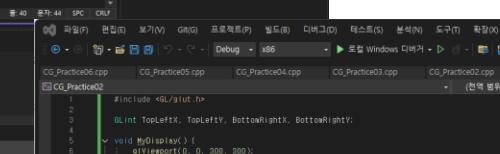


```
CG_Practice06.cpp CG_Practice05.cpp CG_Practice04.cpp CG_Practice03.cpp CG_Practice02.cpp CG_Practice01.cpp CG_Practice00.cpp CG_Practice06.h CG_Practice05.h CG_Practice04.h CG_Practice03.h CG_Practice02.h CG_Practice01.h CG_Practice00.h

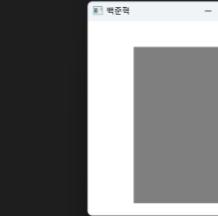
#include <GL/glut.h>
2
3
4 GLint TopLeftX, TopLeftY, BottomRightX, BottomRightY;
5
6 void MyInit() {
7     glClearColor(0.0, 0.0, 0.0);
8     glEnable(GL_COLOR_BUFFER_BIT);
9     glShadeModel(GL_SMOOTH);
10    glPixelZoom(0.5, 0.5);
11
12    glBegin(GL_POLYGON);
13        glVertex3f(TopLeftX / 300.0, (300 - TopLeftY) / 300.0, 0.0);
14        glVertex3f(TopLeftX / 300.0, (300 - BottomRightY) / 300.0, 0.0);
15        glVertex3f(BottomRightX / 300.0, (300 - BottomRightY) / 300.0, 0.0);
16        glVertex3f(BottomRightX / 300.0, (300 - TopLeftY) / 300.0, 0.0);
17    glEnd();
18
19 void MyMouseClick(GLint Button, GLint State, GLint X, GLint Y) {
20     if (Button == GLUT_LEFT_BUTTON && State == GLUT_DOWN) {
21         TopLeftX = X;
22         TopLeftY = Y;
23     }
24 }
25
26 void MyMouseMove(GLint X, GLint Y) {
27     BottomRightX = X;
28     BottomRightY = Y;
29     glutPostRedisplay();
30 }
31
32 int main(int argc, char** argv) {
33     glutInit(&argc, argv);
34     glutInitDisplayMode(GLUT_RGB);
35     glutCreateWindow("OpenGL Practice");
36     glutInitWindowSize(300, 300);
37     glutCreateWindow("OpenGL Practice");
38     glutReshapeFunc(MyReshape);
39     glutDisplayFunc(MyDisplay);
40     glutMouseFunc(MyMouseClick);
41     glutMotionFunc(MyMouseMove);
42     glutKeyboardFunc(MyKeyboard);
43
44     glutMainLoop();
45
46     return 0;
47 }
```



```

7         glClearColor(GL_COLOR_BUFFER_BIT);
8         glClear(GL_COLOR_BUFFER_BIT);
9         glColor3f(0.5, 0.5, 0.5);
10        glVertex3f(TopLeftX / 300.0, (300 - TopLeftY) / 300.0, 0.0);
11        glVertex3f(TopLeftX / 300.0, (300 - BottomRight) / 300.0, 0.0);
12        glVertex3f(BottomRight / 300.0, (300 - BottomRight) / 300.0, 0.0);
13        glVertex3f(BottomRight / 300.0, (300 - TopLeft) / 300.0, 0.0);
14    glEnd();
15    glFlush();
16
17
18    void MyMouseClick(GLint Button, GLint State, GLint X, GLint Y) {
19        if (Button == GLUT_LEFT_BUTTON && State == GLUT_DOWN) {
20            TopLeftX = X;
21            TopLeftY = Y;
22        }
23
24
25    void MyMouseMove(GLint X, GLint Y) {
26        BottomRightX = X;
27        BottomRightY = Y;
28        glutPostRedisplay();
29
30
31    int main(int argc, char** argv) {
32        glutInit(&argc, &argv);
33        glutInitDisplayMode(GLUT_RGB);
34        glutInitWindowSize(300, 300);
35        glutInitWindowPosition(0, 0);
36        glutCreateWindow("OpenGL Project");
37        glClearColor(0.0, 1.0, 1.0);
38        glMaterialfv(GL_FRONT, GL_AMBIENT, {0.0, 1.0, -1.0, 1.0});
39        glutDisplayFunc(MyDisplay);
40        glutMouseFunc(MyMouseClick);
41        glutMotionFunc(MyMouseMove);
42        glutMouseLeaveFunc();
43        glutMainLoop();
44
45        return 0;
46

```



Windows 디버그 Debug x86 GitHub Copilot

CG_Practice06.cpp CG_Practice05.cpp CG_Practice04.cpp CG_Practice03.cpp CG_Practice02.cpp CG_Practice01.cpp

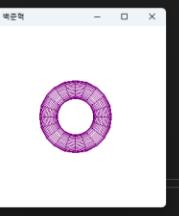
```
#include <GL/glut.h>
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
```

void MyDisplay() {
 if(isSphere == true) {
 glColor3f(0.5, 0.0, 0.5);
 if(isSphere)
 glutWireSphere(0.2, 15, 15);
 else
 glutWireTorus(0.1, 0.5, 40, 20);
 glFlush();
 }
}

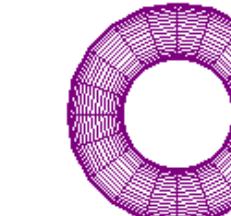
void MyIdle(int entryID) {
 if(entryID == 1)
 isSphere = true;
 else if(entryID == 2)
 isSphere = false;
 else if(entryID == 3)
 exit(0);
 glutPostRedisplay();
}

int main(int argc, char** argv) {
 glutInit(&argc, argv);
 glutInitDisplayMode(GLUT_RGB);
 glutInitWindowSize(500, 500);
 glutCreateWindow("백준학");
 glutReshapeFunc(MyReshape);
 glutIdleFunc(MyIdle);
 glutMainLoop();
 return 0;
}

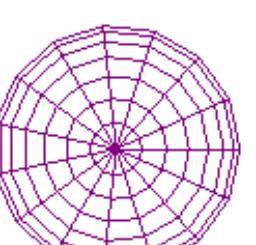
백준학



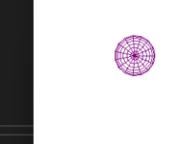
백준학



백준학



백준학



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

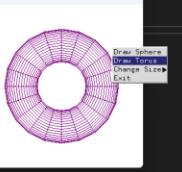
GLboolean isSphere = true;
GLboolean isSmall = true;

void MyDisplay() {
    glClear(GL_COLOR_BUFFER_BIT);
    gluOrtho2D(0, 1000, 0, 1000);
    if ((isSphere) && (!isSmall))
        gluSphere(isSmall ? 0.2 : 15, 15);
    else if ((isSphere) && (!isSmall))
        gluSphere(isSmall ? 0.4 : 15, 15);
    else if ((!isSphere) && (!isSmall))
        gluSphere(isSmall ? 0.4 : 40, 20);
    else
        gluSphere(isSmall ? 0.2 : 0.5, 40, 20);
    glutPostRedisplay();
}

void MyMenu(int entryID) {
    if (entryID == 1) isSphere = true;
    else if (entryID == 2) isSmall = false;
    glutPostRedisplay();
}

void MyMenuItem(int entryID) {
    if (entryID == 1) isSphere = true;
    else if (entryID == 2) isSphere = false;
    else if (entryID == 3) exit(0);
    glutPostRedisplay();
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize(1000, 1000);
    glutInitWindowPosition(0, 0);
    glutCreateWindow("OpenGL Practice");
    gluOrtho2D(-1.0, 1.0, -1.0, 1.0);
    gluLookAt(0, 0, 0, 0, 0, 0, 1.0);
    glutAddMenuEntry("Draw Sphere", 1);
    glutAddMenuEntry("Draw Torus", 2);
    glutAddMenuEntry("Change Size", 3);
    glutAttachMenu(GLUT_RIGHT_BUTTON);
    glutDisplayFunc(MyDisplay);
    glutCreateMenu(MyMenu);
    glutAddSubMenu("Sphere", MySubmenu);
    glutAddMenuEntry("Small One", 1);
    glutAddMenuEntry("Big One", 2);
    glutCreateMenu(MyMenuItem);
    glutAddMenuEntry("Draw Sphere", 1);
    glutAddMenuEntry("Draw Torus", 2);
    glutAddMenuEntry("Change Size", 3);
    glutAttachMenu(GLUT_RIGHT_BUTTON);
    glutMainLoop();
}
```



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

GLboolean isSphere = true;
GLboolean isSmall = true;

void MyDisplay() {
    glClear(GL_COLOR_BUFFER_BIT);
    gluOrtho2D(0, 1000, 0, 1000);
    if ((isSphere) && (!isSmall))
        gluSphere(isSmall ? 0.2 : 15, 15);
    else if ((isSphere) && (!isSmall))
        gluSphere(isSmall ? 0.4 : 15, 15);
    else if ((!isSphere) && (!isSmall))
        gluSphere(isSmall ? 0.4 : 40, 20);
    else
        gluSphere(isSmall ? 0.2 : 0.5, 40, 20);
    glutPostRedisplay();
}

void MyMenu(int entryID) {
    if (entryID == 1) isSphere = true;
    else if (entryID == 2) isSmall = false;
    glutPostRedisplay();
}

void MyMenuItem(int entryID) {
    if (entryID == 1) isSphere = true;
    else if (entryID == 2) isSphere = false;
    else if (entryID == 3) exit(0);
    glutPostRedisplay();
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize(1000, 1000);
    glutInitWindowPosition(0, 0);
    glutCreateWindow("OpenGL Practice");
    gluOrtho2D(-1.0, 1.0, -1.0, 1.0);
    gluLookAt(0, 0, 0, 0, 0, 0, 1.0);
    glutAddMenuEntry("Draw Sphere", 1);
    glutAddMenuEntry("Draw Torus", 2);
    glutAddMenuEntry("Change Size", 3);
    glutAttachMenu(GLUT_RIGHT_BUTTON);
    glutDisplayFunc(MyDisplay);
    glutCreateMenu(MyMenu);
    glutAddSubMenu("Sphere", MySubmenu);
    glutAddMenuEntry("Small One", 1);
    glutAddMenuEntry("Big One", 2);
    glutCreateMenu(MyMenuItem);
    glutAddMenuEntry("Draw Sphere", 1);
    glutAddMenuEntry("Draw Torus", 2);
    glutAddMenuEntry("Change Size", 3);
    glutAttachMenu(GLUT_RIGHT_BUTTON);
    glutMainLoop();
}
```



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

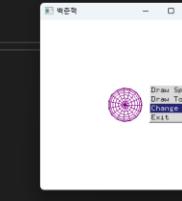
GLboolean isSphere = true;
GLboolean isSmall = true;

void MyDisplay() {
    glClear(GL_COLOR_BUFFER_BIT);
    gluOrtho2D(0, 1000, 0, 1000);
    if ((isSphere) && (!isSmall))
        gluSphere(isSmall ? 0.5 : 0.5);
    else if ((isSphere) && (!isSmall))
        gluSphere(isSmall ? 0.2 : 15, 15);
    else if ((!isSphere) && (!isSmall))
        gluSphere(isSmall ? 0.4 : 40, 20);
    else
        gluSphere(isSmall ? 0.2 : 0.5, 40, 20);
    glutPostRedisplay();
}

void MyMenu(int entryID) {
    if (entryID == 1) isSphere = true;
    else if (entryID == 2) isSmall = false;
    glutPostRedisplay();
}

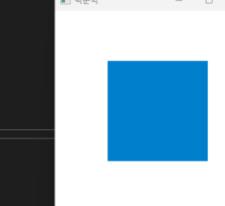
void MyMenuItem(int entryID) {
    if (entryID == 1) isSphere = true;
    else if (entryID == 2) isSphere = false;
    else if (entryID == 3) exit(0);
    glutPostRedisplay();
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize(1000, 1000);
    glutInitWindowPosition(0, 0);
    glutCreateWindow("OpenGL Practice");
    gluOrtho2D(-1.0, 1.0, -1.0, 1.0);
    gluLookAt(0, 0, 0, 0, 0, 0, 1.0);
    glutAddMenuEntry("Draw Sphere", 1);
    glutAddMenuEntry("Draw Torus", 2);
    glutAddMenuEntry("Change Size", 3);
    glutAttachMenu(GLUT_RIGHT_BUTTON);
    glutDisplayFunc(MyDisplay);
    glutCreateMenu(MyMenu);
    glutAddSubMenu("Sphere", MySubmenu);
    glutAddMenuEntry("Small One", 1);
    glutAddMenuEntry("Big One", 2);
    glutCreateMenu(MyMenuItem);
    glutAddMenuEntry("Draw Sphere", 1);
    glutAddMenuEntry("Draw Torus", 2);
    glutAddMenuEntry("Change Size", 3);
    glutAttachMenu(GLUT_RIGHT_BUTTON);
    glutMainLoop();
}
```





```
CG_Practice02.cpp CG_Practice05.cpp CG_Practice04.cpp CG_Practice03.cpp CG_Practice02.cpp CG_Practice01.cpp
CG_Practice02
1 #include <GL/glut.h>
2 #include <GL/glu.h>
3 #include <GL/glx.h>
4
5 GLfloat Delta = 0.0;
6
7 void MyDisplay() {
8     glClear(GL_COLOR_BUFFER_BIT);
9     glLoadIdentity();
10    glColor3f(0.0, 0.5, 0.0);
11    glVertext(-1.0 - Delta, -0.5, 0.0);
12    glVertext(-1.0 + Delta, -0.5, 0.0);
13    glVertext(0.0 + Delta, 0.5, 0.0);
14    glVertext(-1.0 - Delta, 0.5, 0.0);
15    glEnd();
16    glutSwapBuffers();
17}
18
19 void MyIdle() {
20    Delta = Delta + 0.001;
21    glutPostRedisplay();
22}
23
24 int main(int argc, char** argv) {
25    glutInit(&argc, argv);
26    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);
27    glutInitWindowSize(300, 300);
28    glutCreateWindow("비운적");
29    glutInitPosition(0, 0);
30    glutCreateRenderer();
31    glutCreateModel();
32    glutCreateProjection();
33    glutIdentify();
34    glutOrtho(-1.0, 1.0, -1.0, 1.0, -1.0);
35    glutDisplayFunc(display);
36    glutIdleFunc(MyIdle);
37    glutMainLoop();
38    return 0;
39}
```



```
D:\Computer_Graphics\CG_Practice02\CG_Practice02
CG_Practice02.cpp CG_Practice05.cpp CG_Practice04.cpp CG_Practice03.cpp CG_Practice02.cpp CG_Practice01.cpp
CG_Practice02
1 #include <GL/glut.h>
2 #include <GL/glu.h>
3 #include <GL/glx.h>
4
5 GLfloat Delta = 0.0;
6
7 void MyDisplay() {
8     glClear(GL_COLOR_BUFFER_BIT);
9     glLoadIdentity();
10    glColor3f(0.0, 0.5, 0.0);
11    glVertext(-1.0 - Delta, -0.5, 0.0);
12    glVertext(-1.0 + Delta, -0.5, 0.0);
13    glVertext(0.0 + Delta, 0.5, 0.0);
14    glVertext(-1.0 - Delta, 0.5, 0.0);
15    glEnd();
16    glutSwapBuffers();
17}
18
19 void MyIdle() {
20    Delta = Delta + 0.001;
21    glutPostRedisplay();
22}
23
24 int main(int argc, char** argv) {
25    glutInit(&argc, argv);
26    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);
27    glutInitWindowSize(300, 300);
28    glutCreateWindow("비운적");
29    glutInitPosition(0, 0);
30    glutCreateRenderer();
31    glutCreateModel();
32    glutCreateProjection();
33    glutIdentify();
34    glutOrtho(-1.0, 1.0, -1.0, 1.0, -1.0);
35    glutDisplayFunc(display);
36    glutIdleFunc(MyIdle);
37    glutMainLoop();
38    return 0;
39}
```



```
D:\Computer_Graphics\CG_Practice02\CG_Practice02
CG_Practice02.cpp CG_Practice05.cpp CG_Practice04.cpp CG_Practice03.cpp CG_Practice02.cpp CG_Practice01.cpp
CG_Practice02
1 #include <GL/glut.h>
2 #include <GL/glu.h>
3 #include <GL/glx.h>
4
5 GLfloat Delta = 0.0;
6
7 void MyDisplay() {
8     glClear(GL_COLOR_BUFFER_BIT);
9     glLoadIdentity();
10    glColor3f(0.0, 0.5, 0.0);
11    glVertext(-1.0 - Delta, -0.5, 0.0);
12    glVertext(-1.0 + Delta, -0.5, 0.0);
13    glVertext(0.0 + Delta, 0.5, 0.0);
14    glVertext(-1.0 - Delta, 0.5, 0.0);
15    glEnd();
16    glutSwapBuffers();
17}
18
19 void MyIdle() {
20    Delta = Delta + 0.001;
21    glutPostRedisplay();
22}
23
24 int main(int argc, char** argv) {
25    glutInit(&argc, argv);
26    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);
27    glutInitWindowSize(300, 300);
28    glutCreateWindow("비운적");
29    glutInitPosition(0, 0);
30    glutCreateRenderer();
31    glutCreateModel();
32    glutCreateProjection();
33    glutIdentify();
34    glutOrtho(-1.0, 1.0, -1.0, 1.0, -1.0);
35    glutDisplayFunc(display);
36    glutIdleFunc(MyIdle);
37    glutMainLoop();
38    return 0;
39}
```

```
#include <iostream>
#include <GL/glut.h>
#include <GL/glu.h>
#include <GL/glx.h>

GLfloat Delta = 0.0;

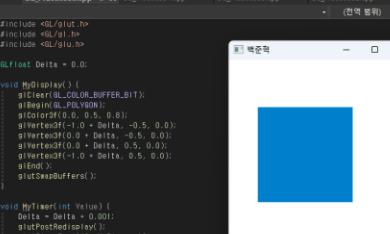
void MyDisplay() {
    glClear(GL_COLOR_BUFFER_BIT);
    glPushMatrix();
    glTranslatef(0.0, 0.0, 8.0);
    glScalef(Delta, -0.5, Delta);
    glTranslatef(0.0 + Delta, -0.5, 0.0);
    glTranslatef(0.0 + Delta, 0.5, 0.0);
    glTranslatef(-1.0 - Delta, 0.5, 0.0);
    glPopMatrix();
    glutSwapBuffers();
}

void MyTimer(int Value) {
    Delta += 0.01;
    glutPostRedisplay();
    glutTimerFunc(40, MyTimer, 1);
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);
    glutInitWindowSize(800, 300);
    glutInitWindowPosition(0, 0);

    glutCreateWindow("OpenGL Practice 06");
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glMatrixMode(GL_PROJECTION);
    gluPerspective(45.0, 1.0, -1.0, 1.0, -1.0);
    glutDisplayFunc(MyDisplay);
    glutIdleFunc(MyTimer);
    glutMainLoop();
    return 0;
}
```





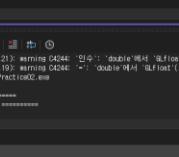
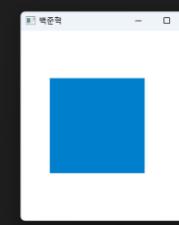
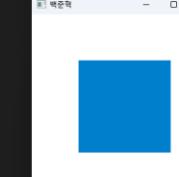
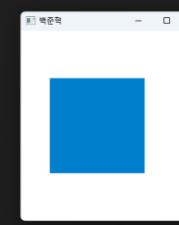
```
#include <GL/glut.h>
#include <GL/glu.h>
#include <math.h>

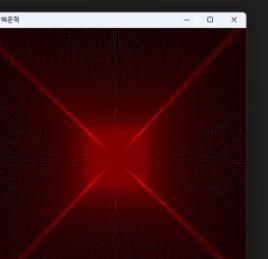
GLfloat Delta = 0.0;

void MyDisplay() {
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_POLYGON);
    glVertex3f(0.0 + Delta, -0.5, 0.0);
    glVertex3f(0.0 + Delta, -0.5, 0.0);
    glVertex3f(0.0 + Delta, 0.5, 0.0);
    glVertex3f(-0.0 + Delta, 0.5, 0.0);
    glEnd();
    glutSwapBuffers();
}

void MyTimer(int Value) {
    Delta += Delta + 0.001;
    glutPostRedisplay();
    glutTimerFunc(40, MyTimer, 1);
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GL_RGB | GL_DEPTH_DOUBLE);
    glutInitWindowSize(300, 300);
    glutCreateWindow("My OpenGL");
    glutCreateMLayer("My OpenGL");
    glClearColor(1.0, 1.0, 1.0, 1.0);
    gluOrtho2D(-1.0, 1.0, -1.0, 1.0);
    glutDisplayFunc(MyDisplay);
    glutIdleFunc(MyDisplay);
    glutTimerFunc(40, MyTimer, 1);
    glutMainLoop();
    return 0;
}
```





```
#include <GL/glut.h>
#include "PALETTE16x16.h"

void display();
void keyboard(unsigned char key, int x, int y);
void mouse(int button, int state, int x, int y);
void motion(int x, int y);

int main()
{
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(500, 500);
    glutCreateWindow("백준학");
    glutDisplayFunc(display);
    glutKeyboardFunc(keyboard);
    glutMouseFunc(mouse);
    glutMotionFunc(motion);
    glutMainLoop();
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    glTranslatef(0.0f, 0.0f, -1.0f);
    glScalef(0.1f, 0.1f, 0.1f);
    glRotatef(45.0f, 0.0f, 0.0f, 1.0f);

    int index = 0;
    for (int i = 0; i < 16; i++)
        for (int j = 0; j < 16; j++)
            if (PALETTE16x16[i][j] == 0)
                glColor3f(1.0f, 0.0f, 0.0f);
            else
                glColor3f((float)PALETTE16x16[i][j] / 256.0f,
                        (float)PALETTE16x16[i][j] / 256.0f,
                        (float)PALETTE16x16[i][j] / 256.0f);

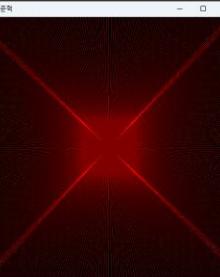
            glBegin(GL_TRIANGLES);
            glVertex3f(-1.0f, -1.0f, 0.0f);
            glVertex3f(1.0f, -1.0f, 0.0f);
            glVertex3f(0.0f, 1.0f, 0.0f);
            glEnd();

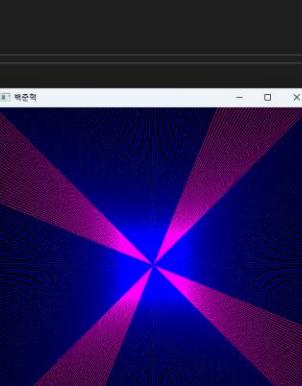
    glutSwapBuffers();
}

void keyboard(unsigned char key, int x, int y)
{
    if (key == 'q' || key == 'Q')
        exit(0);
}

void mouse(int button, int state, int x, int y)
{
    if (button == GLUT_LEFT_BUTTON)
        if (state == GLUT_DOWN)
            glutPostRedisplay();
}

void motion(int x, int y)
{
    if (y > 100)
        glutPostRedisplay();
}
```





```
#include <GL/glut.h>
#include <iostream.h>
unsigned char PALETTE[16][3] = {
    { 0, 0, 0 }, // BLACK
    { 128, 128, 128 }, // GRAY
    { 0, 0, 255 }, // CYAN
    { 255, 0, 255 }, // PURPLE
    { 255, 255, 0 }, // MAGENTA
    { 255, 0, 0 }, // RED
    { 0, 255, 0 }, // GREEN
    { 0, 255, 255 }, // LIGHT BLUE
    { 255, 255, 255 }, // WHITE
    { 128, 128, 128 }, // LIGHT GRAY
    { 0, 128, 0 }, // DARK TEAL
    { 0, 128, 128 }, // DARK CYAN
    { 128, 0, 128 }, // DARK PURPLE
    { 0, 0, 128 }, // DARK BLUE
    { 255, 255, 0 }, // YELLOW
    { 128, 128, 128 }, // DARK GRAY
    { 128, 128, 0 }, // DARK YELLOW
    { 0, 128, 0 }, // DARK GREEN
    { 0, 0, 128 }, // DARK RED
    { 0, 0, 0 } // BLACK
};

float Delta = 0.0;
GLuint Index = 0;
GLfloat Red = 0.0;
GLfloat Green = 0.0;
GLfloat Blue = 0.0;

void MyDisplay() {
    glClear(GL_COLOR_BUFFER_BIT);
    glClearColor(Red, Green, Blue, 1.0);
    glPushMatrix();
    glTranslatef(0.0, 0.0, -10.0);
    glVertices3f(-1.0 + Delta, 1.0, 0.0);
    glVertices3f(-1.0 - Delta, 1.0, 0.0);
    glVertices3f(-1.0 - Delta, -1.0, 0.0);
    glVertices3f(-1.0 + Delta, -1.0, 0.0);
    glVertices3f(1.0, 0.0, 0.0);
    glEnd();
    glutSwapBuffers();
}

void MyInerfntValue() {
    if (Delta < 2.0f) Delta += 0.01;
    else {
        Delta -= 0.01;
        if (++Index == 15) Index = 0;
        glColor3ub(0, 0, 0);
        sIClear(GL_COLOR_BUFFER_BIT);
    }
    glutPostRedisplay();
    glutTimerFunc(10, MyTimer, 1);
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);
    glutCreateWindow("OpenGL Practice02");
    glutDisplayFunc(MyDisplay);
    glutIdleFunc(MyInerfntValue);
    glutTimerFunc(10, MyTimer, 1);
    glutMainLoop();
}
```

