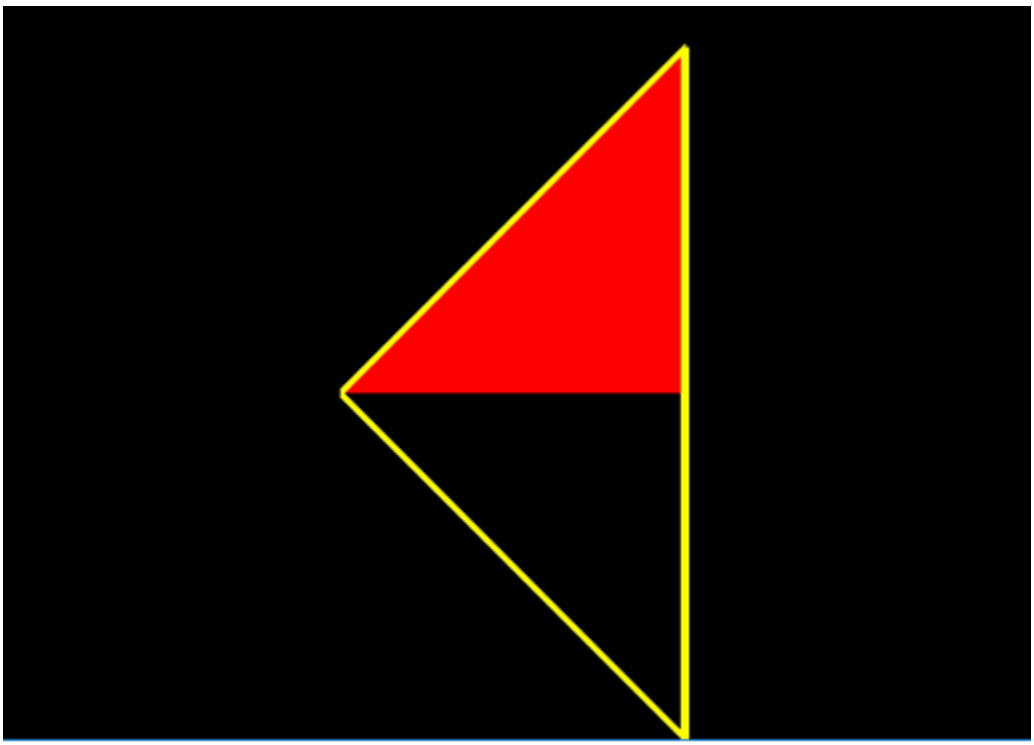
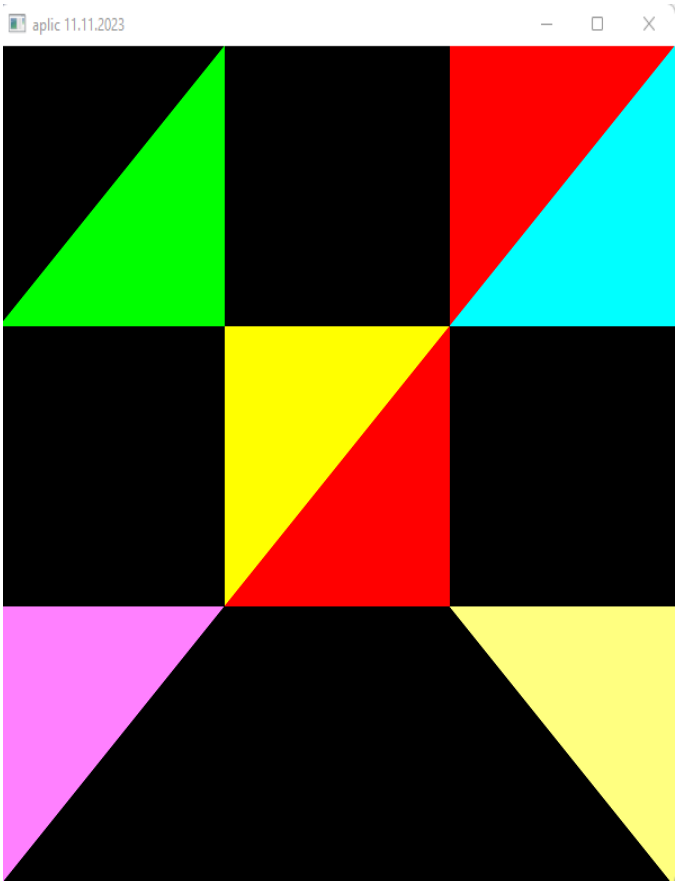
Aplicatie propusa



Aplicatia 1

```
#include <gl/freeglut.h>
void init()//functia initiere
{
	  glClearColor (1.0, 1.0, 1.0, 0.0);//stabileste culoarea de sters
	//  glShadeModel (GL_FLAT);
}
void display()
{
	glClear(GL_COLOR_BUFFER_BIT);//sterge urmele de desene din fereastra curenta
	glBegin(GL_POLYGON);
	glColor3f(0.2, 0.0, 0.0);
	glVertex2f(200.0, 200.0);
	glVertex2f(400.0, 200.0);
	glVertex2f(400.0, 400.0);
	glEnd();//sfisit desenare
	glFlush();//executare functie
	glColor3f(1.0, 1.0, 0.0);//culoarea de desenare
	glBegin(GL_POLYGON);//initializare desen poligon
	glVertex2f(200.0, 400.0);
```

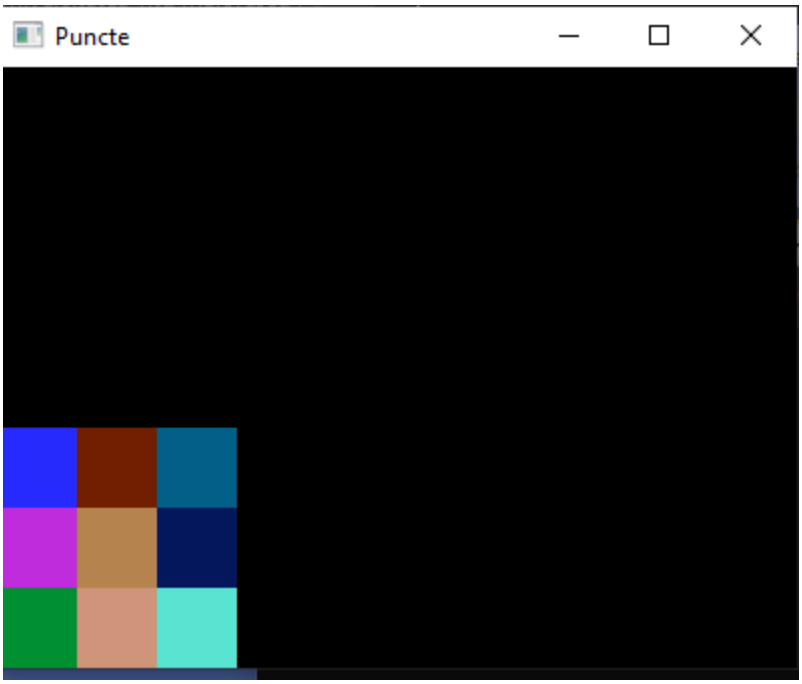
```
        glVertex2f(400.0, 400.0);
        glVertex2f(200.0, 200.0);
        glEnd();//sfisit desenare
        glFlush();//executare functie

}
void reshape(int w, int h)//functia redesenare
{
        glViewport(0, 0, (GLsizei)w, (GLsizei)h);//stabilirea viewportului la dimensiunea
ferestrei
        glMatrixMode(GL_PROJECTION);//specificare matrice modificabila la valoare
argumentului de modificare
        glLoadIdentity();//initializarea sistemului de coordonate
        gluOrtho2D(0.0, (GLdouble)w, 0.0, (GLdouble)h);//stabileste volumul de vedere folosind
o proiectie ortografica
}
int main(int argc, char** argv) //creare fereastra
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);//se specifica modelul de culoare
al ferestrei: un singur buffer si culoare RGB
        glutInitWindowSize(600, 600);//initiaza dimensiunea ferestrei principale 600x600 pixeli
        glutInitWindowPosition(200, 10);//initiaza in fereastra principala fereastra de afisare
        glutCreateWindow("aplic 11.11.2023");
        init();
        glutDisplayFunc(display);//se reimprospateza continutul ferestrei
        glutReshapeFunc(reshape);//functia redesenare
        glutMainLoop();//bucla de procesare a evenimentelor
        return 0;
}
```


aplic 11.11.2023

Aplicatia 2

Aplicatia 1

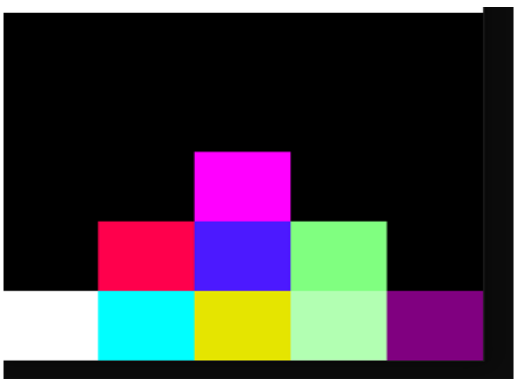
```
for (int dist = 0, i = 1; i <= 3; i++)
{
dist += 40;
glVertex2i(40*i+dist, 40);
}
```




```
#include <iostream>
#include <gl/freeglut.h>
#include<math.h>
int width = 400;
int height = 400;
int psize = 40;
int distx = 0;
int disty = 0;
void display()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glPointSize(psize);
	glBegin(GL_POINTS);
	disty = 20;
	for (int k = 0; k < 3; k++)
	{
		distx = 20;
		for (int j = 0; j < 3; j++)
		{
			double r = ((double)rand() / (RAND_MAX));
			double g = ((double)rand() / (RAND_MAX));
			double b = ((double)rand() / (RAND_MAX));
			glColor3d(r, g, b);
```

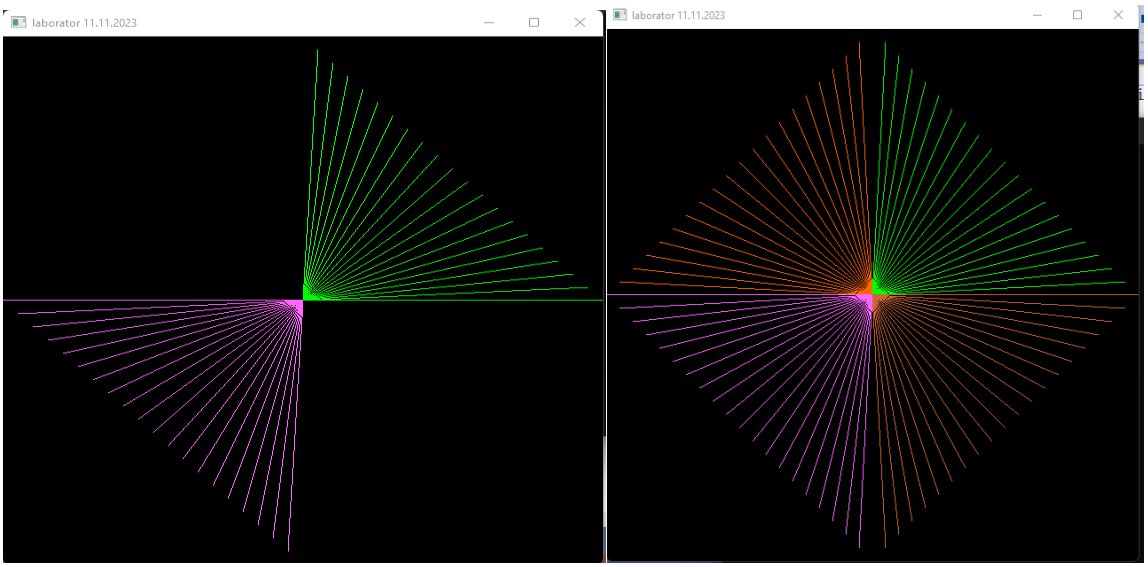
```
                        glVertex2i(40 * j + distx, 40 * k + disty);

                }
                disty += 0;
        }
        glEnd();
        glFlush();

}


void reshape(int w, int h)
{
        glViewport(0, 0, (GLsizei)w, (GLsizei)h); //stabilirea viewportului la dimensiunea
ferestrei
        glMatrixMode(GL_PROJECTION); //specificare matrice modificabila la valoare
argumentului de modificare
        glLoadIdentity(); //initializarea sistemului de coordonate
        gluOrtho2D(0.0, (GLdouble)w, 0.0, (GLdouble)h); //stabileste volumul de vedere
folosind o proiectie ortografica
}//end reshape()

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

        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(400, 300);
        glutCreateWindow("Puncte cu for");
        glutDisplayFunc(display);
        glutReshapeFunc(reshape);
        glutMainLoop();
        return 0;
}
```

Sa se modifice aplicatia pentru a obtine imaginea



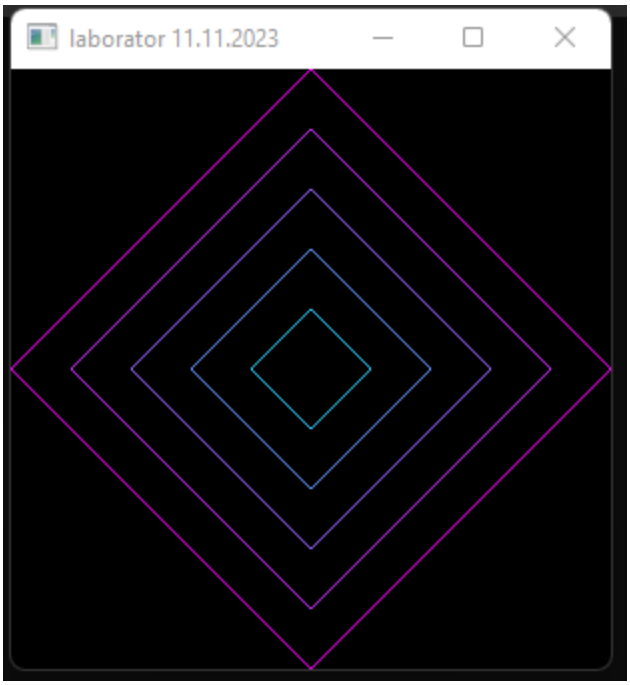
Aplicatie x,y,x

```
#include <iostream>
#include <gl/freeglut.h>
void Display(void)
{
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(0, 1, 0);
	glBegin(GL_LINES);
	// Cadran 1
	for (int i = 0; i < 20; i++)
	{
		glVertex3f(0, 0, 0);
		glVertex3f(1 - i / 20.0, i / 20.0, 0);
	}  // Cadran 3
	glColor3f(1, 0.4, 1);
	for (int i = 0; i < 20; i++)
	{
		glVertex3f(0, 0, 0);
		glVertex3f(-1 + i / 20.0, -i / 20.0, 0);
	}

	glEnd();    glFlush();
}
int main(int argc, char** argv) {
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(600, 600);
	//se specifica modelul de culoare al ferestrei: un singur buffer si culoare RGB
	glutCreateWindow("laborator 11.11.2023");

	glutDisplayFunc(Display);
	glutMainLoop();
	return 0;
}
```



Aplicatie x,y,z

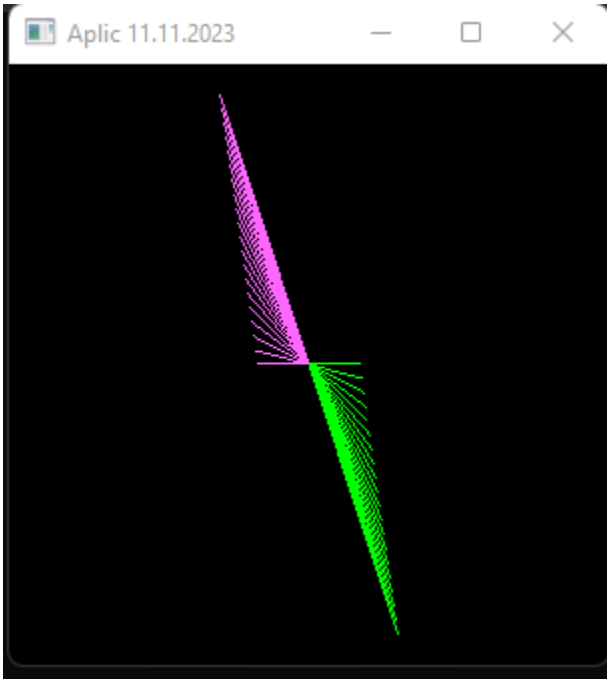
```
void Display(void)
{
    int pas = 5;
    for (double i = 0; i <= pas; i++) {
        glBegin(GL_LINE_LOOP);
        glColor3f(1 - i / pas, i / pas, 1);
        glVertex3f(1 - i / pas, 0, 0);
        glVertex3f(0, 1 - i / pas, 0);
        glVertex3f(-(1 - i / pas), 0, 0);
        glVertex3f(0, -(1 - i / pas), 0);
        glEnd();
    }
    glFlush();

}
```


laborator 11.11.2023

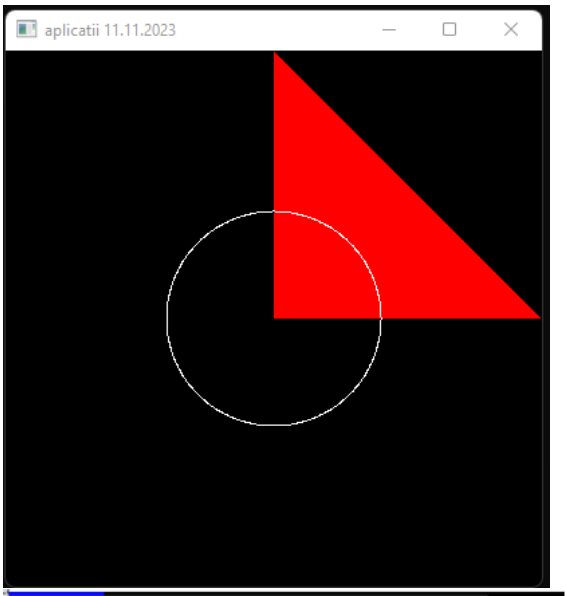
Eveniment tastatura – mouse

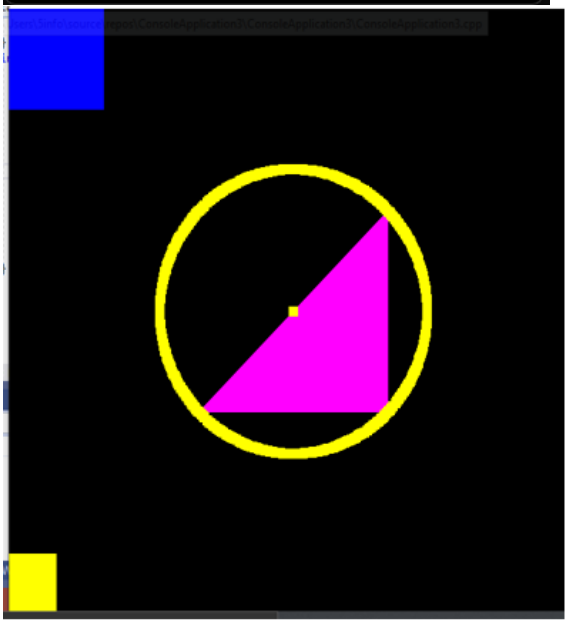
```
#include <iostream>
#include <gl/freeglut.h>
void roteste_Y(int p_grade)
{
	glRotatef(p_grade, .0, 1.0, .0);
	glutPostRedisplay();
}
void roteste_X(int p_grade)
{
	glRotatef(p_grade, 1., 0., .0);
	glutPostRedisplay();
}
void OnKeyPress(unsigned char key, int x, int y)
{
	if (key == 27)
		exit(0);
	switch (key)
	{
	case 'q':
	case 'Q':

		roteste_Y(3);
		break;
	case 'w':
	case 'W':
		roteste_Y(-3);
		break;

	case 'a':
	case 'A':

		roteste_X(3);
		break;
	case 's':
	case 'S':
		roteste_X(-3);
		break;
	}
}
void OnMouseClick(int button, int state, int x, int y)
{
	if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
	{
		roteste_Y(20);
	}
	if (button == GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
	{
		roteste_Y(-20);
	}
}

void Display(void)
{
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(0, 1, 0);
	glBegin(GL_LINES);
```

```
        // Cadran 1
        for (int i = 0; i < 20; i++)
        {
                glVertex3f(0, 0, 0);
                glVertex3f(1 - i / 20.0, i / 20.0, 0);
        }  // Cadran 3
        glColor3f(1, 0.4, 1);
        for (int i = 0; i < 20; i++)
        {
                glVertex3f(0, 0, 0);
                glVertex3f(-1 + i / 20.0, -i / 20.0, 0);
        }

        glEnd();    glFlush();
}

int main(int argc, char** argv) {
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB;
        glutCreateWindow("Aplic 11.11.2023");
        glutKeyboardFunc(OnKeyPress);
        glutMouseFunc(OnMouseClick);
        glutDisplayFunc(Display);
        glutMainLoop();
        return 0;
}
```



Meniuri





```
#include <iostream>
#include <gl/freeglut.h>

void init()
{
	glClearColor(0.0, 0.0, 0.0, 0.0);
	//glPointSize(50.0);
	glShadeModel(GL_FLAT);
}
```

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

	glBegin(GL_TRIANGLES);
	glColor3f(1.0, 0.0, 0.0);
	glVertex2i(1, 0);
	glVertex2i(0, 0);
	glVertex2i(0, 1);
	glEnd();

	glPointSize(1.0);
	glColor3f(1, 1, 1);
	glBegin(GL_POINTS);
	for (int i = 0; i < 1000; ++i)
	{
		glVertex3f(cos(2 * 3.14159 * i / 1000.0) * 0.4, sin(2 * 3.14159 * i /
1000.0) * 0.4, 0);
	}
	glEnd(); glFlush();

}
int meniu_1, meniu_2, meniu_main;

void meniu_principal(int key)
{
	if (key == 0)
	{
		exit(0);
	}
}

void callback_1(int key)
{
	switch (key)
	{
	case 0:
		printf("Cerc 1\n");
		break;
	case 1:
		printf("Cerc 2\n");
		break;
	}
}

void callback_2(int key)
{
	switch (key)
	{
	case 0:
		printf("Ati selectat dreptunghi 1\n");
		break;
	case 1:
		printf("Ati selectat dreptunghi 2\n");
		break;
	}
}
```

```
GLint x = 10;
GLint y = 20;
GLint WindowWidth = 400;
GLint WindowHight = 400;

void mouseHandler(int button, int state, int mouse_x, int mouse_y)
{
	if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
	{
		x = mouse_x;
		y = WindowHight - mouse_y;
		glColor3f(1, 0, 0);
		glBegin(GL_POINTS);
		glVertex2i(x, y);
		printf("x=%d , y=%d \n", x, y);
		glEnd();
		glFlush();
		glClear(GL_COLOR_BUFFER_BIT);
	}
}
int main(int argc, char** argv)
{

	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(400, 400);
	glutInitWindowPosition(400, 100);
	glutCreateWindow("aplicatii 11.11.2023");
	init();
	glutMouseFunc(mouseHandler);

	glutDisplayFunc(display);
	meniu_1 = glutCreateMenu(callback_1);
	glutAddMenuEntry("cerc1", 0);
	glutAddMenuEntry("cerc2", 1);
	meniu_2 = glutCreateMenu(callback_2);
	glutAddMenuEntry("dreptunghi1", 0);
	glutAddMenuEntry("dreptunghi2", 1);
	meniu_main = glutCreateMenu(meniu_principal);
	glutAddSubMenu("cerc", meniu_1);
	glutAddSubMenu("patrat", meniu_2);
	glutAddMenuEntry("Exit", 0);
	glutAttachMenu(GLUT_RIGHT_BUTTON);
	glutMainLoop();
	return 0;
}
```