بنیادهای ترسیم اشکال با OpenGL

مقدمه

هر برنامهٔ گرافیکی با مقداری مقدار دهی اوّلیه برای برقراری مد نمایش مورد علاقه، ایجاد دستگاه مختصات برای مشخص کردن نقاط، خطوط و ... شروع می شود. دستگاههای نمایش دارای روشهای نمایش متفاوتی می توانند باشند.

استفاده از تمامی صفحه

استفاده از پنجره

در حالت اخیر مقداردهی اولیه با ایجاد یک پنجره جدید شروع می شود. معمولاً دستورهای ابتدائی برای رسم نقطه و خط وجود دارند. برخی از سیستمها از دستورهائی مثل moveto و lineto استفاده می کنند. می توان از این دستورهای ابتدائی استفاده کرد و دستورهای پیچیده تری را بوجود آورد. می توان کتابخانه ای را روالهای مهم ایجاد و سپس از آنها برای رسم اشکال متفاوت استفاده نمود. یک مشکل مهم آن است که هر دستگاه نمایش گرافیکی از دستورهای بنیادی متفاوت و ابزارهای متفاوتی برای ایجاد رسمهای ابتدائی استفاده می کند. این باعث عدم قابلیت انتقال پذیری برنامه ها می شود.

OpenGL قابلیت نوشتن برنامه های مستقل از دستگاه را بوجود می آورد. کافی است که ابتدا کتابخانه های لازم بر روی دستگاه مورد نظر نصب شوند.

برنامه نویسی مبتنی بر پنجره

امروزه اغلب برنامه ها مبتنی بر پنجره نوشته می شوند.

برنامه نویسی حادثه ران[1]

یکی از خصوصیات مهم برنامه نویسی پنجره- مبنا. برنامه به حادثه های مختلفی همانند کلیک کردن موش، تغییر اندازهٔ پنجره، و فشار دادن کلید پاسخ می دهد.

برنامه بطور خودکار یک صف حادثه را مدیریت می کند. حادثه ها به ترتیب اولین ورود – اولین سرویس اجرا می شود. برنامه نویس برنامه اش را بصورت یک سری توابع پاسخ[2] سازماندهی می کند. هر برنامهٔ پاسخ به یک حادثه پاسخ می دهد. هنگامی که یک حادثه از صف برداشته می شود برنامهٔ پاسخ متناظر با آن فراخوانی می شود.

ساختار برنامه:

[1] event-driven
[2] callback

کاری نکن تا حادثه ای رخ دهد

سپس کار مشخص شده را انجام بده

نیاز به یک حلقهٔ حادثه[3]. روش وابسته کردن یک تابع پاسخ به هر حادثه وابسته به سیستم است. OpenGL دارای یک Utility toolkit است که سعی در ساده نمودن این کار دارد. یکی از این جعبه ابزارها کتابخانهٔ GLUT (the GL Utility Toolkit) می باشد.

بطور مثال:

```
glutMouseFunc(myMouse);
```

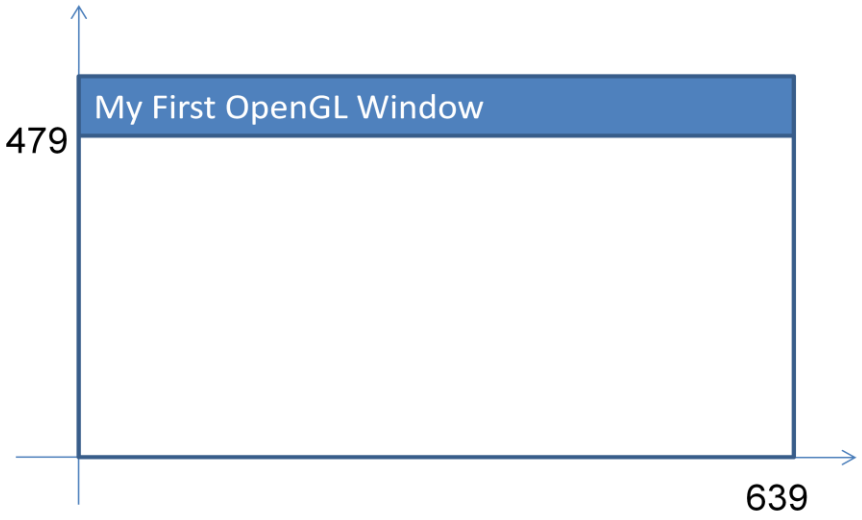
باعث می شود که روال myMouse هنگامی که یک حادثهٔ موش رخ می دهد فراخوانی شود. نام myMouse توسط برنامه نویس انتخاب شده است. بدین کار ثبت توابع پاسخ گفته می شود. اگر در برنامه ای از موش استفاده نمی شود احتیاجی به ثبت تابع پاسخ آن نیست. شکل بعد اسکلت یک برنامهٔ حادثه ران را نشان می دهد.

```
void main(){
        مقداردهی اولیهٔ چیزها
        ایجاد یک پنجره
        glutDisplayFunc(myDisplay);          //ثبت تابع بازرسم
        glutReshapeFunc(myReshape);      //  ثبت تابع بازشکل
        glutMouseFunc(myMouse);          //ثبت تابع موش
        glutKeyboardFunc(myKeyboard);  //ثبت تابع حادثهٔ صفحهٔ کلید
        شاید مقداردهی چیزهای دیگر
        glutMainLoop();                          //وارد شدن به حلقهٔ اصلی بی پایان
}
```

توابع پاسخ

[3] event loop

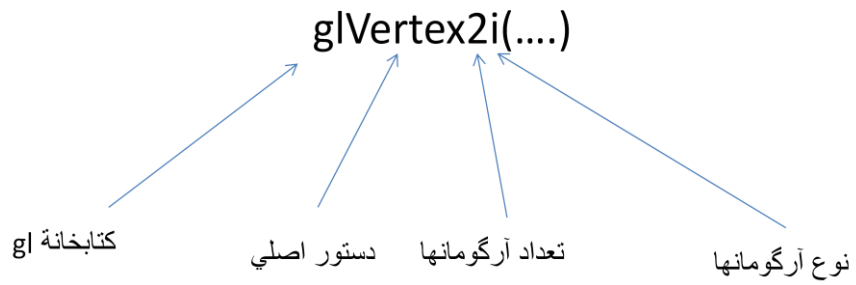
```
void main(int argc, char **argv)
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(640,480);
        glutInitWindowPosition(100,150);
        glutCreateWindow("My First OpenGL Window");

        // ثبت توابع پاسخ
        glutDisplayFunc(myDisplay);
        myInit();
        glutMainLoop();
}
```



شکل ۱

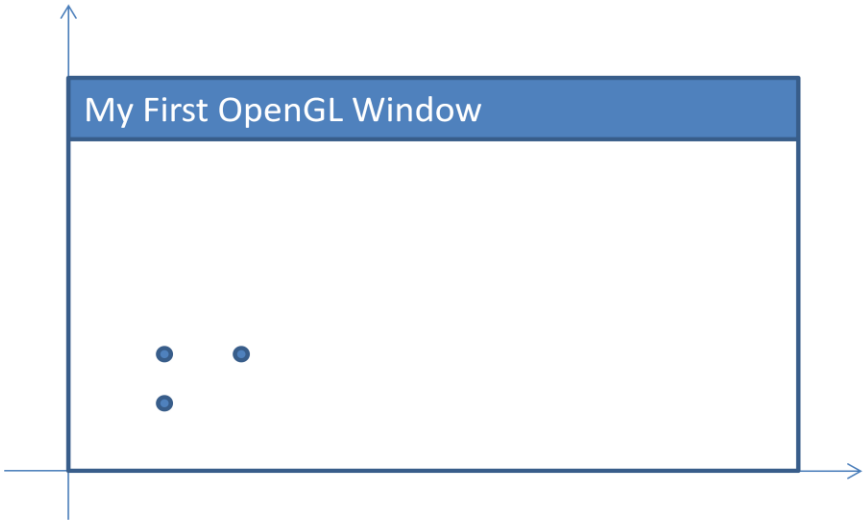
رسم اجزاء ابتدائی گرافیکی

```
glBegin(GL_POINTS);
        glVertex2i(100, 50);
        glVertex2i(100, 130);
        glVertex2i(150, 130);
glEnd();
```



شکل ۲

gl مشخص کنندهٔ روالی از کتابخانهٔ OpenGl است (در مقابل glut برای GL utility toolkit)



شکل ۳

نوعهای داده در OpenGL

جدول ۱

پسوند	نوع داده	نوع C یا C++	نام OpenGL

b	صحیح ۸ بیتی	signed char	GLbyte
s	صحیح ۱۶ بیتی	short	GLshort
i	صحیح ۳۲ بیتی	int یا long	GLint, GLsizei
f	اعشاری ۳۲ بیتی	float	GLfloat, GLclampf
d	اعشاری ۶۴ بیتی	double	GLdouble, GLclampd
ub	صحیح ۸ بیتی بی علامت	unsigned char	Glubyte, GLboolean
us	صحیح ۱۶ بیتی بی علامت	unsigned short	GLushort
ui	صحیح ۳۲ بیتی بی علامت	unsigned int یا unsigned long	GLuint, Glenum, GLbitfield

مراقبت

```
void drawDot(int x, int y)
{
	glBegin(GL_POINTS);
		glVertex2i(x, y);
	glEnd();
}
```

بهتر است بصورت زیر نوشته شود:

```
void drawDot(GLint x, GLint y)
{
	glBegin(GL_POINTS);
		glVertex2i(x, y);
	glEnd();
}
```

حالت OpenGL

OpenGL از بسیاری متغیرهای حالت نگهداری می کند همانند: رنگ فعلی ترسیم، رنگ زمینهٔ فعلی، اندازهٔ فعلی نقطه، .... مقدار یک متغیر حالت تا تغییر داده نشده ثابت باقی می ماند.

glPointSize(3.0) اندازهٔ نقطه

glColor3f(red, green, blue) رنگ ترسیم که مقادیر پارامترها از ۰ تا ۱ تغییر می کند.

```
glColor3f(1.0, 0.0, 0.0);        //رنگ قرمز
glColor3f(0.0, 1.0, 0.0);        //رنگ سبز
glColor3f(0.0, 0.0, 1.0);        //رنگ آبی
glColor3f(0.0, 0.0, 0.0);        //رنگ سیاه
glColor3f(1.0, 1.0, 1.0);        //رنگ سفید
glColor3f(1.0, 1.0, 0.0);        //رنگ زرد
glColor3f(0.7, 0.7, 0.7);        //رنگ خاکستری روشن
glColor3f(0.2, 0.2, 0.2);        //رنگ خاکستری تیره
```

glClearColor(red, green, blue, alpha) رنگ زمینه

alpha درجهٔ شفافیت را نشان می دهد.

مشخص کردن دستگاه مختصات

```
void myInit(void)
 {
    glClearColor(1.0, 1.0, 1.0, 0.0);        // set the bg color to a bright white
    glColor3f(0.0f, 0.0f, 0.0f);              // set the drawing color to black
    glPointSize(4.0);                  //set the point size to 4 by 4 pixels
```

```
    glMatrixMode(GL_PROJECTION);// set up appropriate matrices- to be explained
    glLoadIdentity();// to be explained
    gluOrtho2D(0.0, 640.0, 0.0, 480.0);// to be explained
}
```

وصل کردن همه به یکدیگر

```
#include <windows.h>   // use as needed for your system
#include <gl/Gl.h>
#include <gl/glu.h>
#include <gl/glut.h>

//<<<<<<<<<<<<<<<<<<<<<<< myInit >>>>>>>>>>>>>>>>>>>>
void myInit(void)
{
   glClearColor(1.0, 1.0, 1.0, 0.0);      // set the bg color to a bright white
   glColor3f(0.0f, 0.0f, 0.0f);           // set the drawing color to black
   glPointSize(4.0);                      //set the point size to 4 by 4 pixels
   glMatrixMode(GL_PROJECTION);// set up appropriate matrices- to be explained
   glLoadIdentity();                      // to be explained
   gluOrtho2D(0.0, 640.0, 0.0, 480.0);    // to be explained
}

//<<<<<<<<<<<<<<<<<<<<<<< myDisplay >>>>>>>>>>>>>>>>>>
// the redraw function
void myDisplay(void)
{
      glClear(GL_COLOR_BUFFER_BIT);       // clear the screen
      glBegin(GL_POINTS);
            glVertex2i(100, 50);          // draw some points (don't know how many)
            glVertex2i(100, 130);
            glVertex2i(150, 130);
      glEnd();
      glFlush();                          // send all output to display
}
//<<<<<<<<<<<<<<<<<<<<<<< main >>>>>>>>>>>>>>>>>>>>>>>
void main(int argc, char **argv)
{
      glutInit(&argc, argv);              // initialize the toolkit
      glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB); // set the display mode
      glutInitWindowSize(640,480);        // set the window size
      glutInitWindowPosition(100, 150); // set the window position on the screen
      glutCreateWindow("In the Name of Allah"); // open the window with its title
      glutDisplayFunc(myDisplay);         // register the redraw function
      myInit();
      glutMainLoop();                     // go into a perpetual loop
}
```

شکل ٤

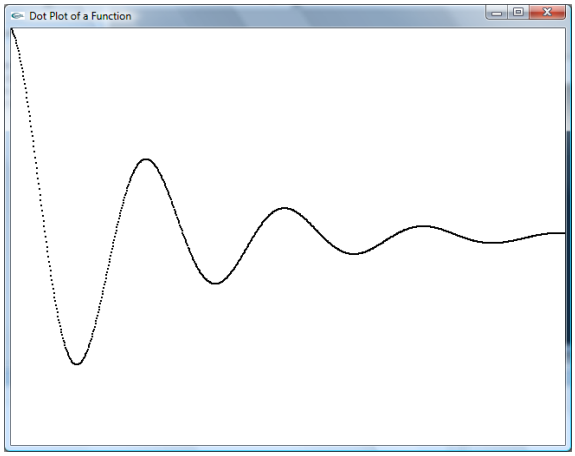
اگر تعداد نقاط کم باشد می توان کد آنها را با دست نوشت. اگر زیاد باشد باید یا در یک پرونده قرار داد یا از معادله استفاده نمود.

مثال: رسم معادلهٔ

$$f(x) = e^{|-x|}\cos(2\pi x)$$

```
#include <windows.h> // use proper includes for your system
#include <math.h>
#include <gl/Gl.h>
#include <gl/Glu.h>
#include <gl/Glut.h>
const int screenWidth = 640;     // width of the screen window in pixels
const int screenHeight = 480;    // height of the screen window in pixels
GLdouble A, B, C, D;  // values used for scaling and shifting
//<<<<<<<<<<<<<<<<<<<<<<< myInit >>>>>>>>>>>>>>>>>>>
 void myInit(void)
 {
    glClearColor(1.0,1.0,1.0,0.0);       // the background color is white
    glColor3f(0.0f, 0.0f, 0.0f);         // the drawing color is black
    glPointSize(2.0);                    // a 'dot' is 2 by 2 pixels
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, (GLdouble)screenWidth, 0.0, (GLdouble)screenHeight);
    A = screenWidth / 4.0; // sets the values used for scaling and shifting
    B = 0.0;
    C = D = screenHeight / 2.0;
}
//<<<<<<<<<<<<<<<<<<<<<<< myDisplay >>>>>>>>>>>>>>>>>
void myDisplay(void)
{
      glClear(GL_COLOR_BUFFER_BIT);     // clear the screen
      glBegin(GL_POINTS);
      for(GLdouble x = 0; x < 4.0 ; x += 0.005)
      {
            GLdouble func = exp(-x) * cos(2 * 3.14159265 * x);
            glVertex2d(A * x + B, C * func + D);
      }
      glEnd();
      glFlush();           // send all output to display
}
//<<<<<<<<<<<<<<<<<<<<<<< main >>>>>>>>>>>>>>>>>>>>>>
void main(int argc, char** argv)
{
      glutInit(&argc, argv);          // initialize the toolkit
      glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB); // set the display mode
      glutInitWindowSize(screenWidth, screenHeight); // set the window size
      glutInitWindowPosition(100, 150); // set the window position on screen
      glutCreateWindow("Dot Plot of a Function"); // open the screen window
      glutDisplayFunc(myDisplay);     // register the redraw function
      myInit();
      glutMainLoop();              // go into a perpetual loop
}
```

شکل ۵ برنامهٔ کامل رسم یک تابع با استفاده از نقاط.



شکل ٦

رسم اشکال ابتدائی دیگر

رسم خط

```
glBegin(GL_LINES);
        glVertex2i(35,50);
        glVertex2i(100, 200);
glEnd();
```

می توان آنرا بصورت روال drawLine() نوشت:

```
void drawLine(Glint x1, Glint y1, Glint x2, Glint y2)
{
      glBegin(GL_LINES);
          glVertex2i(x1, y1);
          glVertex2i(x2, y2);
    glEnd();
}
```

رنگ خط با همان glColor3f() انتخاب می شود. ضخامت خط با glLineWidth() مثلاً glLineWidth(4.0)

بیش از دو نقطه باعث رسم خط بین هر جفت نقطه می شود

```
glBegin(GL_LINES);
        glVertex2i(35,50);
        glVertex2i(100, 200);
        glVertex2i(70,150);
        glVertex2i(900, 250);
glEnd();
```

رسم چند خطی

```
glBegin(GL_LINE_STRIP);
            glVertex2i(35,50);
            glVertex2i(100, 200);
        glVertex2i(70,150);
        glVertex2i(95, 250);
glEnd();
glFlush();
```

رسم چندخطی از یک پرونده

```
21              number   of polylines in the file
4               number   of points in the first polyline
169     118
174       120
179       124
176       126
5               number    of points in the second polyline
298      86
304       92
310         104
314         114
314         119
29
          32        435
ادامه
```

```
void drawPolyLineFile(char* fileName){
      fstream inStream;
      instream.open(fileName, ios::in);  //open the file
      if (inStream.fail())
            return;
      glClear(GL_COLOR_BUFFER_BIT);      //clear the screen
      GLint numpolys, numlines, x, y;
      inStream >> numpolys;
      for (int j=0; j<numpolys; j++){
            inStream >> numLines;
            glBegin(GL_LINE_STRIP);
               for (int i=0; i<numLines; i++){
                  inStream >> x >> y;
                  glVertex2i(x, y);
               }
            glEnd();
      }
      glFlush();
      inStream.close();
}
```

رسم چند ضلعی

GL_LINE_LOOP

GL_POLYGON

بوسیلهٔ GL_LINE_LOOP نمی توان داخلش را رنگ یا الگودار کرد ولی با GL_POLYGON می توان

مستطیل قائم

glRecti(GLint x1, GLint y1, Glint x2, Glint y2); . (x1,y1) نقطهٔ پائین- چپ و (x2,y2) نقطهٔ بالا- راست مستطیل هستند.

```
glClearColor(1.0, 1.0, 1.0, 0.0);   //white background
glClear(GL_COLOR_BUFFER_BIT);  //clear the window
glColor3f(0.6, 0.6, 0.6);                  //bright gray
glRecti(20,20, 100, 70);
glColor3f(0.2, 0.2, 0.2);              //dark gray
glRecti(70, 50, 150, 130);
glFlush();
```

شکل ۷

برای رنگ نمودن چندضلعیها نیز همانند مستطیل ابتدا رنگ ترسیم را مشخص نموده و سپس دستور رسم چندضلعی را صادر می کنیم. البته در این حالت چندضلعیهای درست رنگ می شوند که مقعر باشند.

اشکال ابتدائی دیگر

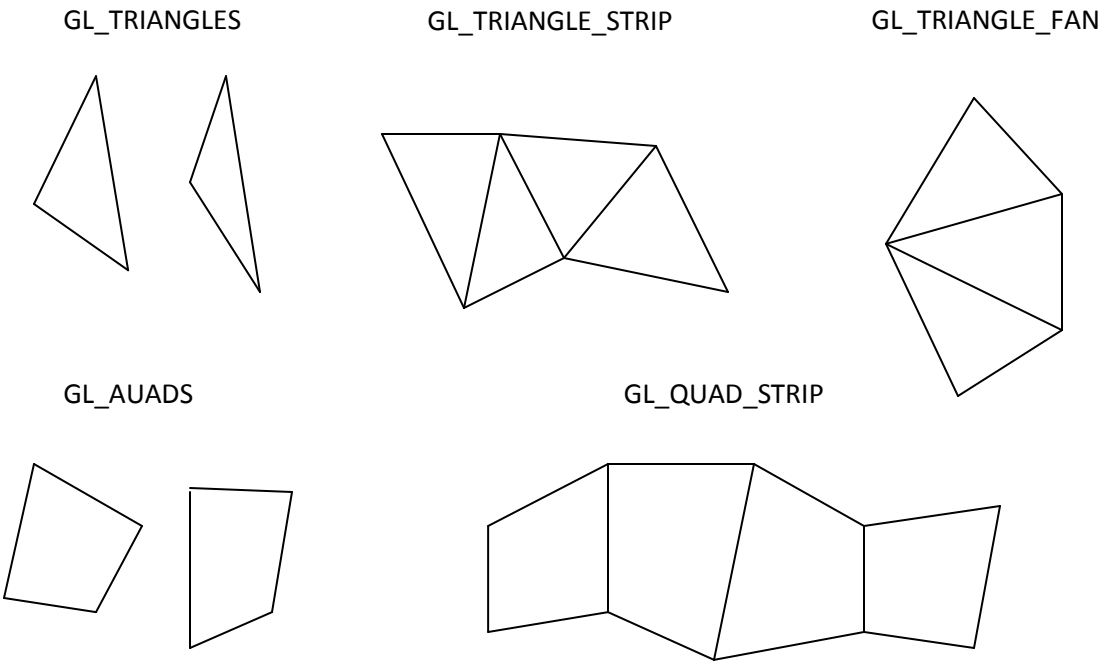
GL_TRIANGLES: دریافت سه نقطه و رسم یک مثلث

GL_QUADS: دریافت چهار نقطه و رسم یک چهار ضلعی

GL_TRIANGLE_STRIP: رسم یک سری مثلث به هم متصل بر اساس سه تائیهائی از نقاط v0,v1,v2 و v2,v1,v3 و v2,v3,v4 و ... (به ترتیبی که همهٔ مثلثها به یک ترتیب مثلاً خلاف جهت عقربه های ساعت طی شوند.

GL_TRIANGLE_FAN: رسم یک سری مثلث متصل بر اساس سه تائیهائی از نقاط v0,v1,v2 و v0,v2,v3 و v0,v3,v4 ... (شبیه بادبزن)

GL_QUAD_STRIP: رسم یک سری چهار ضلعی متصل بر اساس چهارتائیهائی از نقاط v0,v1,v3,v2 و v2,v3,v5,v4 و v4,v5,v7,v6 و ... به گونه ای که همهٔ چهارضلعیها به یک ترتیب مثلاً خلاف جهت عقربه های ساعت طی شوند.



شکل ۸ اشکال ابتدائي ديگر

تعامل با موش و صفحهٔ کلید

تعامل با موش

برای ثبت تابع پاسخ حادثهٔ فشرده شدن یکی از دگمه های موش از glutMouseFunc(myMouse) استفاده می کنیم. نام myMouse دلخواه است.

void myMouse(int button, int state, int x, int y);

- مقادیر button
  - GLUT_LEFT_BUTTON
  - GLUT_MIDDLE_BUTTON
  - GLUT_RIGHT_BUTTON
- مقادیر state
  - GLUT_UP
  - GLUT_DOWN

مقادیر x و y مکان موش در زمان وقوع حادثه را نشان می دهند. باید دقت نمود که x تعداد پیکسلها از سمت چپ پنجره و y تعداد پیکسلها را از بالای پنجره نشان می دهند.

مثال: رسم نقاط با موش

- با فشار دگمهٔ چپ یک نقطه در آن مکان رسم می شود. با فشار دگمهٔ راست رنگ زمینهٔ پنجره تغییر و صفحه با آن رنگ می شود. دقت: نقطه را در (x,y) رسم نمی کنیم. در(x,screenHeight-y)

```
void myMouse(int button, int state, int x, int y) {

   if(state == GLUT_DOWN)
   {
      if(button == GLUT_LEFT_BUTTON)
      {
        glPointSize(5.0);
        glBegin(GL_POINTS);
           glVertex2i(x, screenheight - y);
        glEnd();
        glFlush();
      }
      else if (button == GLUT_RIGHT_BUTTON){
         glClearColor(1.0f, 0.0f, 0.0f, 0.0f); // Red
         glClear(GL_COLOR_BUFFER_BIT);
         glFlush();
      }
   }
}
```

مثال: رسم مستطیل با موش:

```
#include <windows.h>
#include <gl/Gl.h>
#include <gl/glut.h>

struct GLintPoint {

	GLint x, y;

};

GLintPoint	corners[2];
bool		selected = false;
int			screenWidth = 640, screenHeight = 480;

void myDisplay() {

	glClear( GL_COLOR_BUFFER_BIT );
	glMatrixMode( GL_MODELVIEW );
	glLoadIdentity();
	glColor3f( 1.0f, 1.0f, 1.0f );

	if( selected ) {

		glBegin( GL_QUADS );

		glVertex2i( corners[0].x, corners[0].y );
		glVertex2i( corners[0].x, corners[1].y );
		glVertex2i( corners[1].x, corners[1].y );
		glVertex2i( corners[1].x, corners[0].y );

		glEnd();

	}

	glutSwapBuffers();
	glFlush();

}

void myMouse( int button, int state, int x, int y ) {

	if( button == GLUT_LEFT_BUTTON && state == GLUT_DOWN ) {

		corners[0].x = x;
		corners[0].y = screenHeight - y;
		selected = true;

	}

	glutPostRedisplay();

}

void myPassiveMotion( int x, int y )  {

	corners[1].x = x;
	corners[1].y = screenHeight - y;
	glutPostRedisplay();

}
```

شکل ۹ برنامه ترسیم مستیل با حرکت موش.

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

        glutInit( &argc, argv );

        // initialize window
        glutInitWindowSize( screenWidth, screenHeight );
        glutInitWindowPosition( 0, 0 );
        glutInitDisplayMode( GLUT_RGB | GLUT_DOUBLE );

        // create window
        glutCreateWindow( "Rubber Rect Demo" );

        // set the projection matrix
        glMatrixMode( GL_PROJECTION );
        glLoadIdentity();
        gluOrtho2D( 0, screenWidth, 0, screenHeight );
        glMatrixMode( GL_MODELVIEW );
        // clear rendering surface
        glClearColor(0.0f, 0.0f, 0.0f, 0.0f);  // background is black
        glViewport(0, 0, screenWidth, screenHeight);


        glutMouseFunc( myMouse );
        glutDisplayFunc( myDisplay );
        glutPassiveMotionFunc( myPassiveMotion );
        glutMainLoop();

        return( 0 );

}
```

شکل ۹ برنامهٔ ترسیم مستطیل با استفاده از حرکت موش (ادامه).

تعامل با صفحهٔ کلید

هنگام فشردن یک کلید یک حادثه رخ می دهد. ثبت توسط

glutKeyboardFunc(myKeyboard);

تابع پاسخ

void myKeyboard(unsigned int key, int x, int y);

key مقدار ASCII کلید فشرده شده می باشد. مقادیر x و y مکان موش در هنگام فشرده و آزاد شدن کلید را نشان می دهند. همانند قبل y نسبت با بالای پنجره سنجیده می شود. توسط این روال می توان گزینه های گوناگونی را در مقابل فشردن هر کلید برای کاربر مهیا ساخت. این کار معمولاً توسط استفاده از دستور switch و وجود یک case برای هر کلید در برنامه انجام می شود.

مثال:

رسم یک نقطه در مکان موش اگر کلید p فشرده و خروج اگر کلید E فشرده شود.

```
void myKeyboard(unsigned char key, int mouseX, int mouseY)
{
	GLint x = mouseX;
	GLint y = screenHeight - mouseY;
	glPointSize(5.0);
	switch(key){
		case 'p':
			glBegin(GL_POINTS);
			{
				glVertex2i(x, y);
			}
			glEnd();
			break;
		case 'E':
			exit(-1);
		default:
			break;
	}
	glFlush();

}
```

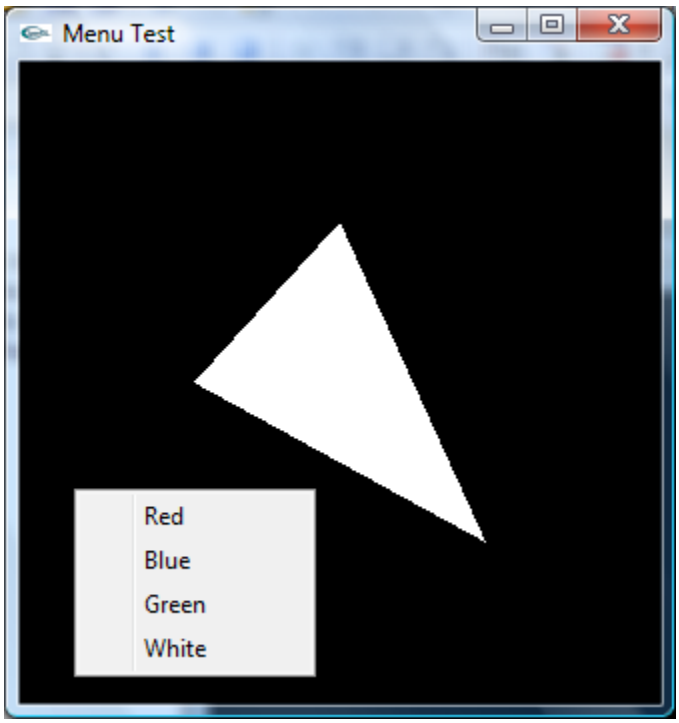
استفاده از منوها

ایجاد منو توسط دستور glutCreateMenu(processMenuEvents) انجام می شود. روال processMenuEvents باید نوشته شود. اعمالی که توسط انتخاب هر گزینهٔ منو باید انجام شود را می گرداند. معولاً توسط دستور switch . اضافه کردن منو توسط glutAddMenuEntry()

```
glutAddMenuEntry("Red",RED);
glutAddMenuEntry("Blue",BLUE);
glutAddMenuEntry("Green",GREEN);
glutAddMenuEntry("White",WHITE);
```

کاربر می تواند هر تعداد منو که بخواهد اضافه کند. چسباندن به کلید راست موش توسط دستور glutAttachMenu(GLUT_RIGHT_BOTTOM); انجام می شود. هر کلید دیگر موش نیز می تواند استفاده شود.

مثال:



شکل ۱۰ یک منوي GLUT.

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

#define RED 1
#define GREEN 2
#define BLUE 3
#define WHITE 4

float angle = 0.0;
float red=1.0, blue=1.0, green=1.0;

void renderScene(void) {
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glLoadIdentity();
	glRotatef(angle,0.0,1.0,0.0);

	glColor3f(red,green,blue);

	glBegin(GL_TRIANGLES);
		glVertex3f(-0.5,-0.5,0.0);
		glVertex3f(0.5,0.0,0.0);
		glVertex3f(0.0,0.5,0.0);
	glEnd();
	angle++;
	glutSwapBuffers();
}

void processMenuEvents(int option) {

	switch (option) {
		case RED : red = 1.0; green = 0.0; blue = 0.0; break;
		case GREEN : red = 0.0; green = 1.0; blue = 0.0; break;
		case BLUE : red = 0.0; green = 0.0; blue = 1.0; break;
		case WHITE : red = 1.0; green = 1.0; blue = 1.0; break;
	}
}

void main(int argc, char **argv) {
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
	glutInitWindowPosition(100,100);
	glutInitWindowSize(320,320);
	glutCreateWindow("Menu Test");
	glutDisplayFunc(renderScene);
	glutIdleFunc(renderScene);

	//call our function to create the menu

	glutCreateMenu(processMenuEvents);
	glutAddMenuEntry("Red",RED);
	glutAddMenuEntry("Blue",BLUE);
	glutAddMenuEntry("Green",GREEN);
	glutAddMenuEntry("White",WHITE);
	glutAttachMenu(GLUT_RIGHT_BUTTON);

	glutMainLoop();
}
```