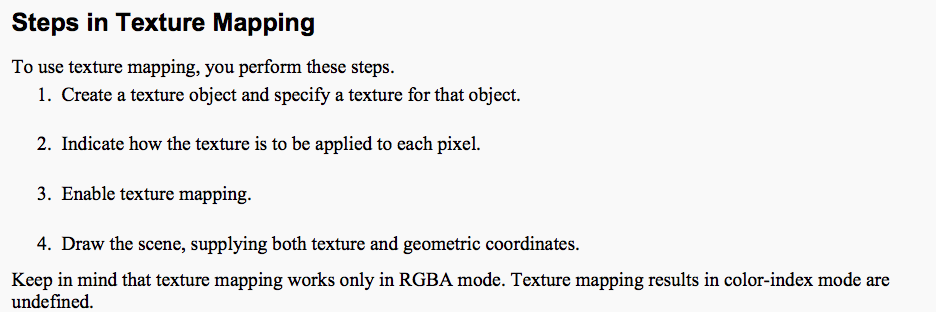
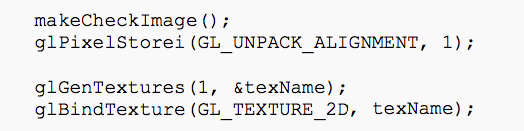
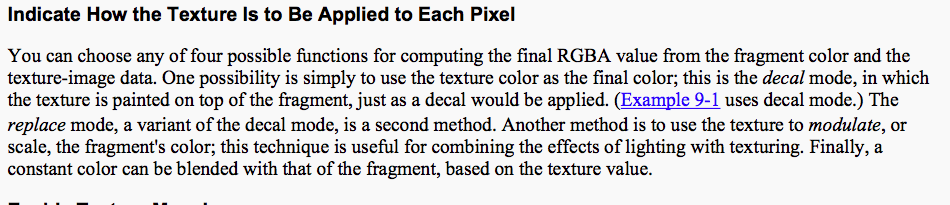
OpenGL RedBook Texture discussion



1.



2. 

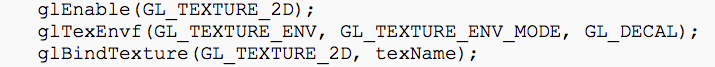
If using ENV\_MODE

* Decal – paint texture over fragment
* Replace – replaces fragment color with texture color
* Blend – blends fragment color with texture color based on alpha value of RGBA
* Modulate – scales fragment color by texture value (used with lighting of textured scenes)

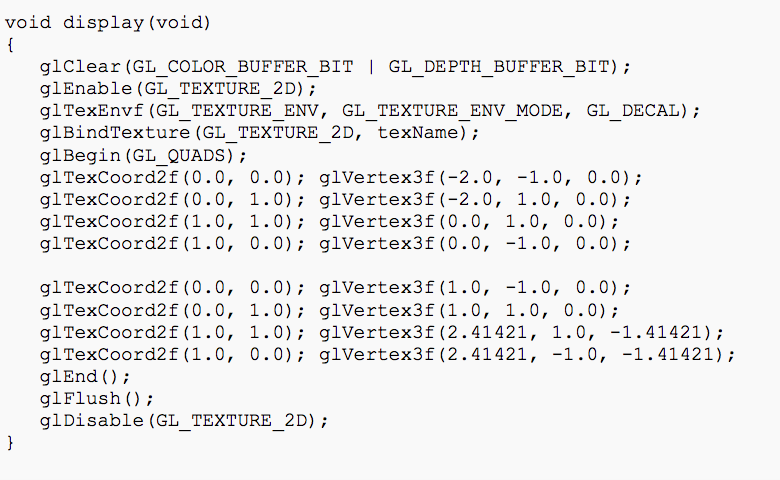
If using ENV\_COLOR and

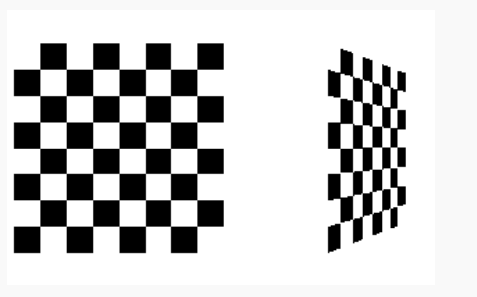
* Blend – blends environmental color (an RGBA value) with texture color

2 and 3 (set Decal mode, enable texturing and bind the texture ID to the texture image)

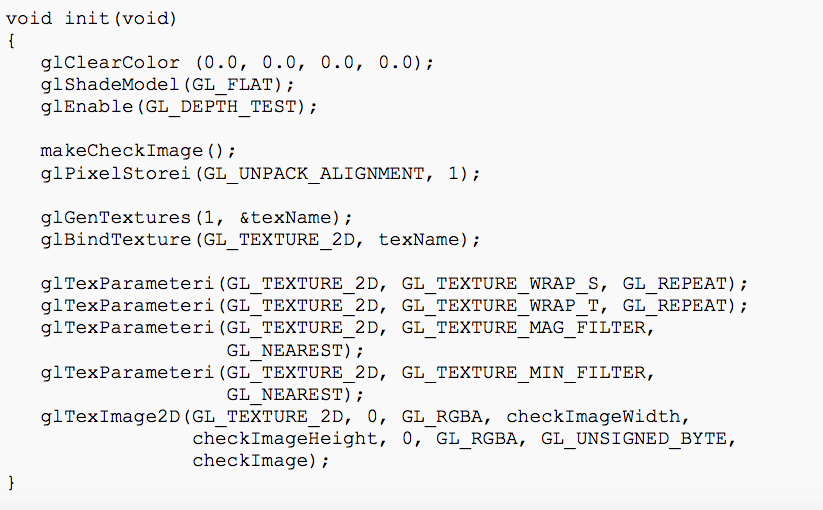


4. Draw the Scene, Supplying Both Texture and Geometric Coordinates

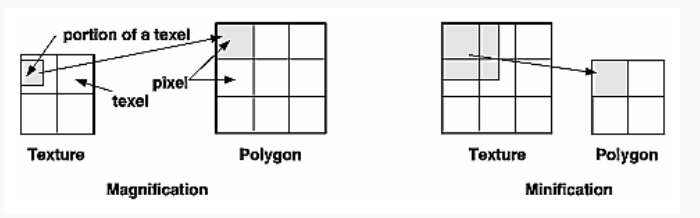


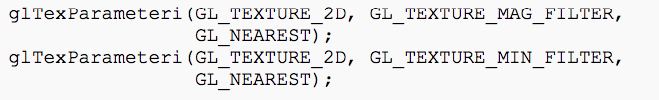


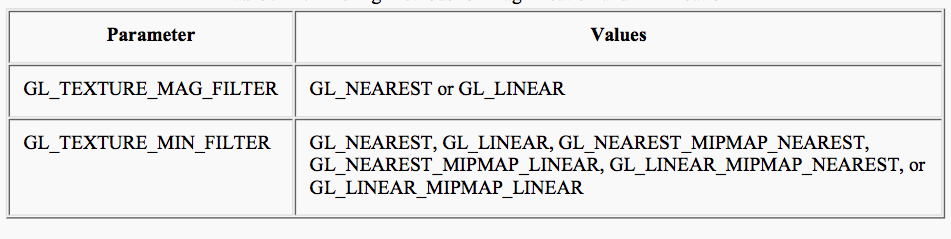
5. Still have to specify how the texture is filtered – in the init we see an example of applying magnification and minification filters.



Filtering







* Nearest – texel value with coodinates nearest center of pixel is used
* Linear – weighted average of a 2X2 array of texels that lie nearest the center of the pixel is used (better results, more expensive)
* MipMap – used for different resolutions of texture image (as an object moves into the distance its texture is minified – we see less detail).

Complete code for this example is,

/\* checker.c

\* This program texture maps a checkerboard image onto

\* two rectangles.

\*

\* If running this program on OpenGL 1.0, texture objects are

\* not used.

\*/

#include <GL/glut.h>

#include <stdlib.h>

#include <stdio.h>

/\* Create checkerboard texture \*/

#define checkImageWidth 64

#define checkImageHeight 64

static GLubyte checkImage[checkImageHeight][checkImageWidth][4];

#ifdef GL\_VERSION\_1\_1

static GLuint texName;

#endif

void makeCheckImage(void)

{

int i, j, c;

for (i = 0; i < checkImageHeight; i++) {

for (j = 0; j < checkImageWidth; j++) {

c = ((((i&0x8)==0)^((j&0x8))==0))\*255;

checkImage[i][j][0] = (GLubyte) c;

checkImage[i][j][1] = (GLubyte) c;

checkImage[i][j][2] = (GLubyte) c;

checkImage[i][j][3] = (GLubyte) 255;

}

}

}

void init(void)

{

glClearColor (0.0, 0.0, 0.0, 0.0);

glShadeModel(GL\_FLAT);

glEnable(GL\_DEPTH\_TEST);

makeCheckImage();

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1);

#ifdef GL\_VERSION\_1\_1

glGenTextures(1, &texName);

glBindTexture(GL\_TEXTURE\_2D, texName);

#endif

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_REPEAT);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_REPEAT);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_NEAREST);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_NEAREST);

#ifdef GL\_VERSION\_1\_1

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGBA, checkImageWidth, checkImageHeight,

0, GL\_RGBA, GL\_UNSIGNED\_BYTE, checkImage);

#else

glTexImage2D(GL\_TEXTURE\_2D, 0, 4, checkImageWidth, checkImageHeight,

0, GL\_RGBA, GL\_UNSIGNED\_BYTE, checkImage);

#endif

}

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glEnable(GL\_TEXTURE\_2D);

glTexEnvf(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_DECAL);

#ifdef GL\_VERSION\_1\_1

glBindTexture(GL\_TEXTURE\_2D, texName);

#endif

glBegin(GL\_QUADS);

glTexCoord2f(0.0, 0.0); glVertex3f(-2.0, -1.0, 0.0);

glTexCoord2f(0.0, 1.0); glVertex3f(-2.0, 1.0, 0.0);

glTexCoord2f(1.0, 1.0); glVertex3f(0.0, 1.0, 0.0);

glTexCoord2f(1.0, 0.0); glVertex3f(0.0, -1.0, 0.0);

glTexCoord2f(0.0, 0.0); glVertex3f(1.0, -1.0, 0.0);

glTexCoord2f(0.0, 1.0); glVertex3f(1.0, 1.0, 0.0);

glTexCoord2f(1.0, 1.0); glVertex3f(2.41421, 1.0, -1.41421);

glTexCoord2f(1.0, 0.0); glVertex3f(2.41421, -1.0, -1.41421);

glEnd();

glFlush();

glDisable(GL\_TEXTURE\_2D);

}

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei) w, (GLsizei) h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(60.0, (GLfloat) w/(GLfloat) h, 1.0, 30.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glTranslatef(0.0, 0.0, -3.6);

}

void keyboard (unsigned char key, int x, int y)

{

switch (key) {

case 27:

exit(0);

break;

default:

break;

}

}

int main(int argc, char\*\* argv)

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowSize(250, 250);

glutInitWindowPosition(100, 100);

glutCreateWindow(argv[0]);

init();

glutDisplayFunc(display);

glutReshapeFunc(reshape);

glutKeyboardFunc(keyboard);

glutMainLoop();

return 0;

}