Assignment 4: A lit, textured and interactive virtual world

Out: 27th October 2016 **Due: 15th November 2016 at 11:59pm**

In this assignment you will extend your program from Assignment 3 by adding another animated object, lights, textures and making your camera keyboard-interactive.

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

Note: Since you are modifying scene graph files provided to you, please move the sgraph folder from your extras/headers to your project folder, so that it is included in your submission. Doing this will expedite grading.

Part 1 : Lighting and Textures (50 points)

In this part, you must add light and texture 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).

1. 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.
2. 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.
3. 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.
4. 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.
5. 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.
6. Drawing a leaf should now pass material properties instead of simply a "color".

Think about how you will implement 5-6 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/Qt), 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 spot lights. 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.

Texture mapping (20 points)

An image can be specified in the XML file using the <image> tag. Each image tag specifies a unique name. All such images are converted into TextureImage objects (look at the TextureImage class in your extras/headers folder). Then, an object can use a texture using the "texture" attribute. Look at the provided XML file for an example of this.

1. Look at the XML parser to see how it is parsing the "image" tag and "texture" attribute of the object tag. This will help you how the texture is being loaded and stored.
2. Look at the Scenegraph and Scenegraph renderer classes to see how textures are being stored.
3. Add code so that it enables the appropriate texture object, and uses it.

Note: The code in the shaders that does texture mapping will now expect a texture bound to every object. For those objects that are not textured, this will produce an incorrect result. In order to avoid this, create a 1x1 "white" texture image, and include it in your scene XML file. Then, either change the parser, or the actual drawing so that if there is no texture bound to the object, it uses this white texture instead. A white texture essentially means "no texture", since the shader is blending the texture color with the shading color.

Add a single texture to your small scene graph to test this part.

Part 2: Your scene (20 points)

Enhance the scene graph model that you created in Assignment 3 in the following two ways:

1. Lighting and texturing changes (10 points)
    a. Instead of "color", each part of your model should now have material (with ambient, diffuse, specular, shininess tags).
    b. At least two different texture images should be used in your scene. They should be clearly visible in the rendering (i.e. don't texture a part that is so tiny in the rendering that it is difficult to see the texture).
    c. Your scene should contain at least one stationary light, and at least two lights that are tied with a moving part of the models. At least one light should be a spotlight, and its effect should be clearly visible in the rendering.
2. Add a second animated scene graph model to your scene. This should be DIFFERENT than the one you have from Assignment 3. It should also have materials instead of color. (10 points)

You are free to use images that you obtained from other sources, assuming you are legally allowed to. HOWEVER if you do not own them, you must cite their source in a separate README file clearly mentioning which image was taken from where. Failure to do this will result in a grade penalty!

Part 3: Object-stationary camera (15 points)

In this part, you will add a camera that is fixed to one of the moving models. Pressing 'O' or 'o' should switch to this camera, and pressing 'g' or 'G' should revert to the global stationary camera.

Extra credit 2: More elaborate scene (5 points)

Add more objects to your scene. They may be multi-node scene graphs, or directly mesh files. Your program should be able to load the entire scene with 1 XML file (which may refer to other XML files)!

To receive credit, the objects should be relevant to your scene (i.e. don't place random objects): the extra credit is for your creativity since your program already has all the machinery to render additional objects.

Part 4: Document (15 points)

Add a chapter to your document from Assignment 3, which describes:

1. Write a brief description of your scene, in story form. This should give context to the user about what he/she is seeing. The moving camera (and the extra credit interactive camera) is also part of the context. Be creative!
2. A user manual for this program, confined to showing features after it is run. This should include screen shots of what it can do, and any features that it supports (including control of the camera). This is your chance to show me what you have achieved, so make sure you capture good screen shots!
3. A description of each model in your scene. This description should include which parts it has, which shapes it is made of. This description, in contrast with the description in Assignment 3, is directed at a user of your program and not me. Thus, it is important to be non-technical.
4. Clearly identify which partner worked on which parts of these programs (Assignment 3+4), if applicable.
5. Provide an estimate of the time it took to complete all the features in Assignment 4, along with a rating "Difficult", "Manageable", "Easy".

Extra credit: (Due by November 21st by demo only, give demo during office hours or schedule with me)

Extra credit 1: Keyboard/mouse-controlled camera (10 points)

Add a third camera that can be controlled by the keyboard or mouse. Pressing 'i' or 'I' should switch to this camera.

You can add the controls as they fit within your scene, with the following constraints:

1. The camera should be able to "nod" left/right, up/down and clockwise/counter-clockwise (2 points each).
2. The camera should be able to "shift" left/right and up/down (2 points each).

You are free to decide the keyboard/mouse controls for the above motions.

I highly recommend NOT making the camera tied to an object keyboard-controlled. Visualizing it would be difficult and debugging it would be time-consuming.

Extra credit 2: (Due to by November 21st by email or email link) (5 points)

Make an animation video of your final rendering (in mp4 or avi format). The video should showcase your achievement in these assignments.

What to submit

Submit the IntelliJ/Qt project folder set up correctly with your scene files, your document, and any images you used as a zipped file (make sure you include the README file with citations, keyboard control and a summary of what you could and could not complete). **Please make sure you have included the sgraph folder as part of your project folder (as specified in the beginning of this assignment!)**