



# **COMPUTER GRAPHICS**

## **CH-Two**

By  
Assis. L. Mohamed A. Abdul-Hamed  
Computer science & IT collage – Basra – Iraq  
2019

# Outline

- **Writing programs that produce pictures.**
- **Learn the basic ingredients found in every OpenGL program.**
- **Develop some elementary graphics tools for drawing lines, polylines, and polygons.**
- **Develop tools that allow the user to control a program with mouse and keyboard.**

# GETTING STARTED MAKING PICTURES

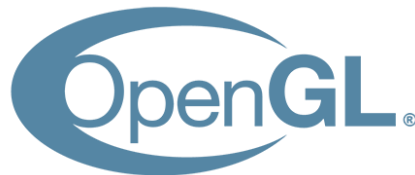
- To get started, you need an environment that let you write and execute programs.
- This environment includes hardware to display pictures and a library of software tools.
- **Every graphics program begins with :**

A- Initializations that establish the desired display mode.


B- Set up a coordinate system for specifying points, lines, ... etc.

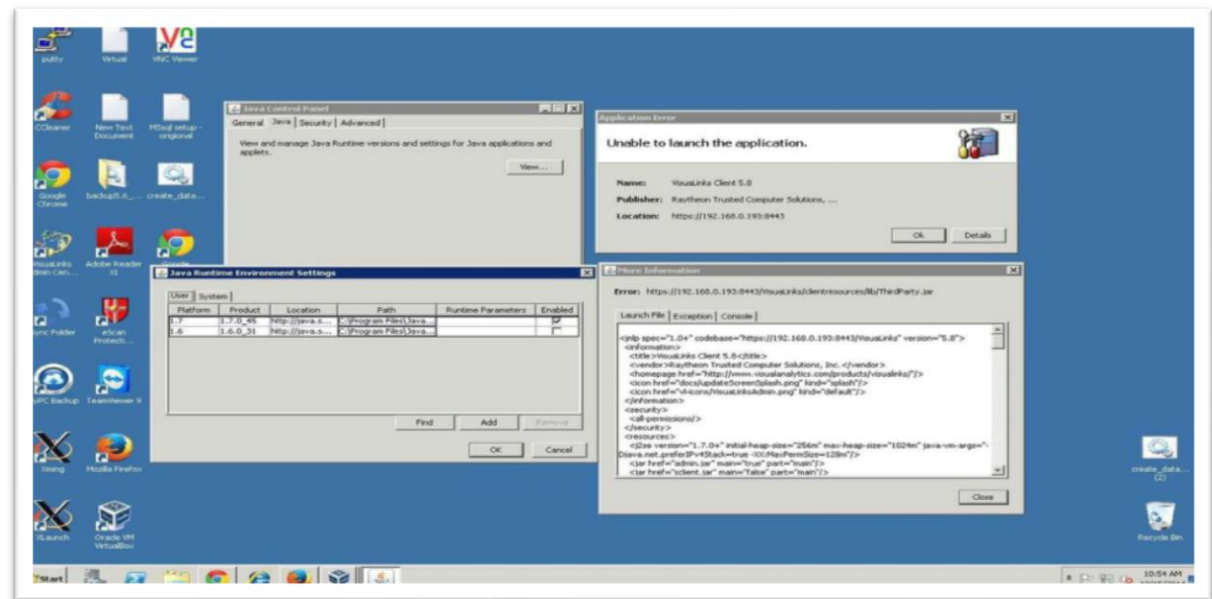
# Device-independent Programming

- Writing graphics applications is made available, can be compiled and run on a variety of graphics environments.
- This is known as **device-independent** graphics programming, **OpenGL** offer such a tool.
- OpenGL : is an “application programming interface” (API): a collection of routines that the programmer can call to produce graphics.



# Windows-based Programming

- Many graphics systems are windows based and manage the display of multiple overlapping (تداخل) windows.
  - The user can move windows around and resize them.
- 



# Event-driven Programming

- Means that the program responds to various events, such as click a mouse, press a key, resize a window.
- The programmer organizes a program as a collection of **callback functions** (دوال استدعاء) that are executed when events occur.
- The new programming structure is more like “do nothing until an event occurs, and then do specified thing.”
- Such as: **glutMouseFunc(myMouse); // register the mouse action function**  
**register** the function **myMouse()** as the function to be executed when a mouse event occurs.

# Main Structure of OpenGL Program

```
void main () {  
    initialize things  
    create a screen window  
    glutDisplayFunc (myDisplay) ; // register the redraw function  
    glutReshapeFunc (myReshape) ; // register the reshape function  
    glutMouseFunc (myMouse) ; // register the mouse action function  
    glutKeyboardFunc (myKeyboard) ; // register the keyboard action function  
    perhaps initialize other things  
    glutMainLoop ();  
}
```

all of the callback functions are defined here

# There are four principal types of events for OpenGL

## 1- **glutDisplayFunc (myDisplay);**

- window should be redrawn on the screen, it issue a “redraw” event.
- This happens when the window is first opened and when the window is exposed (تتعرض) by moving another window off of it.
- The function **myDisplay ()** is registered as the callback function for a redraw event.



# There are four principal types of events for OpenGL

## 2- **glutReshapeFunction (myReshap);**

- screen windows can be reshaped by the user, dragging a corner of the window to new position with mouse.
- myReshape () is registered with the reshape events, and automatically passed arguments that specify the new width and height of the reshape window.

# There are four principal types of events for OpenGL

## 3- **glutMouseFunc (myMouse);**

- when one mouse buttons is pressed or released (تصدر), a mouse event occurs.
- The function **myMouse ()** is automatically passed arguments that describe the location of the mouse and nature of the action.

# There are four principal types of events for OpenGL

## 4- **glutKeyboardFunc (myKeyboard);**

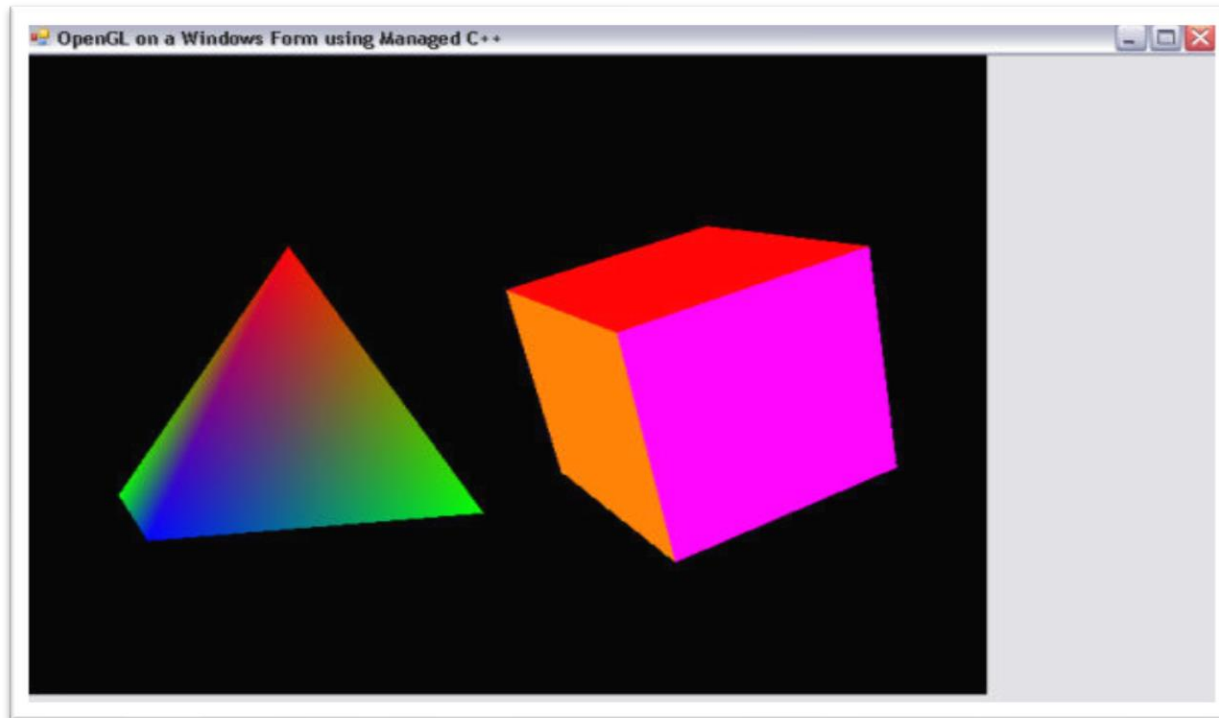
- registers the function **mykeyaboard ()** with event of pressing or releasing some key on the keyboard.
- automatically passed arguments that tell which key was pressed.

# Note

If a program does not make use of a mouse (or keyboard), the corresponding callback function need not be registered or written. The mouse click (or key pressed) have no effect in the program.

# Opening a Window for Drawing

- The first task in making pictures is to open a screen window for drawing.



# Entire *main ()* function to draw graphics in a screen window.

```
void main (int argc, char** argv) {  
    glutInit (&argc, argv);      // initialize the toolkit  
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB); // set display mode  
    glutInitWindowSize (640,480); // set window size  
    glutInitWindowPosition (100, 150); // set window position on screen  
    glutCreateWindow ("my first attempt"); // open the screen window  
    // register the callback functions  
    glutDisplayFunc (myDisplay);  
    glutReshapeFunc (myReshape);  
    glutMouseFunc (myMouse);  
    glutKeyboardFunc (myKeyboard);  
    myInit ();                    // additional initializations as necessary  
    glutMainLoop();              // go into a perpetual loop  
}
```

# Conti...

- The first five calls use the OpenGL Utility Toolkit to open a window for drawing.
- **glutInit (&argc, argv);**
- This function initialize the OpenGL Utility Toolkit.
- Its arguments (معاملات) are the standard ones for passing information about the command lines.

# Conti...

- **glutInitDisplayMode (GLUT\_SINGLE | GLUT\_RGB);**
- specifies how the display should be initialized.
- The built-in constants GLUT\_SINGLE and GLUT\_RGB, which are Oared (التي طرأت عليه) together, indicate that a single display buffer should be allocated and that colors are specified.
- **glutInitWindowSize (640,480);**
- specifies the screen window should initially be 640 pixel wide by 480 pixel high.
- You may resize the window as desired.



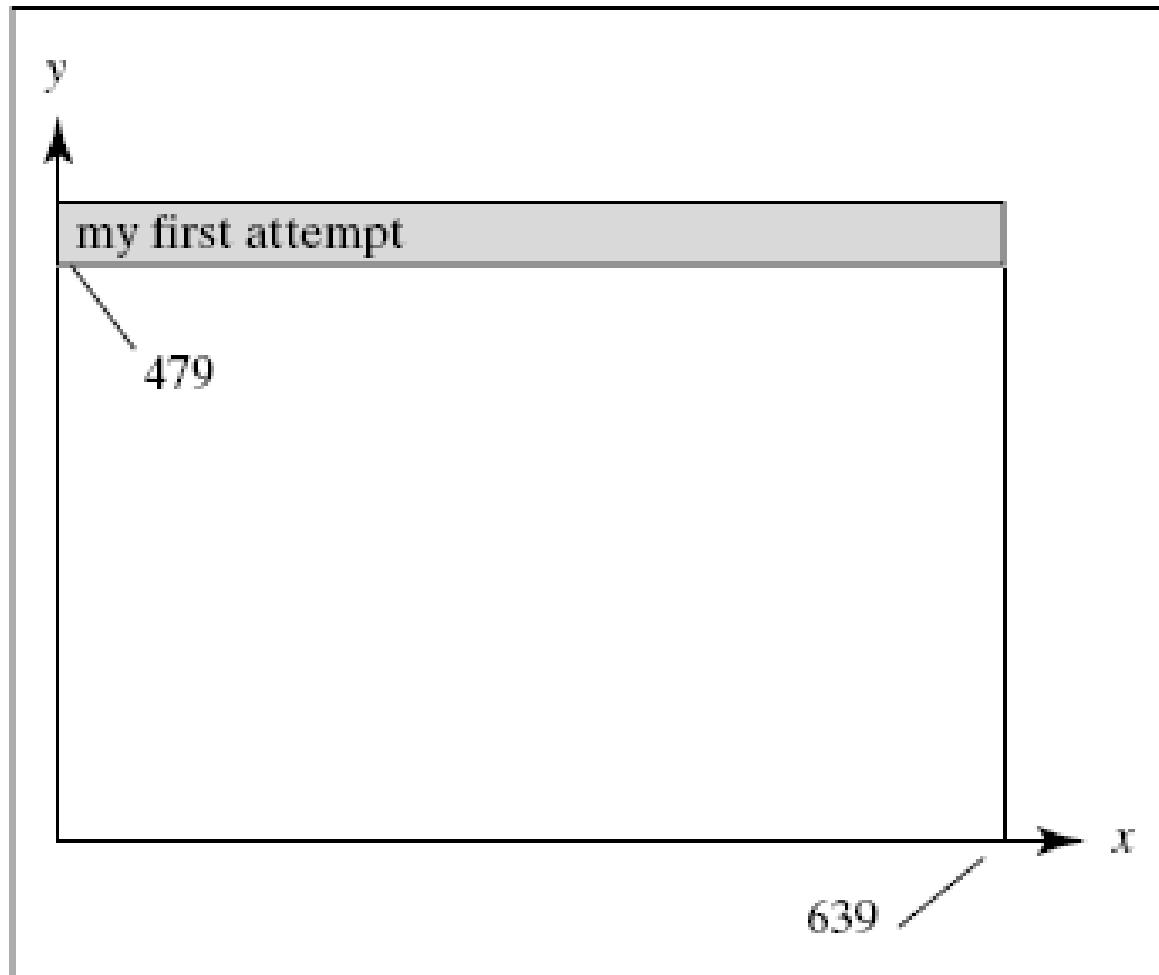
# Conti...

- **glutInitWindowPosition (100, 150);**
- specifies that the window's upper left corner should be positioned on the screen 100 pixels over from the left edge and 150 pixels down from the top.
- You can move this window wherever desired.
- **glutCreateWindow ("my first attempt");**
- This function actually opens and displays the screen window, putting the title “my first attempt” in the title bar.

# DRAWING BASIC GRAPHICS PRIMITIVES

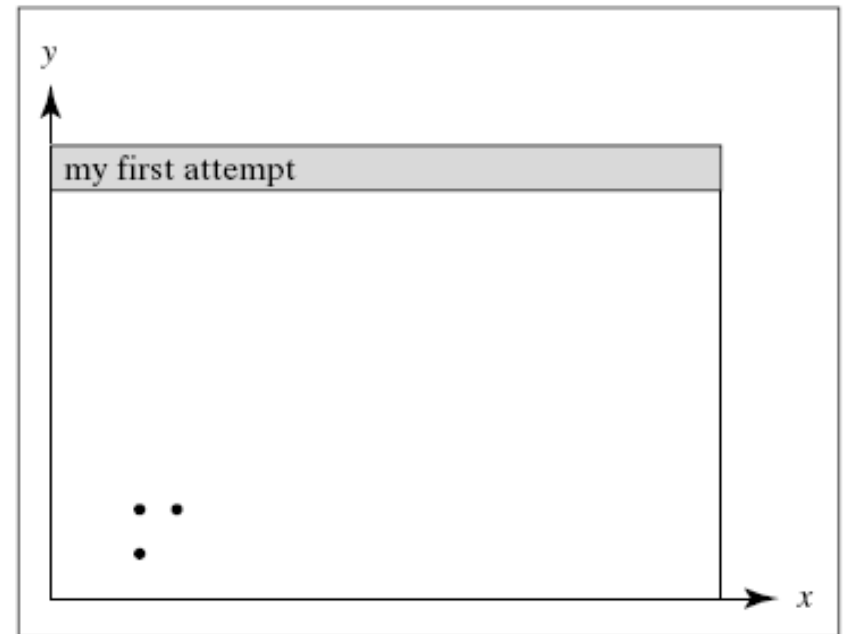
- The drawing commands will be placed in the **callback function** associated with a redraw event, such as myDisplay() function.
- We first establish the **coordinate system** where the objects will appear.
- We begin with an intuitive (بدیهی) coordinate system, which is 640 pixel wide by 480 pixels high.

# The initial coordinate system for drawing



- OpenGL provides tools for drawing all of the output primitives.
- To draw object, you pass a list of **vertices**.
- The list occurs between two functions **glBegin ()** and **glEnd ()**.

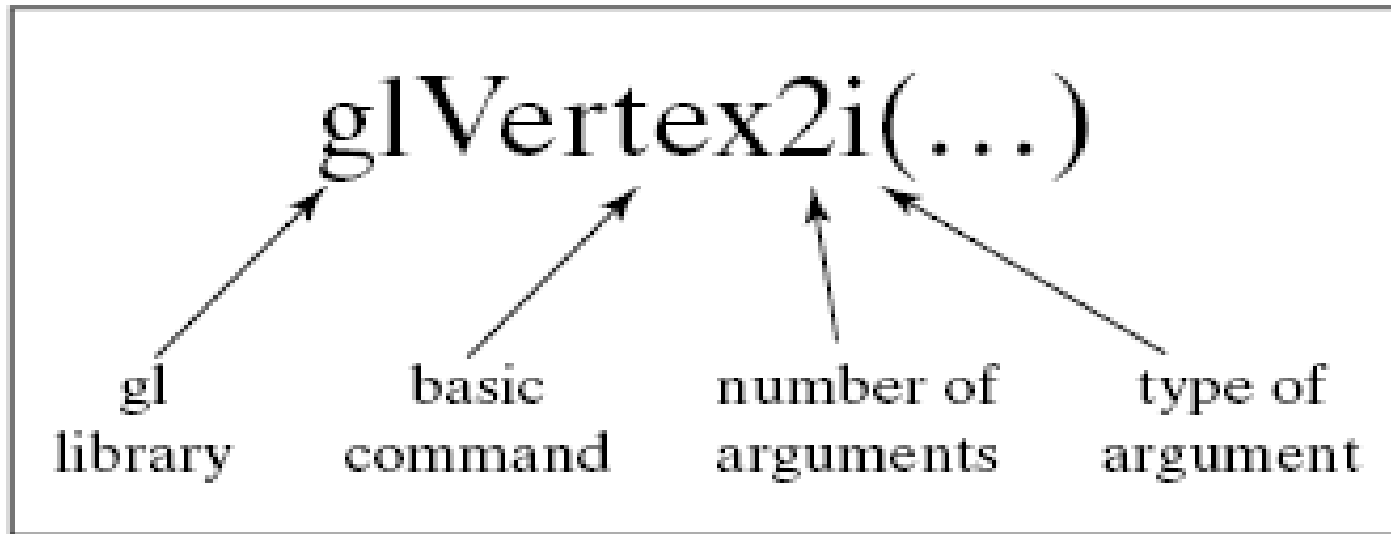
```
glBegin (GL_POINTS);  
    glVertex2i (100, 50);  
    glVertex2i (100, 130);  
    glVertex2i (150, 130);  
glEnd ();
```



# Describe previous code

- The constant **GL\_POINTS** is built-into OpenGL.
- To draw other primitives, you replace **GL\_POINTS** with **GL\_LINES**, **GL\_POLYGON**, etc.
- The function **glVertex2i ()** have several variations, that distinguish the number and type of arguments passed to the function, as in next slide.

# The function **glVertex2i()**



- ❖ **(gl)** indicates a function from OpenGL library.
- ❖ Then basic command root, followed by the number of arguments sent to the function (3 and 4), the argument **(i)** for integer, **(f)** or **(d)** for a floating-point value.

# OpenGL Data Types

Suffix	Data type	Typical C or C++ type	OpenGL type name
b	8-bit integer	signed char	GLbyte
s	16-bit integer	short	GLshort
i	32-bit integer	int or long	GLint, GLsizei
f	32-bit floating point	float	GLfloat, GLclampf
d	64-bit floating point	double	GLdouble, GLclampd
ub	8-bit unsigned number	unsigned char	GLubyte, GLboolean
us	16-bit unsigned number	unsigned short	GLushort
ui	32-bit unsigned number	unsigned int or unsigned long	GLuint, GLenum, GLbitfield

- A function using the suffix (لاحقة) i “expects” a 32-bit integer, when your use 16-bit integer, this caused a problem as in the following code.

The wrong code

```
void drawDot (int x, int y) {           ← danger: pass ints
    glBegin (GL_POINTS); // draw dot at integer point (x, y)
    glVertex2i ( x, y);
    glEnd ();
}
```

The safer code

```
void drawDot (GLint x, GLint y) {
    glBegin (GL_POINTS); // draw dot at integer point (x, y)
    glVertex2i ( x, y);
    glEnd ();
}
```



# The OpenGL State

- OpenGL keeps track (يتتبع) of many state variables, such as current size of a point, color, background, ... etc.
- The values of a state variable remains active until a new value is given.
- The color of a drawing can be specified using

```
glColor3f (1.0,0.0, 0.0); // set drawing color to red  
glColor3f (0.0,0.0, 0.0); // set drawing color to black  
glColor3f (1.0,1.0, 1.0); // set drawing color to white
```

# The OpenGL State

- **glClearColor (red, green, blue, alpha);**
- The background color, where **alpha** specifies a degree of transparency (شفافية).
- **glClear (GL\_COLOR\_BUFFER\_BIT);**
- The argument `GL_COLOR_BUFFER_BIT` is a constant.
- This function used to clear the entire window to the background color.

# Complete program that draws the lowly three dots

[illegible]

# Display Function

```
>>>>>>>>>>>>>>>>>>> myDisplay <<<<<<<<<<<<<<<<<<<<<<<<
```

```
void myDisplay(void) {  
    glClear (GL_COLOR_BUFFER_BIT); // clear the screen  
    glBegin (GL_POINTS);  
    glVertex2i(100, 50);           // draw three points  
    glVertex2i(100, 130);  
    glVertex2i(150, 130);  
    glEnd ();  
    glFlush();                     // send all output to display
```



# Main Function

[illegible]

```
void main (int argc, char** argv) {
    glutInit (&argc, argv);      // initialize the toolkit
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB); // set display mode
    glutInitWindowSize (640,480); // set window size
    glutInitWindowPosition (100, 150); // set window position on screen
    glutCreateWindow ("my first attempt"); // open the screen window
    glutDisplayFunc (myDisplay);  // register redraw function
    myInit ();
    glutMainLoop ();              // go into a perpetual loop
}
```

*Thanks!*