GRAPHICS PROGRAMMING IN LINUX

This article covers some basics of graphics programming in C.

I was a very avid graphics programmer, using Turbo C (actually, using graphics.c, a library usually taught to engineering students during labs). Trying to push graphics.c to its limits, I got stuck with its very limited graphics functionality and maximum resolution. When I discussed this with a teacher, he introduced me to OpenGL. I tried a few programs, and got fascinated with the rich graphics experience and platform independence. I thought of sharing my experiences with graphics programming using OpenGL, and contacted the LFY editorial team members, who were ready to provide me a platform.

OpenGL basics

OpenGL (Open-source Graphics Library) has three major library header files: gl.h, glu.h and glut.h. The gl.h (graphics library) is a low-level header file, with which you can draw lines, polygons, colour the background or the line, etc. The glu.h (graphics library utility) is a medium-level header. It uses the lower-level OpenGL functions, such as matrices for specific viewing orientation, rendering surfaces, etc. The glut.h (graphics library utility tool-kit) is a high-level library file and a window system-independent tool-kit to hide the complexity of different window system APIs.

Installation

Now, we need a Linux-based OS (I use Kubuntu 11.10); a compiler (GCC); an editor (Kate, Gedit, Kwrite, etc); the OpenGL header files… and some basic knowledge of C programming.
Make sure your PC has an Internet connection. Open a terminal and run sudo apt-get install gcc, entering the password when prompted, to install GCC. Next, for the

editor, run sudo apt-get install kate (or gedit, kwrite or whatever you prefer, instead of kate). For the OpenGL headers, run □sudo apt-get install freeglut3-dev.

The Hello World first step

Now for the classic first-time program. Open the editor, and enter the following code:

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


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

    glutInit(&argc, argv);

    glutInitWindowPosition(100,100);

    glutInitWindowSize(500,500);

    glutCreateWindow(□Hello World□);

    glutMainLoop();

}
```

Save the file as basic.c, and to compile it, run gcc basic.c -l glut at the terminal. Run the compiled binary with ./a.out. Here, we have imported glut.h (in the GL folder). To locate it, run whereis GL and it will give you the installed location, like GL: /usr/include/GL. There are just five basic functions used to create a window for drawing. The first, glutInit(), initialises the display. Then glutInitWindowPosition (int x, int y) specifies the screen location (upper-left corner) of the window. Next, glutInitWindowSize(int x, int y) specifies the size of the window; glutCreateWindow(char *string) names the window for identification, and creates it; andGlutMainLoop() is used to display the window and begin event processing.

The compilation command specified -l glut, which means, a link with the glut library file. By default, GCC names the output compiled binary a.out; to change it, specify the -o (output file name) switch for example, gcc begin.c -l glut -o hello.
Change the background colour of a window

The OpenGL function glutDisplayFunc (void function_name) is used to specify a user function that will handle the display of a window. Let’s use this to change the colour of a window:

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

#include<GL/gl.h>

void display() {

    glClearColor(1,0,0,0);

    glClear(GL_COLOR_BUFFER_BIT);

    glFlush();

}

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

    glutInit(&argc, argv);

    glutInitWindowPosition(100,100);

    glutInitWindowSize(500,500);

    glutCreateWindow(□Hello World□);

    glutDisplayFunc(display);

    glutMainLoop();
```

```
}
```

Save it as basic2.c, compile and run it as in the earlier example.

The three GL functions we used to colour the window background are:

glClearColor(R, G, B, alpha): This is used to set the colour for a window. The numbers you can use for each colour are between 0 and 1; you can use float numbers like 0.1, 0.11, etc.

glClear(GL_COLOR_BUFFER_BIT): This clears the screen to the desired colour set by glClearColor.

glFlush() executes the commands to the screen rather than storing it in a buffer.

In the next part of the series, I plan to explore how to draw lines and triangles, and use keyboard inputs.

Source : http://www.opensourceforu.com/2013/04/graphics-programming-in-linux/