Introduction to Computer Graphics

Assignment Three

October 15, 2002

Due: Thursday, November 7, 2002, 11:59 PM

Credit: 15 points (Relative, and roughly absolute weighting)

This assignment may be done in pairs if you prefer.

There are two options for this assignment, with the second option having two sub-alternatives. One is to do the assignment described below. The second option is to propose a project, which is either a large two part project (with part one being due on Nov 7), or small project (which could be extended). A possible instantiation of this would be to take on one of several small research projects that I can provide. This likely has most appeal if there is some chance that you would like to continue on with the work in some context, but this is most definitely not required, and the projects that I have in mind are small enough that a contribution can be made in the time allotted.

IMPORTANT—if you wish to do a project, I need at least an informal proposal soon, and at the very latest, by October 24. Otherwise we will be expecting the default assignment.

Default Assignment

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

The first step is to incorporate visibility. You can have OpenGL do the work here. Probably the easiest way to build on assignment two is to 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.

Regardless of your approach, I would like you to keep the representation of the (transformed) block as a collection of planes, and implement the shading, the back-face

culling, and the picking yourself. (At this stage, it is likely easiest to this anyway, and doing so will make the next assignment easier).

The first noticeable improvement is to allow multiple objects, and handle multiple box commands in sequence. Now the xrot, yrot, and zrot commands apply to the most recently mentioned box. Also implement xscale, yscale, and zscale, so now we can stretch a rotated box. The argument to xscale, yscale, and zscale can be an integer designating an amount out of 100.

You can use whatever scheme you like for the default coloring of the boxes (except all black).

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 point sources. 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. 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. You will need to add up the contributions of all the lights.

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. Once selected, the interaction should be as in assignment 2, except that you should assume the more adventurous interpretation of the ambiguity regarding scaling, and therefore provide for scaling the box in arbitrary position (so it can become something other than a box).

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

Provide a way to select a particular face, and then set the color of that face from a menu of options. Further provide the option to add some specularity which is added to the Lambertian shading color using the Phong model. You can make this as fancy as you like (extra points for nice implementations), but you should provide at least two levels of specularity strength, and two levels of specular sharpness (controlled through the

exponent). If you choose to provide minimal capability, you need to make sure that the choices given provide noticeable and varied effects. Since I am purposely leaving this part of the interface up to your imagination, be sure to put the user incantations in the README file.

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 turnin name is cs433_hw3.