Project 2
CSCI-4967: Three-Dimensional Computer Graphics

Due: Friday, October 15, 2004, 11:59:59pm

1 Overview

This project is intended to familiarize you with hierarchical 3D modeling transformations and lighting in OpenGL. You are to write a program to display an articulated humanoid created from ellipsoids, and select appropriate lighting and material properties for your scene. You will use both OpenGL and GLUT functions for modeling the humanoid and lighting the scene, and use mouse and keyboard input events. You will also need to set up the viewing and projection transformations.

Please see the course web page at `www.cs.rpi.edu/~sakella/graphics/` for project updates and additional information.

2 Project Tasks

You are to write an OpenGL program to draw an articulated humanoid, affectionately known as "The Egghead". The humanoid (see Figure 1) has a head and a neck, a torso, two shoulders, two arms consisting of an upper and lower arm each, a hip, and two legs consisting of an upper leg and lower leg each. Each body part except the hip can rotate with one degree of freedom. So any configuration of the Egghead is specified by thirteen angles. You can use `glutSolidSphere()` to draw a sphere. You will need to draw the body parts as scaled spheres with transformations specified as in the figure.

1. Draw the humanoid with its hip centered at the origin (0, 0, 0). Select the initial camera location to be at (0, 0, 16), and pointing down the negative Z axis. Let the default projection mode be perspective projection with a field of view of 45 degrees.
2. The dimensions of each of the body parts of the humanoid are specified in the file `proj2_support.h` that is available on the course web page. This file also has code for loading in a simple animation file.
3. In the default initial configuration with all angles set to zero, the humanoid is in the "touch-down" position (Figure 1). You may find it helpful to draw the local XYZ coordinate axes fixed to each part. All the local (part) coordinate frames are specified to be parallel to the hip coordinate frame at the origin in the initial configuration. (See the image on the course web site for a depiction of the coordinate frames.)

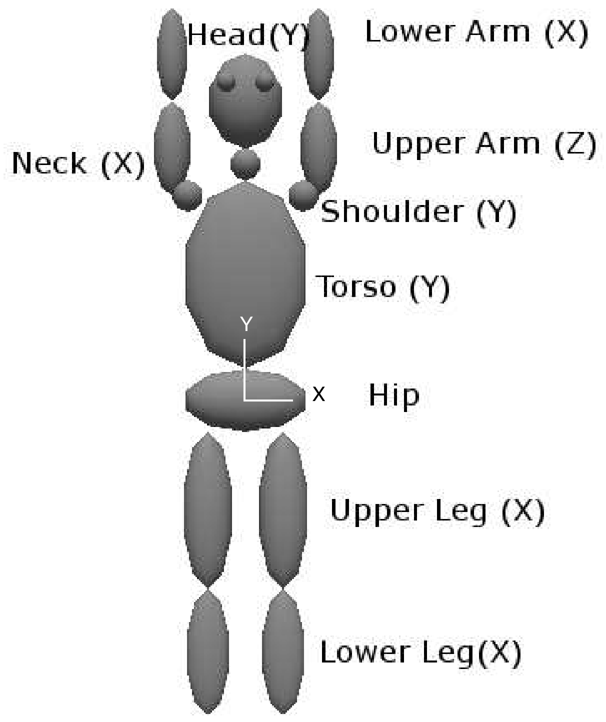

Figure 1: A humanoid in its initial configuration with all angles set to zero. The XYZ coordinate frame for the hip is shown, with the positive Z axis pointing out of the page. The local coordinate frames for all other parts of the humanoid are parallel to this coordinate frame in this initial configuration. The rotation axis for each part, in its local coordinate frame, is indicated in parentheses.

Each (scaled) part is translated as shown in the initial configuration in Figure 1, which is drawn to scale. Each body part rotates about a rotation axis specified in its local coordinate frame (Figure 1). The local coordinate frame depends on the hierarchy of transformations from the hip. You may wish to refer to Chapter 3 of the OpenGL red book for examples of such transformations.

4. Use lighting and select material properties to illuminate the scene to show off the Egghead. Use different colors to help distinguish adjacent parts of the humanoid.

5. To enable user interaction using both the mouse and keyboard, write functions to rotate the individual joints of the humanoid, change the viewpoint around the humanoid, and to translate away from and towards the humanoid.

   The main keyboard events and the corresponding desired behaviors are:

   - To enable manual control of the individual joints, provide a specified key control for each joint. A detailed listing of the key mappings for controlling the individual joints is provided on the course web site. For example, if `a` and `A` specify the motion of the upper arm of the humanoid, `a` should rotate the upper arm counterclockwise and `A` should rotate it clockwise. These keys will be helpful when you create your own animations. The joint rotation amount at each key-press should be 10 degrees.

- `i` key: The camera and objects are returned to the initial configuration.
- `l` key: Loads animation file.
- `o` key: Output all current angles to `stdout`. You will need this to take keyframe snapshots of the humanoid configuration (i.e., the set of joint angles that specify the orientations of the humanoid) for animations. These angles must be output in the order: head, neck, torso, left-shoulder, left-upper-arm, left-lower-arm, right-shoulder, right-upper-arm, right-lower-arm, left-upper-leg, left-lower-leg, right-upper-leg, right-lower-leg. (Note: By "left-shoulder" we mean the shoulder seen on the left by the viewer in the initial configuration. Similarly for other parts.)
- `p/P` key: Starts and pauses the animation, assuming an animation file has been loaded.
- `j` key: To quit ("eJect from") the program.

The mouse events and the corresponding desired behaviors are:

- With the left mouse button pressed, a horizontal motion of the mouse should rotate the humanoid about its Y axis, and a vertical motion of the mouse should rotate the humanoid about its X axis (or the global X axis). You will need the `glutMouseFunc()` and `glutMotionFunc()` functions to be able to do this.
- With the right mouse button pressed, vertical motion of the mouse should zoom closer to and further away from the humanoid.

6. A simple animation file `simple_anim.txt` that has the angles for each joint at a discrete set of keyframes is provided to you on the course web site. Your program should be able to read in the joint angle data, and animate the humanoid accordingly. There are thirteen joint angles to specify a configuration of the humanoid, and each line of the animation file will specify these angles in the order: head, neck, torso, left-shoulder, left-upper-arm, left-lower-arm, right-shoulder, right-upper-arm, right-lower-arm, left-upper-leg, left-lower-leg, right-upper-leg, right-lower-leg. Each line is a keyframe of the animation and should be repeated for five frames of the animation. The animation should keep looping until it is paused. (Note: By "left-shoulder" we mean the shoulder seen on the left by the viewer in the initial configuration. Similarly for other parts.)

7. Finally, you should create a valid animation file of your own in the same format, and submit it. The name of this file should be `animate.txt` and should have at least twenty keyframes.

You may add objects to the scene or otherwise enhance it to make it more realistic.

3 Grading

Your project will be graded for a total of 100 points as follows:

- 25 points for modeling the humanoid.
- 15 points for lighting and material properties for the humanoid.
- 25 points for implementing joint control.
- 10 points for viewpoint control
- 10 points for supporting animation of the humanoid (`l`, `o`, `p` keys).
- 5 points for your animation file of the humanoid.
- 5 points for code structure, functionality, and documentation.
- 5 points for any special features or creative enhancements.

Describe any enhancements or special features in your README file, and ensure that they do not interfere with the required project functionality.
Lateness policy: Please read the lateness policy in the course syllabus. Use your late days carefully, if at all.

4 Submission

The code must be submitted no later than **11:59:59 pm on October 15, 2004**. You will follow the same submission procedure as for Project 1. Specific instructions for handing in your code are provided on the course web page. **You are responsible for ensuring that your code can compile and run on the PCs in VCC South to get project credit.** Your source code must be readable and commented. The comments need not be extremely long, just explain clearly the purpose of each block of code.

You must submit a single zip file containing your README file, your `animate.txt` file, your source code (source and header files), and if necessary, a Makefile to compile it. Your submission should NOT include any object files or executable files. Every file (except the `animate.txt` file) should have your name in a comment line at the top. The README file should contain the following information: your name, email address, instructions on how to compile the code and run it, known bugs or limitations, any extra credit enhancements, and any other relevant information.

Proper submission is entirely **your responsibility**. Contact the TA if you have any doubts whatsoever about your submission. Do **NOT** submit your project via email. Be sure to keep copies of your files and **do not change them after submitting**. After grades are posted, you have exactly one week to resolve all problems. All grades are final two weeks after they are posted.

A project that does not follow the submission guidelines will receive a **10 point deduction**. Please observe the academic integrity guidelines for the course as described in the course syllabus.