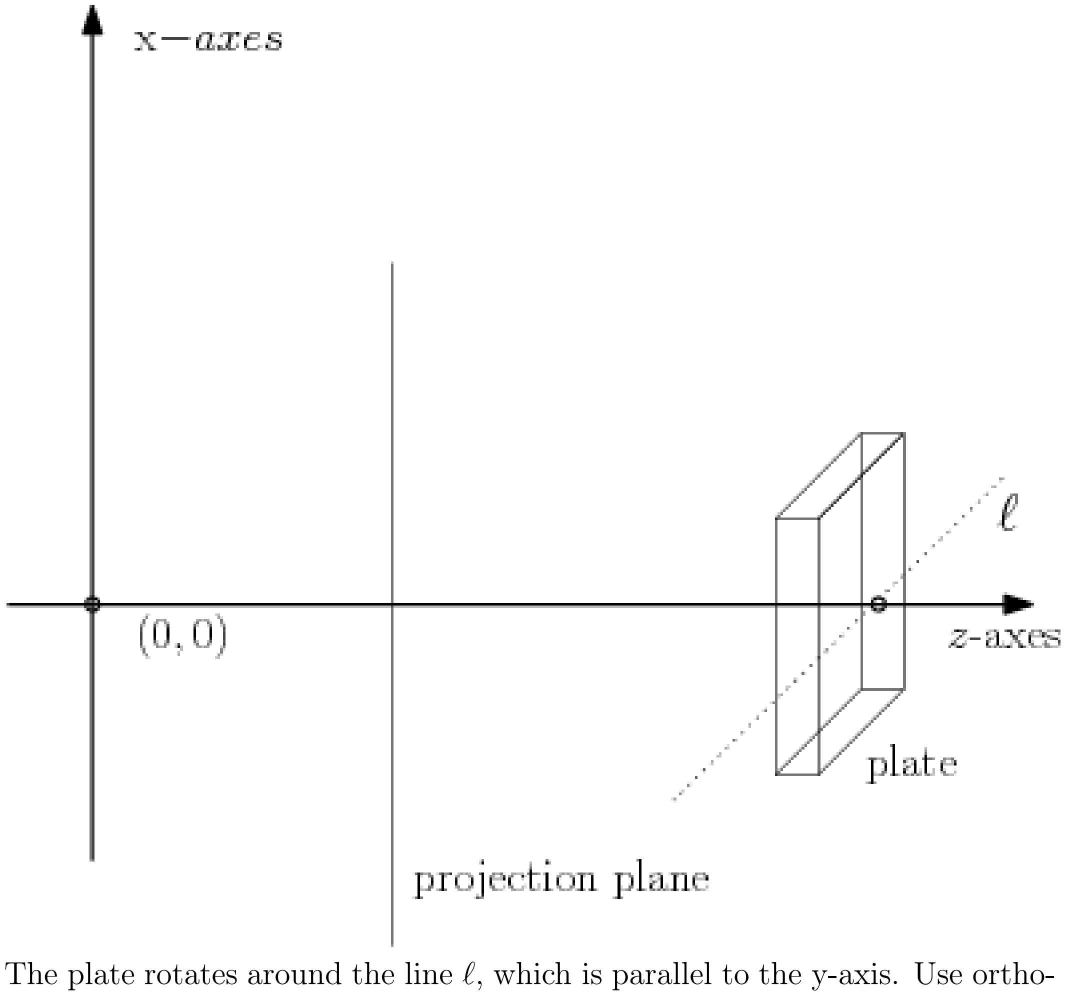
HOMEWORK #1

Abstract. This exercise explores a fundamental process in computer graphics: how we construct 2D images, consisting of discrete pixels, from a 3D scene. For this assignment, you'll to write a program that animates a plate rotating around a line in 3D space. The **plate** is a very thin square, (e.g. a thin book) and has a picture textured on each of its two sides. These pictures should be snapshots of each student submitting the project. The plate revolves around the line (see details below) at a constant angular velocity (RPM), finishing a full cycle every few seconds. This assignment may be done in teams of two. *Due September 25, 2008 by 11:59PM.*



(1)
(2) The plate rotates around the line ℓ, which is parallel to the y-axis. Use orthographic projection to view the pictures "printed" on the plate, and pick the size of the plate such that when orthogonal to the z-axis, it will occupy most of the window.

(3) Animate your plate so that it rotates around the line ℓ over time. Try to generate smooth motion by incrementing the plate's orientation slightly on each consecutive animation frame (i.e. the angle it creates with the z-axis). Recompute the projection and rasterization (i.e. conversion to pixels) after each increment.

(4) For this assignment, you're being asked to construct a simple graphics "pipeline" that maps your rotating plate in 3D space into a series of 2D pictures. As such, you may *not* use OpenGL's built-in 3D functionality, which includes, but is not limited to, texture mapping, built-in camera manipulation, and Z-buffer. Use OpenGL only to paint pixels into the 2D window (i.e. the screen "canvas"); the output of your graphics pipeline will be the color of each pixel on this canvas. An easy way to setup the drawing canvas is to initialize GL with the command *glOrtho2D(0,width,height,0)*, where the width and height parameters for the canvas are equal to the size of your GLUT window. Then, you can color the individual pixels of that window using the command *glVertex2i*. If you have questions regarding whether or not a particular GL function is "out-of-bounds," please email your TA *before* using it!

(5) The course assignments webpage will have a link to a demo program ("paint-demo") which constructs a simple GLUT window and paints pixels into it using the 2D canvas method described above. It also includes example C code for decompressing a JPEG file into a matrix of pixels (using LibJPEG), and demonstrates how to manipulate that matrix. You are free to reuse any/all of this code in your assignment without citation.

(6) Hints: You may assume that the "camera" is always in a fixed position and orientation, i.e. at the origin of the world coordinate system, and pointing down the positive z-axis. Furthermore, you can disregard the effects of lighting, and assume that the scene is pre-lit with only ambient light.

(7) Your submission should contain all the code and build files needed for the grader to easily recompile and run your program on the machines in the graphics lab (GS 930). Assume that, in addition to the standard C libraries, the *only* libraries available by default on the test machine are OpenGL (GL, GLU, and GLUT) and LibJPEG. If your program uses libraries in addition to or in exception of these, you should provide those libraries with your submission. Further, make the necessary adjustments to your build files so that your project will still compile transparently. Finally, your submision should include the face images (e.g. JPEGs) that you're using to texture the polygon and a readme file with the following info: names of project group member(s), known errata, acknowledgements for any external code/libraries used, and a summary of any additional features that you might want considered for (modest) extra credit.

(8) Submit your code using the turnin function on lectura; the turnin name is cs433_hw1. Due September 25, 2008 by 11:59PM.

(9) Please check the course newsgroup regularly for updates and/or clarifications to this spec.