Assignment 3: Modeling and Scene Setting

Out: 14th October 2016

Preliminary writeup due: 19th October 2016

Program due: 25th October 2016 at 11:59pm

You will work in pairs for this assignment. Both partners will get the same grade, so please make sure everybody contributes equally.

In this assignment you will interact with a scene graph implementation. Several scene graph models have been provided to you in XML files. A program that should read these models and render them is also provided to you. In this assignment, you will learn how to model an object hierarchically using a scene graph. You will then create a 3D world with at least two instances of the same animated scene graph model. Finally you will implement two cameras: one that is stationary and another that is tied to one of the moving models.

Overall Design of World

In the next two assignments, you will create an interactive 3D program that has the following aspects:

1. (This assignment) At least two instances of the same animated model, each of which is made of at least 10 object instances.
2. (This assignment) Each model should have two independent motions. For example, the entire model moves a certain way, while one of its parts moves another way.
3. (Divided in this and next assignment) The models are placed in a world that must have a planar ground, plus any other models.
4. (Next assignment) A second animated model.
5. (Next assignment) An object-stationary camera.
6. (Next assignment) Lights and textures, at least one moving light.

Given these constraints, you must come up with a vision of a program that does something meaningful. An example may be to model a scene of two locomotives moving along a pre-defined path with a fly-through keyboard-controlled camera, or an amusement park scene with two rides and a keyboard-controlled FPS camera moving on the ground. You are not constrained to choose one of these: you can come up with your own so long as it has the above elements.

Preliminary Writing (Due Wednesday October 19th at 11:59pm) (20 points)

Your preliminary writeup should be in the form of powerpoint or PDF slides (no more than 4 slides + 1 title slide). This is like a "vision pitch" to increase your and my anticipation of what you will produce. These slides should contain:

1. Names of both partners on the title slide.
2. A bullet-point description of what your program will do at the end of Assignment 4. Describe the scene, the models you will create and a meaningful goal for the navigation of the camera.
3. A description of your hierarchical model (what it will be, what will be the above motions and how you will create the scene graph so that it will have the above structure and motions).
4. Include a schematic of the scene graph that you will implement (a drawing with nodes and edges as we drew in class. Hand-drawn and scanned is OK, provided it is legible and neat).

Think of these slides as "planning" your assignments. You don't have to actually implement anything to show on the slides.

Starting code

Please re-download the extras (Java) or headers (C++) files: they have changed.

A Scenegraphs project has been provided to you. It uses an inbuilt SAX parser for reading in the scene graph as an XML file.

Javadoc-style documentation has been provided with the project and in each class to help you understand the structure and design of the program. I recommend you step through the program to understand how it is working (specifically View's draw where it draws the scene graph). It is a good way to quickly understand how various parts of the program work.

Several scene graph models have been provided to you that show various features of this program.

1. Face-hierarchy.xml shows a simple Jack-in-the-box model as a scene graph. This is probably the smallest and simplest example.
2. humanoid.xml draws a simple humanoid model.
3. two-humanoids.xml draws how to combine two xml files into one.

Look at the XML files and look for comments that point to the specific transformations. This will give you an idea about how to add transformations in specific places to move the scene graph accordingly.

Part 2: The Hierarchical model (60 points)

In this part, you will create the beginnings of your "scene" using two animated instances of the same hierarchical model.

Part 2A: Y,M,C,A poses (10 points)

Modify the humanoid.xml file to create 2 files, which show the humanoid in ANY TWO of the Y,M,C,A poses from the famous song. Finally create a file called "humanoid-YMCA.xml" that arranges the two humanoids side by side, so that all of them can be loaded as one scene graph (see two-humanoids.xml for an example of how to do this).

Part 2B: Create the model (30 points)

You are free to create any model, so long as the constrained at the beginning of this assignment are obeyed.

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!
b. The only node that can store an animation transform is the TransformNode. So make sure you use a TransformNode for the parts that you wish to animate later. Look at the comments in the humanoid.xml file for examples.
c. Use descriptive names for nodes appropriately, using the "name=<value>" attribute to any node.
d. Assemble (in a separate file) the scene graph of your entire world with at least two instances of this model. Look at "two-humanoids.xml" to see how to do this. **Warning: when the program puts one scenegraph as part of a bigger one, it pre-pends all the names of the nodes in the scene graph with the name of the group that is used to import the scene graph.**

This part should not require any Java coding! All you have to do is type an XML file in your favorite text editor and load it in the given program.

Part 2C: Animate the model (20 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.

Documentation (20 points)

Create a single doc/PDF file with the following:

1. A more detailed description (no more than 1 page) of your envisioned world that you motivated in your slides.
2. The description of the animated model along with a screen shot.
3. A short description explaining how you produced this animation. Refer to your scene graph file, point to specific line numbers that have nodes that you animate, and then include snippets from your code that animate them. A reader should be able to figure out how to produce the animation, given this description, your scene graph file and my starter program.

Program details

1. Prepare your program so that when I build and run, your scene is visible from the stationary camera.
2. In your project, include a README text file that has the names of the partners and a list of which items you were able to complete and which items you were not able to fully complete. **Be sure to specify how your program takes the input scene graph file.**

What to submit

Submit the slides separately by the deadline. For the actual program deadline, submit your entire IntelliJ/Qt project and your documentation in as a single zipped file on Blackboard. Only one submission per group is expected.