What is the workshop about?

We use software such as AutoCAD, 3D Studio, Maya and many others for a host of applications ranging from Technical Drawings to Machine Design to Game Making to Special Effects in Movies and even entire movies... The list is nearly endless. This workshop is not about using such software, but rather its about creating them. It introduces the basics of 3D graphics programming which are necessary for writing such software and for other applications. For this workshop, we will be using Microsoft's Visual C++ Development Environment and OpenGL.

What is OpenGL?

From OpenGL Programming Guide : "OpenGL is a powerful software interface used to produce high-quality computer generated images and interactive applications using 2D and 3D objects and color bitmaps and images"

We will need some basic idea of Computer Graphics before we can discuss 3D graphics and OpenGL. The BIOS (Basic Input/Output System) of the computer provides low level access to all the devices connected to the system, including video hardware. Using the functions it provides, you can perform operations like plotting a pixel on screen etc. But these functions are very limited. Only basic operations are provided for. Anything more requires implementing algorithms. Many efficient algorithms have been developed for drawing lines, circles and other figures. Similarly, other graphics concepts like transparency, lighting etc. have to be implemented by the programmer. However, since these requirements are pretty standard, every graphics programmer need not implement them. This job is done by companies, groups of programmers or even individuals and provided to the programming community in the form of libraries or APIs (Application Programming Interfaces). Programmers can use these libraries of precompiled code and write their applications. They are still free to implement any or all of these functions themselves, if they like. For example, Borland's Turbo C provides a basic graphics library in the form of BGI (Borland Graphics Interface). We can use the functions that it provides by including the graphics.h header file in our programs and linking against the graphics library while compiling.

Thus for Graphics Programming (in fact, for any sort of programming)

1) We can implement the required basic functions ourselves and then build applications using these functions or
2) Use existing code libraries for these basic functions and concentrate more on the applications to be developed

OpenGL (Open Graphics Library) is one such library, meant for 2D and 3D graphics. There are many APIs for 3D graphics but the most notable among them are OpenGL and DirectX. DirectX is Microsoft's proprietary API and is supported mostly (read only) on Windows platforms whereas OpenGL is an open standard implemented on almost all major platforms such as Linux, Mac, BeOS, Windows and many others. Although OpenGL was originally developed by SGI for its high-end IRIX workstations, it has grown by leaps and bounds and now its development is looked after by a group consisting of industry leaders such as 3DLabs, HP, Evans & Sutherland, IBM, Intel, Intergraph, Microsoft, NVIDIA and SGI itself. This group is called the ARB (OpenGL Architecture Review Board).

Most of these libraries can be used with any programming language and development environment that supports external libraries. OpenGL, for instance, can be used with C/C++ (Microsoft Visual C++, Borland C for Windows, GNU C/C++ and others), VB, Delphi, Java, Dot NET to name just a few.

What is GLUT?

OpenGL provides all the functions necessary for 3D graphics. These are typically implementations of well known algorithms. But before we can draw anything on the screen, we need to create a window for our application. To handle user interaction, we need to respond to keyboard, mouse, joystick and other hardware events. These operations are tightly coupled to the underlying system architecture and operating system being used. As such, familiarity with the OS specific APIs is necessary. However if your goal is just to learn OpenGL and graphics in general, you may have lost your interest by the time you become familiar with the Operating System APIs. This problem is addressed by the GLUT library, written by Mark Kilgard, a former employee of SGI. It stands for OpenGL Utility Toolkit and provides functions for the necessary OS specific tasks described above. These functions remain same no matter what OS you are using, as long as a GLUT implementation is available for that particular OS. GLUT carries out the platform dependent tasks by compiling different code on different platforms and presents a uniform interface to the programmer. Your GLUT OpenGL program will compile, almost without changes, on most platforms.

GLUT is an invaluable tool for beginners but it can also be used in applications that are fairly complex. However, applications that require heavy system resources and speed such as 3D games, 3D Application Software etc need to be written using platform specific APIs to achieve the maximum performance possible.

How do I install OpenGL and GLUT on my system?

For developing and running OpenGL programs that do not use GLUT, mostly nothing needs to be done. Almost all major Operating Systems come preinstalled with the required libraries. However, in the rare case that an installation is necessary, the required libraries can be downloaded from the links provided at www.opengl.org – the official site for OpenGL.

For developing and running programs that use GLUT, the required libraries need to be installed on Windows. Most Linux distributions come preinstalled with the GLUT library. GLUT can be downloaded from www.opengl.org. It links to the GLUT homepage. GLUT is also available for download along with a sample program at my website. Point your browser to http://geocities.com/abhijeet_maharana and go to the Downloads section. Alternately, the source code of GLUT can also be downloaded and compiled.

Where can I learn more about OpenGL and Graphics in general?
Web Sites

www.opengl.org	Official OpenGL site. The discussion forum is very good. Has nearly exhaustive Links to OpenGL sites.
www.gamedev.net	A general site about graphics and game programming. More about efficient algorithms etc.
www.nehe.gamedev.net	A site dedicated entirely to OpenGL. My OpenGL journey (which is negligible) started here.
http://www.cs.utah.edu/~narobins/opengl.html	OpenGL tutorial programs. Highly recommended.

www.gametutorials.com	Tutorials on OpenGL, Windows Programming, DirectX etc. Has some pretty good examples.
www.naturewizard.com	Simulation of nature using OpenGL for graphics. Algorithms for L-systems, generating planets and terrains etc. Advanced stuff.

Books

OpenGL Programming Guide (Pearson Education, ISBN 81-297-0058-1)
Dave Shreiner
Jackie Neider
Mason Woo
Tom Davis

OpenGL SuperBible

Computer Graphics, C version (Pearson Education, ISBN 81-7808-794-4)
Donald Hearn
M. Pauline Baker

Computer Graphics (Schaum's Series)

Sample Program for Workshop

(Original program downloaded from www.dev-gallery.com)

The workshop will be conducted in the following steps:

1 Depth Testing and Lighting
2-1 Transformations with constant parameters
2-2 Transformations with varying parameters
2-3 Negating the effect of previous transformations
[glPushMatrix() and glPopMatrix()]
3 Hierarchical Model example
4 Playing with the color buffer

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

2           void display(void);
3           void keys(unsigned char, int, int);
4           void init(void);
5 (2-2)     void anim(void);
6           void reshape(int, int);

7 (2-2)     float f = 0.0;
8 (4)       bool color = true;

9           void main(int argc, char** argv)
10          {
11              // Initialise glut
12              glutInit(&argc, argv) ;
```

```
13              // Create a window for our program
14              glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH) ;
15              glutInitWindowSize(400,400) ;
16              glutCreateWindow("OpenGL Workshop") ;

17              // Initialisations for our program
18              init() ;

19              // Register Callbacks
20              glutDisplayFunc(display) ;
21              glutReshapeFunc(reshape) ;
22              glutKeyboardFunc(keys) ;
23              glutIdleFunc(anim);

24              // Start the Main Loop
25              glutMainLoop() ;
26          }

27          void init(void)
28          {
29              glClearColor(0.0, 0.0, 0.0, 0.0) ;
30 (1)         glEnable(GL_DEPTH_TEST);
31 (1)         glEnable(GL_LIGHTING);
32 (1)         glEnable(GL_LIGHT0);
33 (1)         glEnable(GL_COLOR_MATERIAL);
34          }

35          void display(void)
36          {
37 (4)          if(color)
38 (4)              glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT) ;
39 (4)          else
40 (4)              glClear(GL_DEPTH_BUFFER_BIT) ;

41 (2-3)        glPushMatrix();

42 (2-1)        // glRotatef(45, 1.0, 1.0, 1.0);
43 (2-2)        // glRotatef(f, 1.0, 1.0, 1.0);

44 (2-1)        // glTranslatef(10, 0, 0);
45 (2-2)        // glTranslatef(f, 0, 0);

46 (2-1)        // glScalef(0.5, -2, 1);
47                  // absolute value is taken if negative parameters are passed
48 (2-2)        // glScalef(f, 1, 1);

49              glRotatef(f, 1.0, 1.0, 1.0);
50              glColor3f(1.0, 0.5, 0.0);
51              glutSolidSphere(2.5, 20, 20);
52              glColor3f(1.0, 0.0, 0.0);
53              glutSolidCube(4.0) ;

54              // glutWireTetrahedron();
```

```
55              // glutWireCone(5, 5, 3, 5);
56              // glutWireTorus(0, 4, 3, 10);
57              // glutWireTeapot(5);

58 (3)           //glPushMatrix();
59 (3)           //glTranslatef(10.0, 0.0, 0.0, 0.0);
60 (3)           //glutSolidIcosahedron();
61 (3)           //glPopMatrix();
62 (3)           //glTranslatef(5.0, 0.0, 0.0);
63 (3)           //glutSolidCube(2);

64              for(int i=0; i<10; i++)
65              {
66                  glPushMatrix();

67                  glRotatef(36*i, 0, 0, 1);
68                  glTranslatef(10, 0, 0);
69                  glRotatef(-2*f, 0, 1, 0);
70                  glTranslatef(0, 0, 2);
71                  glutSolidIcosahedron();

72                  switch(i)
73                  {
74                          case 0:
75                                  glColor3f(1.0, 0.1, 0.7);
76                                  break;
77                          case 1:
78                                  glColor3f(1.0, 0.0, 1.0);
79                                  break;
80                          case 2:
81                                  glColor3f(0.0, 0.0, 1.0);
82                                  break;
83                          case 3:
84                                  glColor3f(0.0, 0.5, 1.0);
85                                  break;
86                          case 4:
87                                  glColor3f(0.0, 1.0, 0.7);
88                                  break;
89                          case 5:
90                                  glColor3f(0.0, 1.0, 0.0);
91                                  break;
92                          case 6:
93                                  glColor3f(0.5, 1.0, 0.0);
94                                  break;
95                          case 7:
96                                  glColor3f(1.0, 1.0, 0.0);
97                                  break;
98                          case 8:
99                                  glColor3f(1.0, 0.5, 0.0);
100                                 break;
101                         case 9:
102                                 glColor3f(1.0, 0.0, 0.0);
103                                 break;
104                 }
```

```
105                 glRotatef(3*f, 1, 0, 1);
106                 glutSolidTorus(0.2, 1, 10, 10);
107                 glRotatef(-2*f, 1, 0, 1);
108                 glutSolidTorus(0.2, 1, 10, 10);

109                 glPopMatrix();
110             }

111 (2-3)      glPopMatrix();
112             glutSwapBuffers() ;
113         }

114         void keys(unsigned char key, int x, int y)
115         {
116             switch (key)
117             {
118 (4)             case 'a':
119 (4)             case 'A':
120 (4)                 color = !color;
121 (4)                 break;
122                 case 27:
123                     exit(0);
124             }
125         }

126 (2-2)  void anim(void)
127 (2-2)  {
128 (2-2)      f += 0.1;
129 (2-2)      glutPostRedisplay();
130 (2-2)  }

131         void reshape(int w, int h)
132         {
133             glViewport(0,0,(GLsizei)w,(GLsizei)h) ;
134             glMatrixMode(GL_PROJECTION) ;
135             glLoadIdentity() ;
136             gluPerspective(60.0,(GLfloat)w/(GLfloat)h,1.0,60.0) ;

137             glMatrixMode(GL_MODELVIEW) ;
138             glLoadIdentity() ;
139             gluLookAt(0.0,0.0,25.0,   0.0,0.0,0.0,   0.0,1.0,0.0) ;
140         }
```

Some GLUT functions and their parameters

Apart from providing functions for handling OS specific tasks, the GLUT library also provides helper functions for creating graphics primitives which would require quite a few lines of code and additional efforts when created using the OpenGL library directly.

```
glutWireSphere(GLdouble radius, GLint slices, GLint stacks);
glutSolidSphere(GLdouble radius, GLint slices, GLint stacks);

glutWireCone(GLdouble base, GLdouble height, GLint slices, GLint stacks);
```

```
glutSolidCone(GLdouble base, GLdouble height, GLint slices, GLint stacks);

glutWireCube(GLdouble size);
glutSolidCube(GLdouble size);

glutWireTorus(GLdouble innerRadius, GLdouble outerRadius, GLint rings, GLint sides);
glutSolidTorus(GLdouble innerRadius, GLdouble outerRadius, GLint rings, GLint sides);

glutWireTeapot(GLdouble size);
glutSolidTeapot(GLdouble size);

glutWireTetrahedron( );
glutSolidTetrahedron( );

glutWireOctahedron( );
glutSolidOctahedron( );

glutWireDodecahedron( );
glutSolidDodecahedron( );

glutWireIcosahedron( );
glutSolidIcosahedron( );
```

Troubleshooting

1) Check if the solids have been drawn in the same colour as the background colour, which is black by default.

2) Check if the parameters to the solids are valid. For example, a sphere with radius 0 won't show up on the screen. Also you can't have a cone with just one slice.

3) Check the transformations. If you translate, rotate, scale or combine these transformations and go out of screen, the solids won't be visible.

4) Check if the reshape( ) function is exactly as given in the printout. The matrices required for 3D graphics are set up here. If they are initialised incorrectly, part or whole of the drawing may not be visible.

Workshop on 3D Graphics Programming
Feedback

Name :

Class :

Comments and Suggestions :