ICS 280 F02: Computer Graphics
Programming Assignment 3

Gopi Meenakshisundaram

Assigned: Oct 15, 2002
Due Date: Oct 22, 2002 11:59pm.

PROJECT GOAL: Write a restricted OpenGL library.
The goal of the project is do to *compute(not perform)* all the transformation matrices with your library instead of using OpenGL library. Since you are relying on OpenGL for final rasterization, you have to let OpenGL know what the final transformation matrix is, so that it transforms the vertices and triangles and does the final rasterization. Pay attention to software modularity. Read the assignment completely and design your code first before starting on your project. (For example, have a 'stack' class, that can be used for to define a 'matrix stack', which in turn can be instantiated twice – one for model view matrix and another for projection matrix.)

I. Implement two stacks of matrices according to the rules given later in the assignment, and the following functions to operate on the stack of matrices. These two stacks will play the role of what OpenGL calls the ModelView stack and Projection stack.

void I_my_glLoadIdentity(void)
void I_my_glPushMatrix(void)
void I_my_glPopMatrix(void)

void I_my_glLoadMatrixf(const GLfloat *m)
void I_my_glLoadMatrixd(const GLdouble *m)

void I_my_glTranslated(GLdouble x, GLdouble y, GLdouble z)
void I_my_glTranslatef(GLfloat x, GLfloat y, GLfloat z);

void I_my_glRotated(Gldouble angle, GLdouble x, GLdouble y, GLdouble z)
void I_my_glRotatef(Glfloat angle, GLfloat x, GLfloat y, GLfloat z);

void I_my_glScaled(GLdouble x, GLdouble y, GLdouble z)
void I_my_glScalef(GLfloat x, GLfloat y, GLfloat z);

void I_my_glGetMatrixf(const GLfloat *m)
void I_my_glGetMatrixd(const Gldouble *m)

```
void I_my_gluLookAt(Gldouble eyeX, GLdouble eyeY, GLdouble eyeZ,
                 GLdouble centerX, GLdouble centerY,
                 GLdouble centerZ,
                 GLdouble upX, GLdouble upY, GLdouble upZ )
void I_my_glFrustum(GLdouble left, GLdouble right,
                 GLdouble bottom, GLdouble top,
                 GLdouble zNear,   GLdouble zFar )
```

```
void I_my_gluPerspective( GLdouble fovy,   GLdouble aspect,
                          GLdouble zNear, GLdouble zFar )
```
(NOTE: "glu" instead of "gl" for LookAt and Perspective)

II. The above functions operate on the "current stack". The "current stack" is determined by
```
void I_my_glMatrixMode( Glenum mode );
```
This function will allow you to switch between the two matrix modes, `GL_MODELVIEW` and `GL_PROJECTION`.

III. The "I_my_gl" functions are "internal" functions. You should not call them directly from your application code. You have to use wrapper functions (just "my_gl") to call these internal functions. All wrapper functions perform three operations. (a) Call the appropriate "internal function". (b) Get the final matrices from both the stacks using I_my_glGetMatrix function. (c) Use glLoadMatrix (actual OpenGL) function to load these matrices into appropriate OpenGL stacks.

IV. Write a .h file with a few of "#define"s. For example,
#define glLoadIdentity my_glLoadIdentity
Do this for every function you have implemented. As you have not written a substitute for *all* OpenGL functions, I think you have to do this on a function-by-function basis instead of taking a common approach.

V. Use your second assignment to finish this assignment. Your (second assignment) program should call your wrapper functions instead of the OpenGL functions. To achieve this, just include the .h file you have written in every program file.

Rules:

1. Allow a maximum of 16 matrices to be pushed.
2. Report error if the stack is empty when a my_glPopMatrix function is called and continue.
3. Report error if the stack is full (16 elements) when a my_glPushMatrix is called and continue.
4. *m* is a pointer to the array of 16 consecutive values (linear array) of the matrix (4x4 matrix) in *column major order*.
5. Implement a "static" array of matrices so that consecutive calls to your matrix manipulation routines will be accumulated.
6. Include gl.h to make use of the data types GLfloat* and Gldouble*.
7. The meaning of each of the above functions (except my_glGetMatrix*) takes the same semantic meaning as the functions in OpenGL library (without "my_"). Use man pages available on Sun machines to find the semantics of these functions.
8. my_glGetMatrix* function returns through *m* the matrix you have on the top of the stack.
9. All internal computation of composition of matrices should use GLdouble.
10. Use I_my_glFrustum to perform I_my_gluPerspective. Do not directly use the matrix given in the man pages to implement I_my_glFrustum. Divide this into stages (like scale, shear, projection etc.) and implement this function as composition of these stages.