* glClearColor(): Sets the clear color for the window.
* glClear(): Clears buffers (color, depth, etc.).
* glViewport(): Sets the viewport size and location.
* glEnable(): Enables a specific OpenGL capability (e.g., GL\_DEPTH\_TEST).
* glDisable(): Disables a specific OpenGL capability.
* glBegin(), glEnd(): Used to define a primitive (e.g., points, lines, triangles).
* glVertex2f(), glVertex3f(): Specifies vertex coordinates.
* glColor3f(), glColor4f(): Specifies color for subsequent vertices.
* glLineWidth(): Sets the width of lines.
* glPushMatrix(), glPopMatrix(): Saves and restores the current matrix.
* glLoadIdentity(): Resets the current matrix to the identity matrix.
* glTranslatef(), glRotatef(), glScalef(): Apply transformations (translation, rotation, scaling).
* glGenBuffers(): Generates buffer objects.
* glBindBuffer(): Binds a buffer object to a target.
* glBufferData(): Transfers data to a buffer object.
* glGenTextures(): Generates a texture name.
* glBindTexture(): Binds a texture to a target.
* glTexImage2D(): Specifies a 2D texture image.
* glTexParameteri(): Sets texture parameters.
* glCreateShader(): Creates a shader object.
* glShaderSource(): Sets the source code for a shader.
* glCompileShader(): Compiles a shader.
* glCreateProgram(): Creates a program object.
* glAttachShader(): Attaches a shader to a program.
* glLinkProgram(): Links shaders into a program.
* glUseProgram(): Makes a program the current shader program.
* glGenFramebuffers(): Generates a framebuffer object.
* glBindFramebuffer(): Binds a framebuffer object.
* glFramebufferTexture2D(): Attaches a texture to a framebuffer.
* glGetError(): Retrieves the most recent error.
* glGetString(): Retrieves OpenGL-related information (like version, vendor).

SQUARE

#include <GL/glut.h>

#include <math.h>

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_TRIANGLES);

glVertex2f(-0.5f, -0.5f);

glVertex2f(-0.5f, 0.5f);

glVertex2f( 0.5f, 0.5f);

glVertex2f(0.5f, -0.5f);

glVertex2f(-0.5f, -0.5f);

glVertex2f(0.5f, 0.5f);

glEnd();

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutCreateWindow("Sample");

glClearColor(0.0, 0.0, 0.0, 1.0);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

Rectangle

#include <GL/glut.h>

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_QUADS);

glVertex2f(-0.6f, -0.3f);

glVertex2f(-0.6f, 0.3f);

glVertex2f(0.6f, 0.3f);

glVertex2f(0.6f, -0.3f);

glEnd();

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutCreateWindow("Rectangle");

glClearColor(0.0, 0.0, 0.0, 1.0);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

Triangle

#include <GL/glut.h>

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_TRIANGLES);

glVertex2f(-0.5f, -0.5f);

glVertex2f(0.5f, -0.5f);

glVertex2f(0.0f, 0.5f);

glEnd();

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutCreateWindow("Triangle");

glClearColor(0.0, 0.0, 0.0, 1.0);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

Circle

#include <GL/glut.h>

#include <cmath>

const int numSegments = 100;

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_TRIANGLE\_FAN);

glVertex2f(0.0f, 0.0f);

for (int i = 0; i <= numSegments; i++) {

float angle = i \* 2.0f \* M\_PI / numSegments;

float x = 0.5f \* cos(angle);

float y = 0.5f \* sin(angle);

glVertex2f(x, y);

}

glEnd();

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutCreateWindow("Circle");

glClearColor(0.0, 0.0, 0.0, 1.0);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

Hexagon

#include <GL/glut.h>

#include <cmath>

const int numSegments = 6;

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_TRIANGLE\_FAN);

glVertex2f(0.0f, 0.0f);

for (int i = 0; i <= numSegments; i++) {

float angle = i \* 2.0f \* M\_PI / numSegments;

float x = 0.5f \* cos(angle);

float y = 0.5f \* sin(angle);

glVertex2f(x, y);

}

glEnd();

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutCreateWindow("Circle");

glClearColor(0.0, 0.0, 0.0, 1.0);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

Decagon

#include <GL/glut.h>

#include <cmath>

const int numSegments = 10;

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_TRIANGLE\_FAN);

glVertex2f(0.0f, 0.0f);

for (int i = 0; i <= numSegments; i++) {

float angle = i \* 2.0f \* M\_PI / numSegments;

float x = 0.5f \* cos(angle);

float y = 0.5f \* sin(angle);

glVertex2f(x, y);

}

glEnd();

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutCreateWindow("Circle");

glClearColor(0.0, 0.0, 0.0, 1.0);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

Square with rectangle inside

#include <GL/glut.h>

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0, 1.0, 1.0);

glBegin(GL\_LINE\_LOOP);

glVertex2f(-0.5f, -0.5f);

glVertex2f(-0.5f, 0.5f);

glVertex2f(0.5f, 0.5f);

glVertex2f(0.5f, -0.5f);

glEnd();

glColor3f(0.0, 1.0, 0.0);

glBegin(GL\_QUADS);

glVertex2f(-0.4f, -0.2f);

glVertex2f(-0.4f, 0.2f);

glVertex2f(0.4f, 0.2f);

glVertex2f(0.4f, -0.2f);

glEnd();

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutCreateWindow("Square with Rectangle Inside");

glClearColor(0.0, 0.0, 0.0, 1.0);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

Square with Triangle Inside

#include <GL/glut.h>

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0, 1.0, 1.0);

glBegin(GL\_LINE\_LOOP);

glVertex2f(-0.5f, -0.5f);

glVertex2f(-0.5f, 0.5f);

glVertex2f(0.5f, 0.5f);

glVertex2f(0.5f, -0.5f);

glEnd();

glColor3f(0.0, 0.0, 1.0);

glBegin(GL\_TRIANGLES);

glVertex2f(-0.3f, -0.3f);

glVertex2f(0.3f, -0.3f);

glVertex2f(0.0f, 0.3f);

glEnd();

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutCreateWindow("Square with Triangle Inside");

glClearColor(0.0, 0.0, 0.0, 1.0);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

Triangle with Circle Inside

#include <GL/glut.h>

#include <cmath>

const int numSegments = 100;

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0, 1.0, 1.0);

glBegin(GL\_TRIANGLES);

glVertex2f(-0.6f, -0.6f);

glVertex2f(0.6f, -0.6f);

glVertex2f(0.0f, 0.6f);

glEnd();

glColor3f(0.0, 1.0, 0.0);

glBegin(GL\_TRIANGLE\_FAN);

glVertex2f(0.0f, 0.0f); // Center of the circle

for (int i = 0; i <= numSegments; i++) {

float angle = i \* 2.0f \* M\_PI / numSegments;

float x = 0.2f \* cos(angle);

float y = 0.2f \* sin(angle);

glVertex2f(x, y);

}

glEnd();

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutCreateWindow("Triangle with Circle Inside");

glClearColor(0.0, 0.0, 0.0, 1.0);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

Triangle with Centered Square

#include <GL/glut.h>

void drawTriangle() {

glColor3f(1.0, 1.0, 1.0);

glBegin(GL\_TRIANGLES);

glVertex2f(-0.6f, -0.6f);

glVertex2f(0.6f, -0.6f);

glVertex2f(0.0f, 0.6f);

glEnd();

}

void drawSquareInsideTriangle() {

glColor3f(0.0, 1.0, 0.0);

glBegin(GL\_QUADS);

glVertex2f(-0.2f, -0.2f);

glVertex2f(-0.2f, 0.2f);

glVertex2f(0.2f, 0.2f);

glVertex2f(0.2f, -0.2f);

glEnd();

}

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

drawTriangle();

drawSquareInsideTriangle();

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutCreateWindow("Triangle with Centered Square");

glClearColor(0.0, 0.0, 0.0, 1.0);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

Circle with Centered Triangle

#include <GL/glut.h>

#include <cmath>

const int numSegments = 100;

void drawCircle() {

glColor3f(1.0, 0.0, 0.0); // Red color for the circle

glBegin(GL\_TRIANGLE\_FAN);

glVertex2f(0.0f, 0.0f); // Center of the circle

for (int i = 0; i <= numSegments; i++) {

float angle = i \* 2.0f \* M\_PI / numSegments;

float x = 0.4f \* cos(angle);

float y = 0.4f \* sin(angle);

glVertex2f(x, y);

}

glEnd();

}

void drawTriangleInsideCircle() {

glColor3f(0.0, 0.0, 1.0); // Blue color for the triangle

glBegin(GL\_TRIANGLES);

glVertex2f(-0.2f, -0.2f);

glVertex2f(0.2f, -0.2f);

glVertex2f(0.0f, 0.2f);

glEnd();

}

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

drawCircle();

drawTriangleInsideCircle();

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutCreateWindow("Circle with Centered Triangle");

glClearColor(0.0, 0.0, 0.0, 1.0);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}