CPSC 314 Assignment 1

Term: September 2007, Instructor: Wolfgang Heidrich, heidrich@cs.ubc.ca, http://www.ugrad.cs.ubc.ca/~cs314

Due: Programming portion: Oct 4, 5pm; Theory: Oct 4 in class

Assignment 1.1: OpenGL Transformations (5 Points)

What is the 4×4 transformation matrix after the following series of OpenGL commands? Show your work!

```
glLoadIdentity();
glRotatef( 90.0, 1.0, 0.0, 0.0 );
glTranslatef( 1.0, 2.0, 0.0 );
glPushMatrix();
glScalef( 2.0, 1.0, 3.0 );
glPushMatrix();
glRotatef( 45.0, 0.0, 1.0, 0.0 );
glTranslatef( 1.0, 1.0, 1.0 );
glPopMatrix();
glScalef( 1.0, 1.0, 2.0 );
```

Assignment 1.2: Viewing Transformations (5 Points)

Develop a 4×4 transformation matrix that can be used as the viewing transformation, with the following parameters:

- the camera is located at $(1, 2, 2)^T$
- the camera is pointed at the direction $(0, 1, 0)^T$
- the up-vector, which will be mapped to the positive y direction on the image, is the direction $(0, 0, 1)^T$

Show your work!

Assignment 1.3: Programming: Spiders (25 Points)

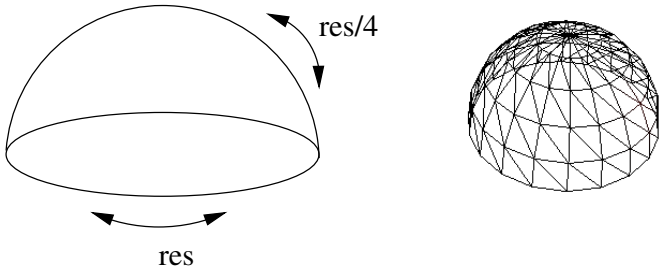
Your goal is to create and animate one or more spiders using a hierarchy of parts. A sample implementation is provided on the web page for reference. The best animations will be nominated for CPSC 314 hall-of-fame. The following is a suggested ordering of steps.

0 points Download and compile the template code: http://www.ugrad.cs.ubc.ca/ cs314/current/A1-Code.tar.gz

When running the program,

- Type 'a' to see a very simple arm animation.
- Type 'k' several times to see the keyframes that are being used to create the animation. The keyframes are stored in the file keyframe.txt.
- Type 'q' to quit.

5 points Create a function hemiSphere(r,res) that draws a hemisphere (i.e. half sphere) as a function of the 2 parameters: the radius of the sphere and the resolution. The resolution res is the number of polygons in which you subdivide the circumference of the sphere. You should use res/4 polygons from the equator to the pole, as depicted below. Use a convenient modeling coordinate system to represent the hemisphere. You will be using it as a modeling primitive for some of the body parts of your spider. For this part of the question, just add a single hemisphere primitive to the scene.



You will be computing all the vertex locations. First use GL_LINE_LOOP in order to draw the hemisphere in wire frame. Once this works, compute a surface normal for each vertex and call glNormal3f() before each glVertex3f() call in order to assign a correct surface normal to each vertex, and change to using GL_POLYGON, GL_QUADS, or GL_QUAD_STRIP, thus producing a solid-shaded hemisphere. You may use GL_TRIANGLES or GL_TRIANGLE_STRIP at the pole, or you can just co-locate two vertices in a quad representation.

6 points Build your articulated figure. You should make use of your hemisphere at least once in your character. Use an appropriate hierarchy of transformations. Implement it using appropriately structured code. You may want to define separate geometry for each type of link, or you may want to use scaled versions of the various glutSolidX() primitives, where X can be any of Sphere, Cube, Cone, Torus, or Teapot. Draw your spider in an appropriate "rest pose" — typically this is a simple standing pose. Make your spider as colorful as you like.

5 points Add the ability to animate your spider figure by making use of Params[n] variable, where n=2,3, ... to animate the position and joint angles of your spider. These are animated according to the keyframe data in the keyframe.txt

file. Note that the first parameter for each keyframe, Param[1], is reserved for modeling the relative time between keyframes (in seconds).

Each line of keyframe data will specify all of the "degrees of freedom of the character". The reference solution that you can download shows an example of how you might do this, although I recommend that you come up with your own set of degrees-of-freedom that work for your particular figure and the animation that you have in mind. You could also use this to animate camera parameters, color parameters, or the movement of other objects. Lastly, you could also use additional keystroke bindings to interactively animate parts of your scene. Any such changes need to be documented in a file README.txt

keyframe 0.4 2.0 1.0 10 20 10 10 -20 10 10 40 50 -40 50 10 20

where:

```
parameter 1:  time stamp (ignore this for now)
parameter 2:  body location, x
parameter 3:  body location, y
parameter 4:  body lean
parameter 5:  first left leg hip angle
parameter 6:  first left leg knee angle
parameter 7:  second left leg hip angle
parameter 8:  second left leg knee angle
...
parameter 20:  fourth right leg knee angle
parameter 21:  tail segment left/right bend
parameter 22:  tail segment up/down bend
parameter 23:  head roll
```

9 points Create an interesting animation for your character. Convert it to an MPEG animation file by dumping PPM images from your application and then using ffmpeg to compile all the individual frames into a single MPEG animation file.

Because you will need to temporarily create a large number of images that will consume a fair bit of disk space, create a temporary directory /tmp/foo where foo is your user name. Edit the appropriate line in the dumpPPM() function call in order to ensure that image files get written to this directory. Now hit 'd' when running your code and you should get a large number of PPM files being written to this directory of the form imgNNN.ppm, where NNN is the frame number. To convert these image files to an MPEG movie, use

ffmpeg -i img%03d.ppm -r 24 a1.mp4

Don't forget to delete all your PPM files once your movie has been created to save some disk space. You can play your movie using

ffplay a1.mp4

If you want to keep your movies reasonably small in size, I recommend choosing an appropriate window size for your animation. For example, a 500 x 300 video will be approximately 1/4 the size of a 1000 x 600 video.

Bonus points You can get up to 5 bonus points for particularly well done animations, detailed models, or other extensions. Bonus points are given at the discretion of the TA.

Hand-in Instructions

- Create a root directory in your account called cs314. Later all the assignment handin files should be put in this directory.
- For assignment 1, create a folder called assn1 under cs314 and put all the source files that you want to handin in it, including your "makefile" and your README.txt file. Do not use subdirectories. Submit your movie file (a1.mp4) if it is less than 1.5 MB in size. If it is larger than this, you should instead provide a URL in your README.txt file that points to a location from which we can download your movie.

- In your README file, please describe how to run your program, what functionalities you have implemented, as well as any kind of information you would like to give us for getting credit for partial implementation. If you don't complete all the requirements, please state clearly what you have tried, what problems you are having, and what you think might be promising solutions.
- The assignment should be handed in with the command:

  handin cs314 assn1

  This will handin your assn1 directory tree by making a copy of your assn1 directory. Note that subdirectories will not be copied. If you want to know more about this handin command, use: man handin. **The deadline is after the Thursday lab (5pm).**
- Fill in and sign the sheet on the next page, and turn it in together with your solutions to the theoretical problems in class (2pm).

Collaboration Policy

In this course, graded assignments are intended to be solved by individual students. Each student must submit their own individual solution.

If you have collaborated with other students on the solution to this assignment, this fact needs to be disclosed in the form below. **Note that disclosing a collaboration may result in reduced points**. Likewise, if external resources (other than the course web pages and text book) were used for solving the assignment, they need to be listed below. **Failure to disclose a collaboration or external resources constitutes an act of academic misconduct, and will be reported to the dean**.

Declaration

I hereby declare that I have read and understood the above statement.

Name:

Student ID:

Signature and Date:

I have used the following external resources:

I have collaborated with the following individuals (explain degree of collaboration):