

CS535: Assignment #1 – Wireframe Renderer

Out: September 6, 2005

Back/Due: September 13, 2005

Objective:

This focus of this assignment is to obtain a good understanding of 3D transformations including translations, rotations, and perspective projections. You have one week and I recommend you start **today**. You will write a program that draws 3D objects to the screen using wireframe rendering and gives the user basic 3D navigation tools to move the camera within the scene and to move the objects within the scene.

Summary:

The assignment is to implement a program which renders a 3D object within a box-shaped area. Using either a text-based or GLUI-based interface, allow the user to choose the size of the box (x-, y-, and z-dimensions), the object filename, the scale of the object (s), the object position (o_x, o_y, o_z), the object rotation (r_x, r_y, r_z), the field of view in degrees for a given near distance (FOV and n), the far distance (f), the viewing position (v_x, v_y, v_z), and the viewing direction (d_x, d_y, d_z). After any change to any of these parameters, the scene should be redrawn.

Specifics:

- (0) Object Representation: each object should consist of a list of vertices and a list of connected edges; thus any polygonal object can be used. You must provide files for at least a cube, pyramid, and an approximation of a cylinder. The file format to use for **all** objects is:

- a. n
- b. $v_{x0} v_{y0} v_{z0}$
- c. $v_{x1} v_{y1} v_{z1}$
- d. ...
- e. $v_{x(n-1)} v_{y(n-1)} v_{z(n-1)}$
- f. e
- g. $v_{a0} v_{b0}$
- h. $v_{a1} v_{b1}$
- i. ...
- j. $v_{a(e-1)} v_{b(e-1)}$

where n = number of vertices, e = number of edges, v_{x0} is the x-coordinate of the first vertex, v_{a0} and v_{b0} are the indices of the vertices of the first edge, etc.; all indices start at 0.

(comment lines will be preceded by a '#' character)

- (1) User Interface: a text-based or GLUI-based user interface should be used to allow the user to specify the dimensions of the "world box" (which should be centered around the origin), object file name, s, (o_x, o_y, o_z), (r_x, r_y, r_z), f, (v_x, v_y, v_z), (d_x, d_y, d_z). Reasonable initial values should be given to all these parameters at program initialization. The default viewing position should be (0,0,0) and default viewing direction should be (0,0,-1). A reasonable field of view is 60 degrees. By default,

- the object should be placed in front of camera (e.g, down the $-z$ axis) and the scale factor should be $s=1$.
- (2) Transformations: you will have to write a library and/or object-classes that support point, vector, and matrix manipulations (addition, subtraction, multiplication, dot product and cross product) as well as 3D transformations including rotation, scaling, translation, and 3D->2D perspective projection.
 - (3) Object transformations: each vertex of the object should be transformed according to the translation (o_x, o_y, o_z) and the rotation about each axis (r_x, r_y, r_z) . The rotation should **always** happen around the mid-point of the object. You will need to compute the mid-point of the object (e.g., average of all the vertices) and rotate the object about that point. This means translating the middle of the object to origin, rotating about x, y, and z, and then translating the object back to its position in space.
 - (4) Camera transformations: the camera (“eye”) should be located at $v=(v_x, v_y, v_z)$ and looking along the direction $d=(d_x, d_y, d_z)$. Assume the viewing up vector is $u=(0,1,0)$. Using these values, you can construct the orthogonal basis vectors for the viewing setup: $a = (d \times u) / \|d \times u\|$, $b = (a \times d) / \|a \times d\|$, $c = (a \times b) / \|a \times b\|$. The matrix $M=[a|b|c]$ transforms from viewing space to canonical space (e.g., $a = x$ -axis, $b = y$ -axis, and $c = z$ -axis). The inverse of M (i.e., M^T for orthogonal basis) goes from canonical space to viewing space.
 - (5) Rendering: using perspective projection, you will have to transform the 3D vertices of the object to screen-space. This transformation will depend on all the aforementioned parameters and on the window size. It is best to think of this piecewise: one transformation matrix for the object, one for the camera, and one for projection. The complete transformation is simply the product of these matrices (remember: order **does** matter). After object vertices are transformed from 3D to 2D, you can render each line in an analogous fashion to Assignment #0, namely draw all pixels between the endpoints of each line to your framebuffer and at the end of frame rendering copy the framebuffer to OpenGL (as in Assignment #0). Also draw the outline of world box in wireframe.
 - (6) For this assignment, you may **not** use OpenGL/GLUT functions or any other downloaded libraries for geometric calculations, manipulations, rendering, or any part of this assignment except as indicated in the template of Assignment #0. You must implement yourself all necessary routines.

GLUI: (optional use in this assignment)

GLUI is a platform independent user-interface builder. You do not need to use this library but it can embellish your program very easily. There are other similar libraries out there but I recommend this one for its simplicity yet completeness. You must install this library. More information about GLUI can be found at <http://www.cs.unc.edu/~rademach/glui>.

Grading:

Your program will be tested against the three object files you provide and additional object files we provide. We will use the interface to alter the aforementioned parameters and expect to see the correct behavior. If the interface does not allow a certain parameter

to be changed, it will be considered not implemented at all. If the program does not compile, zero points will be given.

To give in the assignment, email the TA, Denny Wong (wang124@purdue.edu), your complete project (project files, source code and precompiled executable) by 4:30pm on the due date. It is your responsibility to make sure the email is delivered/dated before it is due. If you wish to receive confirmation of receipt, please ask the TA by email. If you email at 4:29:59pm on the due date and ask for confirmation and the confirmation is negative -- it will be considered late. **Hint: don't wait until the last moment to hand in the assignment.**

IMPORTANT: in your email message, please include a short message saying "This is Assignment #X" and attach your assignment inside a .ZIP file -- do not email a .EXE file as spam filters will probably filter-out the attachment.

If you have more questions, please see myself or the TA.