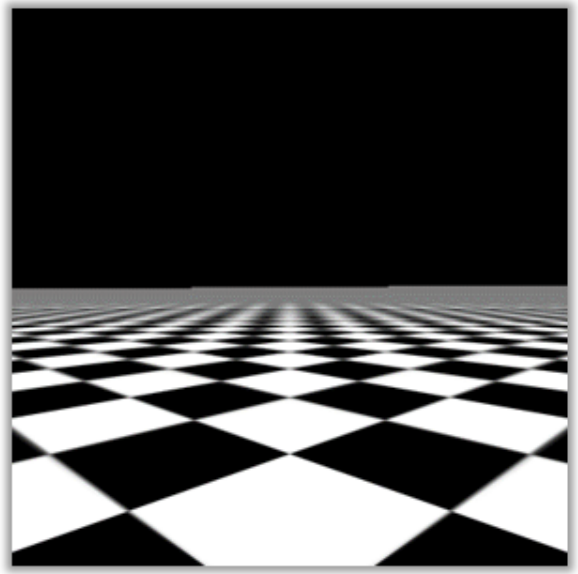




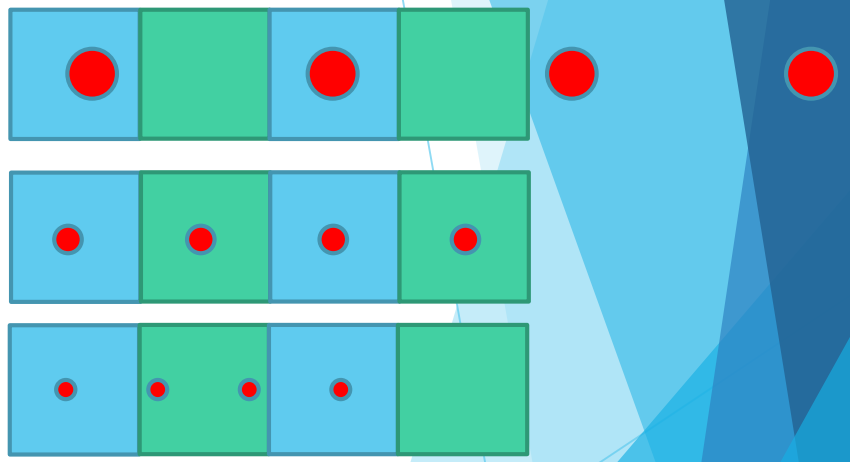
Texture – texture parameters

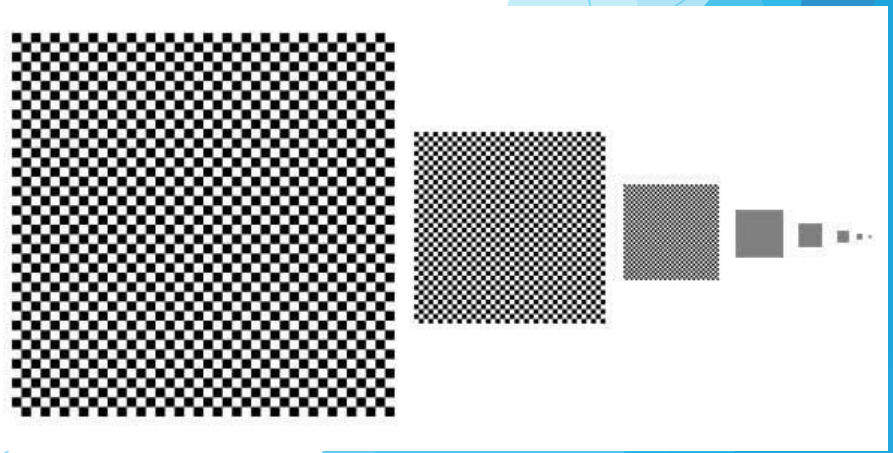
- glTexParameteri(target, GL_TEXTURE_BASE_LEVEL, base_level);
- glTexParameteri(target, GL_TEXTURE_MAX_LEVEL, max_level);
- glGenerateMipmap(GLenum target);
  - Generate the mipmap from the base level to the max level
  - Use it after glTexImage

Nearest



Mipmap

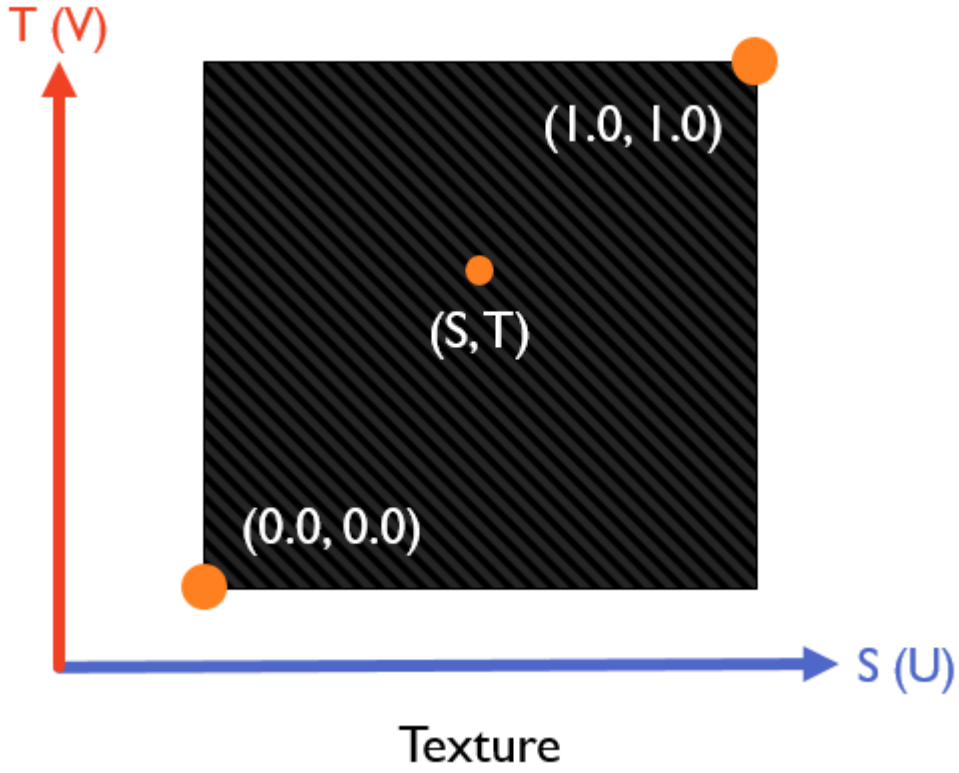




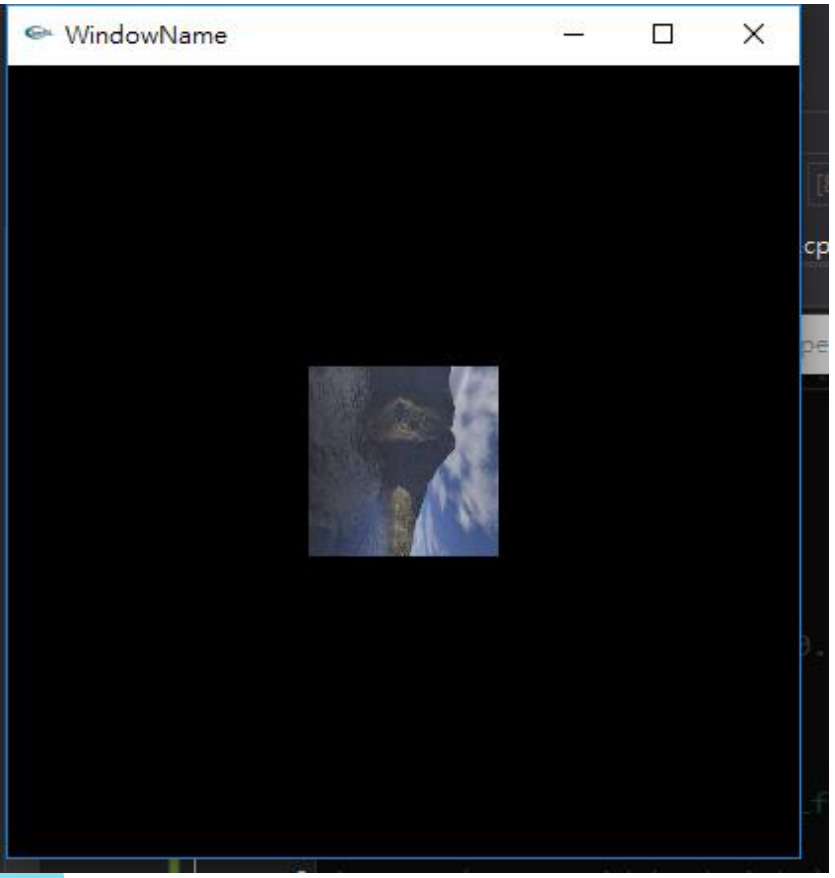
Texture – render with texture

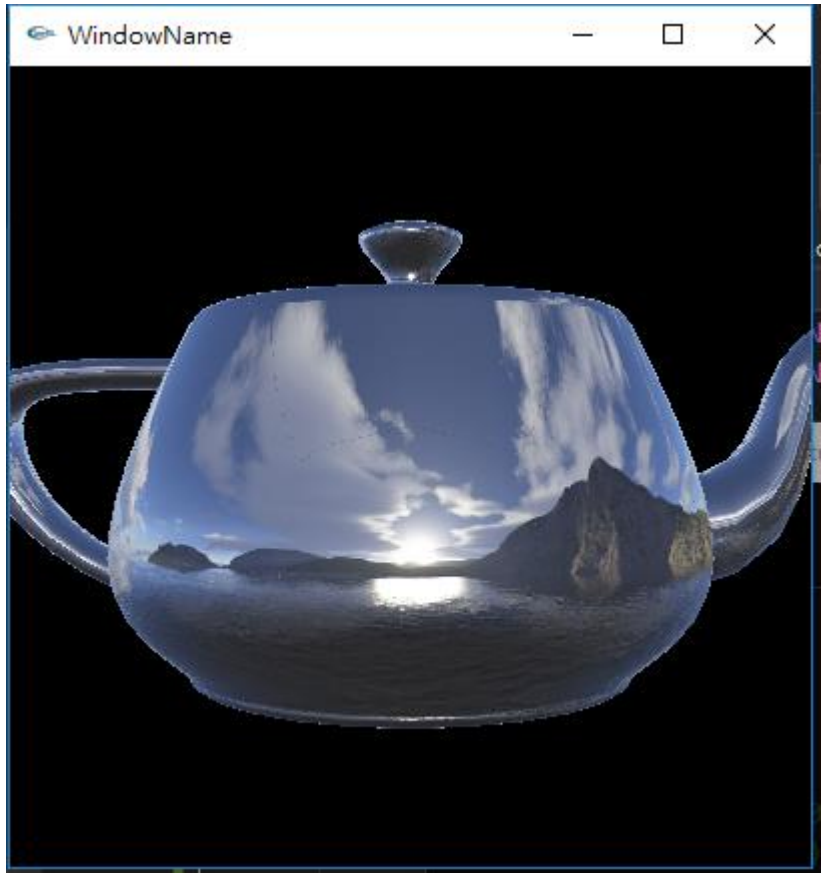
- void glEnable(GLenum cap);
- void glBindTexture(GLenum target, GLuint texture);

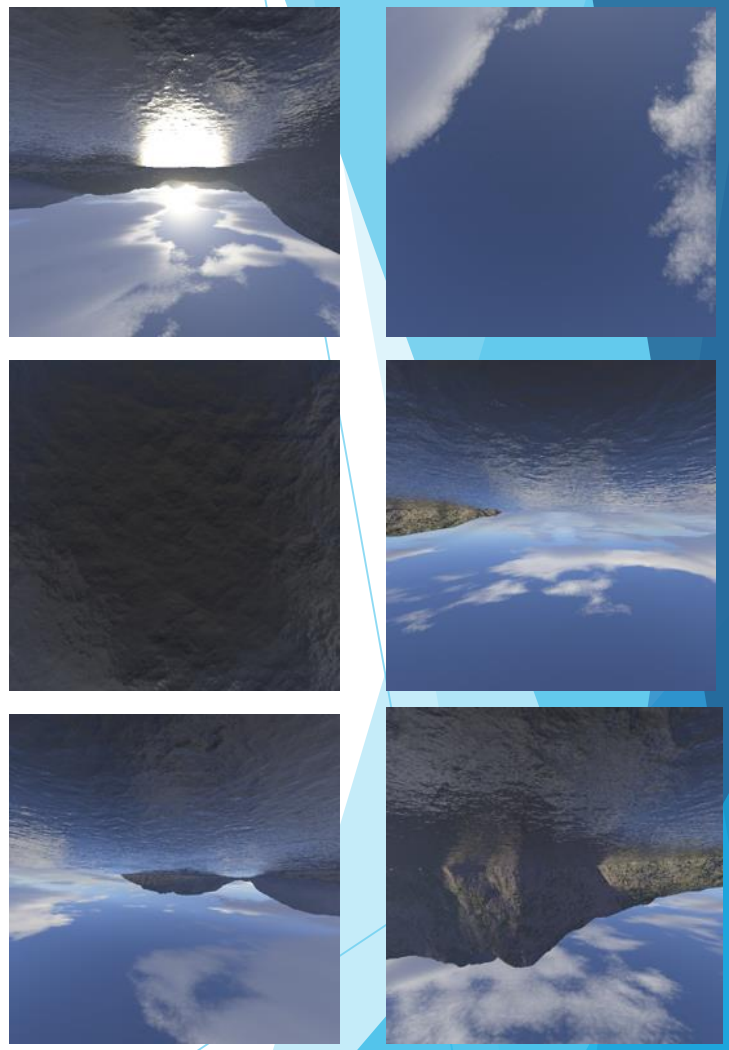
- void glTexCoord{1234}{sifd}[v](TYPE coordinate);
  - Set the current texture coordinates
  - glTexCoord2f is usually used
  - Assign texture coordinate for each vertex
    - glTexCoord2f(u, v);
    - glVertex3f(x, y ,z);



Environment map(cube map)







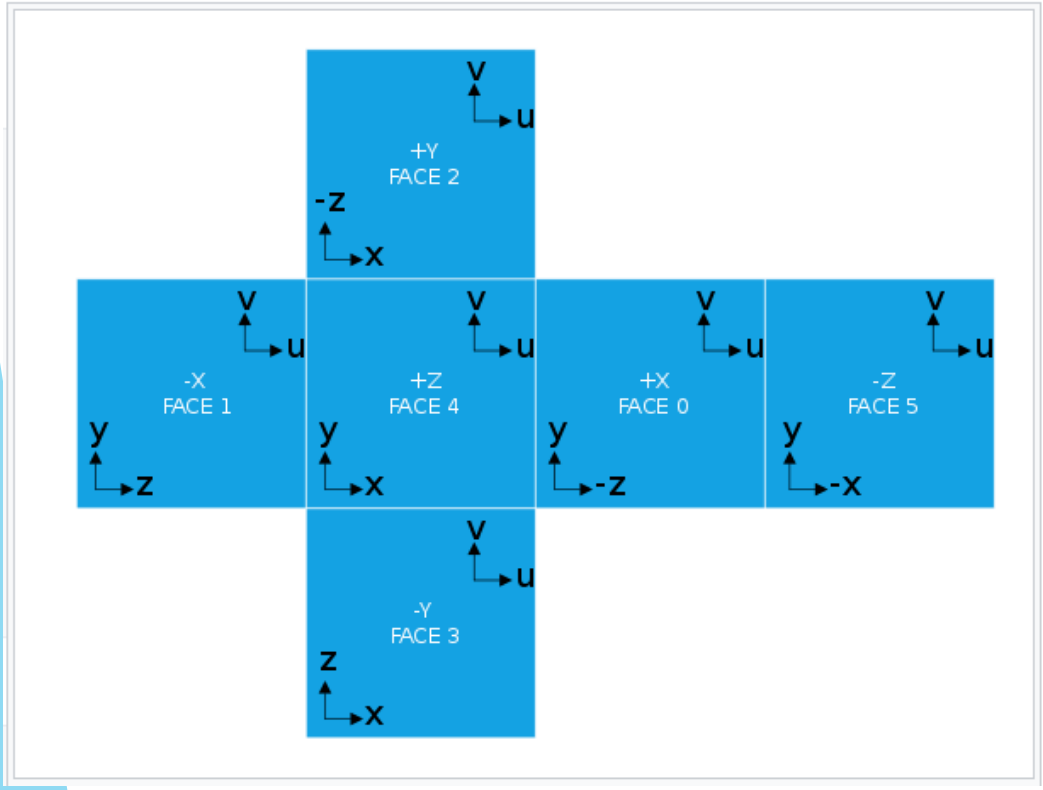
Environment map(cube map)

- Load six textures

```
165 void init_texture()
166 {
167     //enable cube map
168     glEnable(GL_TEXTURE_CUBE_MAP);
169
170     glGenTextures(1, &texture_id);
171     glBindTexture(GL_TEXTURE_CUBE_MAP, texture_id);
172     glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_REPLACE);
173     glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
174     glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
175
176     unsigned char *data = readBMP("./sky2_posx.bmp");
177     glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X, 0, GL_RGB, image_width, image_height, 0, GL_RGB, GL_UNSIGNED_BYTE, data);
178     delete data;
179     data = readBMP("./sky2_negx.bmp");
180     glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_X, 0, GL_RGB, image_width, image_height, 0, GL_RGB, GL_UNSIGNED_BYTE, data);
181     delete data;
182     data = readBMP("./sky2_posy.bmp");
183     glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_Y, 0, GL_RGB, image_width, image_height, 0, GL_RGB, GL_UNSIGNED_BYTE, data);
184     delete data;
185     data = readBMP("./sky2_negy.bmp");
186     glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_Y, 0, GL_RGB, image_width, image_height, 0, GL_RGB, GL_UNSIGNED_BYTE, data);
187     delete data;
188     data = readBMP("./sky2_posz.bmp");
189     glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_Z, 0, GL_RGB, image_width, image_height, 0, GL_RGB, GL_UNSIGNED_BYTE, data);
190     delete data;
191     data = readBMP("./sky2_negz.bmp");
192     glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_Z, 0, GL_RGB, image_width, image_height, 0, GL_RGB, GL_UNSIGNED_BYTE, data);
193     delete data;
194
195     glBindTexture(GL_TEXTURE_2D, 0);
196 }
197
```

Environment map(cube ma

- glTexCoord3f(s, t, r)
  - Positive x:
    - s = 1
    - t, r = -1 ~ 1




```
47  void display()
48  {
49      //ModelView Matrix
50      glMatrixMode(GL_MODELVIEW);
51      glLoadIdentity();
52      gluLookAt(0.0f, 0.0f, 10.0f, 0.0f, 0.0f, 0.0f, 0.0f, 1.0f, 0.0f);
53      //Projection Matrix
54      glMatrixMode(GL_PROJECTION);
55      glLoadIdentity();
56      gluPerspective(45, width / (GLfloat)height, 0.1, 1000);
57      //Viewport Matrix
58      glViewport(0, 0, width, height);
59
60      //clear
61      glClear(GL_COLOR_BUFFER_BIT);
62      glClear(GL_DEPTH_BUFFER_BIT);
63
64      glMatrixMode(GL_MODELVIEW);
65      lighting();
66
67      glBindTexture(GL_TEXTURE_CUBE_MAP, texture_id);
68      glPushMatrix();
69      glBegin(GL_QUADS);
70      glNormal3f(0, 0, 1);
71      glTexCoord3f(1.0f, 1.0f, 1.0f);
72      glVertex3f(1.0f, 1.0f, 0.0f);
73      glNormal3f(0, 0, 1);
74      glTexCoord3f(1.0f, -1.0f, 1.0f);
75      glVertex3f(-1.0f, 1.0f, 0.0f);
76      glNormal3f(0, 0, 1);
77      glTexCoord3f(1.0f, -1.0f, -1.0f);
78      glVertex3f(-1.0f, -1.0f, 0.0f);
79      glNormal3f(0, 0, 1);
80      glTexCoord3f(1.0f, 1.0f, -1.0f);
81      glVertex3f(1.0f, -1.0f, 0.0f);
82      glEnd();
83      glPopMatrix();
84      glBindTexture(GL_TEXTURE_CUBE_MAP, 0);
85
86      glutSwapBuffers();
87  }
```

Environment map(cube map)

- void glTexGen{ifd}[v](GLenum coord, GLuint pname, GLint param);
  - Control the generation of texture coordinates
  - coord: GL_S, GL_T, GL_R, or GL_Q
  - pname: GL_TEXTURE_GEN_MODE
  - param: GL_REFLECTION_MAP, GL_SPHERE_MAP,
- If enable texture generation,
- the texture coordinates (glTexCoord) will be overwrote.

```
47  void display()
48  {
49      //ModelView Matrix
50      glMatrixMode(GL_MODELVIEW);
51      glLoadIdentity();
52      gluLookAt(0.0f, 0.0f, 10.0f, 0.0f, 0.0f, 0.0f, 0.0f, 1.0f, 0.0f);
53      //Projection Matrix
54      glMatrixMode(GL_PROJECTION);
55      glLoadIdentity();
56      gluPerspective(45, width / (GLfloat)height, 0.1, 1000);
57      //Viewport Matrix
58      glViewport(0, 0, width, height);
59
60      //clear
61      glClear(GL_COLOR_BUFFER_BIT);
62      glClear(GL_DEPTH_BUFFER_BIT);
63
64      glMatrixMode(GL_MODELVIEW);
65      lighting();
66
67      glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_REFLECTION_MAP);
68      glTexGeni(GL_T, GL_TEXTURE_GEN_MODE, GL_REFLECTION_MAP);
69      glTexGeni(GL_R, GL_TEXTURE_GEN_MODE, GL_REFLECTION_MAP);
70      glEnable(GL_TEXTURE_GEN_S);
71      glEnable(GL_TEXTURE_GEN_T);
72      glEnable(GL_TEXTURE_GEN_R);
73
74      glBindTexture(GL_TEXTURE_CUBE_MAP, texture_id);
75      glPushMatrix();
76      glRotatef(degree, 0, 1, 0);
77      glutSolidTeapot(3);
78      glBindTexture(GL_TEXTURE_CUBE_MAP, 0);
79
80      glutSwapBuffers();
81  }
```