# OpenGL Texture Mapping Tutorial

Adding textures to objects in OpenGL involves mapping an image onto the surface of a shape. This tutorial explains the steps to apply a texture to an object, specifically a 2D quad.

---

1. **Prerequisites**

Before starting, ensure you have:

- A basic understanding of OpenGL and C++.
- OpenGL installed (along with GLUT).

---

2. **Workflow Overview**

1. Load an image into your program.
2. Generate a texture in OpenGL.
3. Bind the texture and configure texture parameters.
4. Map the texture onto a shape using texture coordinates.

---

3. **Step-by-Step Process**

Step 1: Setup Your OpenGL Program

Start with a basic OpenGL setup with a display function to draw a quad.

```
#include <GL/glut.h>

void display() {
  glClear(GL_COLOR_BUFFER_BIT);
  glLoadIdentity();

  // Draw a simple quad
  glBegin(GL_QUADS);
    glVertex2f(-0.5f, -0.5f);
    glVertex2f(0.5f, -0.5f);
    glVertex2f(0.5f, 0.5f);
    glVertex2f(-0.5f, 0.5f);
  glEnd();
```

```
  glutSwapBuffers();
}

void init() {
  glClearColor(0.2f, 0.3f, 0.3f, 1.0f); // Background color
}

int main(int argc, char** argv) {
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
  glutInitWindowSize(800, 600);
  glutCreateWindow("Basic Quad");

  init();
  glutDisplayFunc(display);
  glutMainLoop();
  return 0;
}
```

---

Step 2: Load the Texture

For simplicity, we use a built-in checkerboard pattern in this tutorial. To use image files (e.g., BMP, PNG), you can use libraries like SOIL, stb_image, or FreeImage.

Generate a Checkerboard Texture:

```
const int TEX_WIDTH = 64;
const int TEX_HEIGHT = 64;
unsigned char textureData[TEX_HEIGHT][TEX_WIDTH][3];

void generateCheckerboardTexture() {
  for (int i = 0; i < TEX_HEIGHT; ++i) {
    for (int j = 0; j < TEX_WIDTH; ++j) {
      int checker = ((i / 8) % 2 == 0) ^ ((j / 8) % 2 == 0);
      unsigned char color = checker * 255;
      textureData[i][j][0] = color; // Red
      textureData[i][j][1] = color; // Green
      textureData[i][j][2] = color; // Blue
    }
  }
}
```

---

Step 3: Create and Bind the Texture

```
GLuint textureID;


void setupTexture() {
  generateCheckerboardTexture();


  glEnable(GL_TEXTURE_2D);
  glGenTextures(1, &textureID);
  glBindTexture(GL_TEXTURE_2D, textureID);


  // Set texture parameters
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);


  // Create the texture
  glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, TEX_WIDTH, TEX_HEIGHT, 0, GL_RGB,
GL_UNSIGNED_BYTE, textureData);
}
```

---

Step 4: Apply Texture to the Quad

Update the display() function to include texture mapping.

```
void display() {
  glClear(GL_COLOR_BUFFER_BIT);
  glLoadIdentity();

  glEnable(GL_TEXTURE_2D);
  glBindTexture(GL_TEXTURE_2D, textureID);

  glBegin(GL_QUADS);
    glTexCoord2f(0.0f, 0.0f); glVertex2f(-0.5f, -0.5f);
    glTexCoord2f(1.0f, 0.0f); glVertex2f(0.5f, -0.5f);
    glTexCoord2f(1.0f, 1.0f); glVertex2f(0.5f, 0.5f);
    glTexCoord2f(0.0f, 1.0f); glVertex2f(-0.5f, 0.5f);
  glEnd();

  glDisable(GL_TEXTURE_2D);
  glutSwapBuffers();
}
```

---

Step 5: Initialize Everything

Update the main() function to call setupTexture().

```
int main(int argc, char** argv) {
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
  glutInitWindowSize(800, 600);
  glutCreateWindow("Textured Quad");
```

```
    init();
    setupTexture(); // Initialize the texture
    glutDisplayFunc(display);

    glutMainLoop();
    return 0;
}
```

---

4. Result

Running the code will display a 2D quad with a checkerboard texture applied.