B.TECH PART-3, SEMESTER-5

Graphics Project

Rotating Globe

Submitted To: Prof. S. K. Singh

2011

Team members:
Amit Kumar Baranwal
Jahnavi Singhal
Kumar Ankit
Vaibhav Ahlwahat

College Name: Institute of Technology, BHU

IT-BHU, VARANASI

TABLE OF CONTENTS

1. Objective:

The project aims at developing a rotating globe using the OpenGL graphics library.

The globe rotates about an axis passing through its centre with its rotation controlled by specific keys.

The globe is basically a sphere with an image of map texture-mapped on it.

2. OpenGL- A brief account

OpenGL (Open Graphics Library) is a standard specification defining a cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. The interface consists of over 250 different function calls which can be used to draw complex three-dimensional scenes from simple primitives.

OpenGL serves two main purposes, to:

1. hide complexities of interfacing with different 3D accelerators by presenting a single, uniform interface
2. hide differing capabilities of hardware platforms by requiring support of the full OpenGL feature set for all implementations (using software emulation if necessary)

We can serve our purpose by using various functions from OpenGL API documentation.

3. Development

The project has been developed using the C++ programming language with the use of the OpenGL graphics library. The tasks such as creating and rendering the sphere, texture mapping it with the image of Earth's map, getting it to rotate and controlling its rotation using specific keyboard keys have been achieved through the functions provided in the OpenGL API.

A detailed description of each of the major tasks and how it has been implemented in the code follows.

A) Drawing the sphere:

The OpenGL library provides a predefined function to draw a sphere.

Function:
void **gluSphere**(GLUquadricObj **qobj*,GLdouble *radius*,GLint *slices*,Glint *stacks*);

Parameters:
qobj: Specifies the quadrics object (created with **gluNewQuadric**).
radius: Specifies the radius of the sphere.
slices: Specifies the number of subdivisions around the *z* axis (similar to lines of longitude).
stacks: Specifies the number of subdivisions along the *z* axis (similar to lines of latitude).

B) Texture mapping the image onto the sphere:

First of all we have defined a structure named *GeImageData* which holds the basic image data namely width, height and a pointer to hold an instance of the image. Then we have used a standard function in the library to define the texture image data as well as its representation in the memory. The pixel data has been used in standard *GL_RGBA* format.

Function:
void glTexImage2D(GLenum *target*,GLint *level*,GLint *components*, GLsizei *width*, GLsizei *height*,GLint *border*,GLenum *format*,GLenum *type*,const GLvoid **pixels*)

Parameters:
target: Specifies the target texture. Must be GL_TEXTURE_2D.
level: Specifies the level-of-detail number.
components: Specifies the number of color components in the texture.
Must be 1, 2, 3, or 4.
width: Specifies the width of the texture image.
height: Specifies the height of the texture image.
border: Specifies the width of the border. Must be either 0 or 1.
format: Specifies the format of the pixel data. The following symbolic values are accepted: GL_COLOR_INDEX, GL_RED,

GL_GREEN,GL_BLUE,GL_ALPHA,GL_RGB,GL_RGBA, GL_LUMINANCE, and GL_LUMINANCE_ALPHA.
type: Specifies the data type of the pixel data. The following symbolic values are *accepted*: GL_UNSIGNED_BYTE, GL_BYTE,GL_BITMAP, GL_UNSIGNED_SHORT, GL_SHORT, GL_UNSIGNED_INT,GL_INT, and GL_FLOAT.
pixels: Specifies a pointer to the image data in memory.

Now as we have used rgb images, we have defined a function *ge_read_rgb()* to manipulate the image pixels in the memory to render it suitable for texture mapping.

C) Making the globe rotate:

To achieve rotation we have decided to toggle between two functions.
1. for displaying the image: displayFunc() and
2. for keeping the display idle: idleFunc().

In each subsequent call we are incrementing the angle parameter of: glRotatef(). glRotatef() basically achieves vector rotation of the specified vectors by the specified angle. Now, we are making subsequent calls to this function with incremented angle parameter when in displayFunc() and then keeping it idle. Thus rotation is achieved. A brief account of the glRotatef() function is below.

Function: void **glRotatef**(GLfloat *angle*, GLfloat *x*, GLfloat *y*, GLfloat *z*)

Parameters: *x*, *y*, *z* - Specify the *x*, *y*, and *z* coordinates of a vector, respectively.

D) Implementing the Keyboard control for rotation:

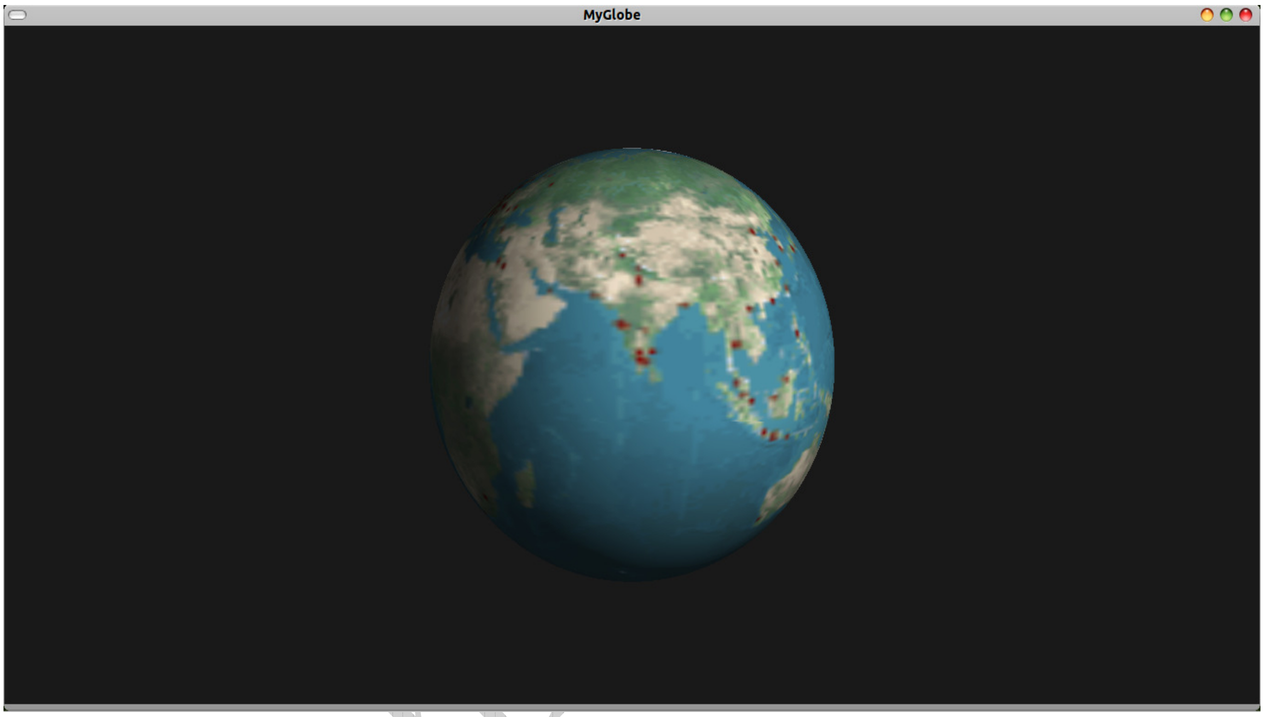
The keyboard control has been achieved by changing the angle parameter of the function: glRotatef() on certain keys being pressed. The following keycontrol has been added to the rotating globe:

KEY	EFFECT
+/=	increments the angle
-/_	decrements the angle
Q/q	exits the application.

The keyboard control has been implemented via a user defined function named keyboardFunc() which uses switch-case block to assign the tasks to the pressed keys.

4. Output

The image below is a still for the rotating globe developed:



5. Platform and IDE:

The project has been developed and tested on Microsoft Windows as well as Ubuntu (a Linux Distro). On windows, the IDEs on which it has been tested are CodeBlocks and DevCpp.

6. Conclusion:

The rotating globe has been developed successfully as well as tested using OpenGL library. Looking into the scope of future development, mouse control in addition to the keyboard control can also be added to enhance the user interactivity. Furthermore, we have only texture mapped *.rgb* images, support for texture mapping *.jpeg* and other image formats can also be added.