Assignment 4: Lights, Camera (from before) and Action!

Out: 21st October 2017 **Due: 1st November 2017 at 8:59pm**

In this assignment you will extend your program from Assignment 3 by adding an animated object and lights while keeping your camera keyboard-controlled.

Look at the provided humanoid-lights.xml for examples of different aspects of this assignment.

Part 1 : Lighting (30 points)

In this part, you must add light support to your scene graph.

Lighting basics (20 points)

A light can be attached to any type of node (in the coordinate system of the node).

Modify your scene graph classes so that any node can store multiple lights. Add functions to add a light to a node so that it can be used by the XML parser. Modify the parser so that it can parse lights, as specified in the example xml file.

1. The material of an object can be specified as shown in the XML file. Your XML reader should already be able to parse all material tags.
   Note: The light uses some of the same tags as the material (e.g. ambient, diffuse, specular). Thus the parsing of these tags depends on whether you are specifying material or light properties.
2. Incorporate the lighting shaders from the example in class in your code. The scene graph will now (automatically) keep track of all the new shader variables.
3. Write a function that will descend the scene graph, convert all the lights into the view coordinate system and return them in a list. Be sure to incorporate the animation transforms if any.
4. Modify the scene graph's draw function so that it:
   a. Obtains all lights in the view coordinate system.
   b. Pass these lights to the shaders (position/direction, colors)
   c. Draws the scene graph.
5. Drawing a leaf should now pass material properties instead of simply a "color".

Think about how you will implement 4 above. This functionality should be suitably divided among the scene graph and the renderer classes. Remember that they have been designed so that the scene graph and nodes are independent of the library used to render them (i.e. JOGL), while all the JOGL-specific code is confined to the renderers.

I highly recommend that you prepare a small, simple scene with 1-2 objects to test your program.

Spot lights (10 points)

The lighting shaders currently do not support more than one light. Add support for spotlights. This involves two things: changing the XML parser so that it supports additional tags for spot direction and angle for a light, and changing the shader so that it takes these parameters and uses them correctly.

Part 2: Lights in your scene (10 points)

Enhance the scene graph model that you created in Assignment 3. Instead of "color", each part of your model should now have material. Your scene should have at least 3 lights, at least one of which must be a spotlight.

Part 3: New model and animation (50 points)

In this part you will create a new model and animate it.

Part 3A: Create the model (25 points)

You are free to create any model, so long as the following constraints are obeyed:

1. The model should have at least 10 object instances.
2. The model should be such that it is movable, and should have two independent degrees of motion. As an example, the entire model moves in one way, while one or more of its parts move another way.

Be sure to use the following features of the XML file that will help you model as well as animate:

a. Build the model "bottom up", like we built the fan example in class. Start with the smaller parts. Then move them to their correct positions by adding them to transform nodes, etc. Build this interactively using your program, rather than typing in the entire file and then viewing it!

   As before, you may find it easier to build parts in separate xml files and then assemble it.

b. Use **descriptive and unique** names for nodes appropriately, using the "name=<value>" attribute to any node.

Part 3B: Animate the model (25 points)

1. The scene graph implementation stores a table of all the named nodes in your scene graph. You can use this table to directly access a node in your scene graph, given its name.
2. An empty animate function has been provided to you in the Scenegraph class. Fill this function to animate your specific model.
   a. For each node that you wish to animate, add an animation transform. Again remember that only TransformNode can store an animation transform, so you may need to add one at specific locations in your scene graph just to produce this animation.
   b. First look up the node by name in the table, determine its transformation for animation using "time" and then set it to the animation transformation of that node.
   c. Your animate function will thus look for specifically named nodes. This will mean that your animate code will be specific to your own scene graph. This is OK.

Extra credit (5 points)

Make an animation video of your final rendering (in mp4 or avi format). The video should showcase your achievement in this assignment. Look at the description on blackboard and the example code.

Program preparation and README file (10 points)

Prepare your program so that lights can be turned on and off, one by one. Specifically, the 1, 2, 3 keys on your keyboard should work to toggle (turn off/on) lights 1, 2 and 3 respectively.

The README file should summarize what features work in your program. It should also include a link to the extra credit video, if you have done it.

What to submit

Submit the IntelliJ/Qt project folder set up correctly with your scene files you used as a zipped file (make sure you include the README file).