

Código 16: Textura procedural

```
/* basado en texbind.c del libro rojo de OpenGL*/

#include <GL/glut.h>
#define checkImageWidth 64
#define checkImageHeight 64

static GLubyte checkImage[checkImageHeight][checkImageWidth][3];
static GLubyte otherImage[checkImageHeight][checkImageWidth][3];

void makeCheckImages(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;
            c = (((i&0x10)==0)^((j&0x10)==0))*255;
            otherImage[i][j][0] = (GLubyte) 0;
            otherImage[i][j][1] = (GLubyte) c;
            otherImage[i][j][2] = (GLubyte) 0;
        }
}

void init(void)
{
    glClearColor (0.0, 0.0, 0.0, 0.0);

    makeCheckImages();
    glBindTexture(GL_TEXTURE_2D, 1000);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, checkImageWidth, checkImageHeight,
                0, GL_RGB, GL_UNSIGNED_BYTE, checkImage);
    glBindTexture(GL_TEXTURE_2D, 1010);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, checkImageWidth, checkImageHeight,
                0, GL_RGB, GL_UNSIGNED_BYTE, otherImage);

    glEnable(GL_TEXTURE_2D);
}

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);

    glBindTexture(GL_TEXTURE_2D, 1000);
    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);
    glEnd();

    glBindTexture(GL_TEXTURE_2D, 1010);
    glBegin(GL_QUADS);
    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();
}
```

```
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);
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(250, 250);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("Textura procedural");
    init();
    glutReshapeFunc(reshape);
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

