Introduction to Computer Graphics

Assignment Three

October 14, 2003

U-grad due date: Thursday, November 13, 2003, 11:59 PM
Grad Student due date: Thursday, November 6, 2003, 11:59 PM
(U-grads doing project should shoot for Grad student due date)

Credit (U-grad): Approximately 15 points (Relative, and roughly absolute weighting)
Credit (Grad): Approximately 12 points (Relative, and roughly absolute weighting)

This assignment may be done in pairs if you prefer.

Hint. Read the assignment *carefully*, and think it through before starting to code.

In this assignment we will improve "parallelepiped -world". The same rules regarding input and output and program exit from the previous two assignments apply.

The first step is to incorporate visibility. Use the facility for rendering 3D orthogonal scenes, and apply it to the orthogonalized version of your world. Check that you get the same image for the same cube as in the second assignment. You are allowed to tell OpenGL that the world is 3D orthogonal, but not perspective. You are to handle the perspective part. You should implement back-face culling as in assignment two.

The first improvement noticeable to the user is to allow multiple objects. Your program should handle multiple "pp" commands in sequence, each one creating an parallelepiped.

You can use whatever scheme you like for the default coloring of the parallelepipeds (except black). Possibly the easiest is to use the same one as you did in assignment two.

In the following , you can use the naïve version of color introduced in class (diagonal model). Also, R,G, or B values that are greater than 255 should be set to 255. OpenGL probably does this for you. (This is called clipping—yes, another use of the word).

Implement the following command to add ambient light:

ambient <r> <g> <b>

(r,g,b) is the color of a perfect white diffuse reflector.

Implement a command to add a single point source. The command should be:

light <x> <y> <z> <r> <g> <b>

The (x,y,z) give the direction. This is over specified (why?), but easy for the user (and for assignment 4). The (r,g,b) tells you the color of a perfect white diffuse reflector when the light is perpendicular to it. You will need to implement Lambertian shading for every surface that you draw.

On program startup the last box created is the selected object. Rotations, stretches, and translations, apply to the selected object. Provide some form of feedback so that the user knows which object is selected.

The user should be able to change the selected object with a single click with the left button while the pointer is on the object to be selected. This "picking" function must be implemented yourselves, and will be used for assignment 4. Once an object is selected, the interaction should be as in assignment 2.

Give the user a way to add a cube through the menu. The cube can simply be a default cube which is moved and stretched and rotated as the user likes. When a cube is added through the menu, it becomes the selected parallelepiped.

Extra credit

If you would like to improve on the program, be sure to explain what you did in the README file, and it will be considered for modest extra credit.

Deliverables

You must electronically submit a README containing any relevant information, but at a minimum, your name; an executable (called a3); and a src directory containing source files and a Makefile which can be used to build the executable.

The program must compile and run on one of the graphics machines (gr01, ... , gr10). Put in the README file the machine which you have verified this on.

Note that the graphics machines can be booted into Windows by people in the lab, so that it is possible that if you are working remotely that you will need to try more than one. We encourage students to use the higher numbered machines for Windows (7 through 10), but this cannot be enforced.

The turnin name is cs433_hw3.