COMP222 Tutorial 3

Playing with the scene graph

The aim of this tutorial is to familiarise with the use of the scene graph to control how to move game entities. We will also add lights and shadows to the scene.

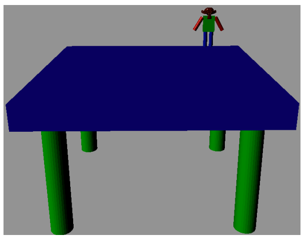
You can either start from scratch or download the tutorial3.zip project from http://intranet.csc.liv.ac.uk/~konev/COMP222/tutorials/Tutorial3.zip You can import the project into your workspace by selecting *File→Import Project→From ZIP…*

The project contains the monkey robot model from http://blender.freemovies.co.uk and the model of a table that you were developing last week. Notice that I have removed lights from the blender files (otherwise, you would be able to see the models even if no light source is added to the scene graph).

Tutorial 3 tasks:

The tutorial consists of several steps. The final code can be found at http://intranet.csc.liv.ac.uk/~konev/COMP222/tutorials/LoadModels.java however, I would encourage you follow the steps one by one rather copy from mine.

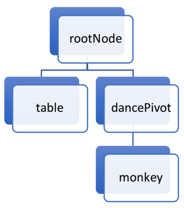
1. Scale and move the monkey so that it looks like a toy on the table (as shown on the right).



   To do that, add the following lines to your `impleInitApp()` method:

```
monkey.scale(0.1f);
monkey.move(1, 1, 0);
```

2. Make the monkey "dance" around.



   To achieve that, you need to introduce a pivot node. Your scene graph should look like the one shown on the right. To do that, create and initialise a dancePivot field in your class

```
private Node dancePivot = new Node();
```

   and replace the

```
rootNode.attachChild(monkey);
```

   line in the `SimpleInitApp()` method with

```
dancePivot.attachChild(monkey);
rootNode.attachChild(dancePivot);
```

   Then, add the following line to the `simpleUpdate(float tpf)` method:

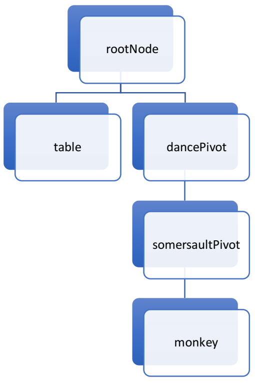
```
dancePivot.rotate(0, tpf, 0);
```

3. Make the monkey "pirouette" as it dances around.

You can do that by specifying that the monkey should rotate around the Y axis. Simply add the following line to the `simpleUpdate(float tpf)` method

```
monkey.rotate(0, 4 * FastMath.PI * tpf, 0);
```

4. Make the monkey "somersault" as it dances and pirouettes around.

   To achieve that, you need to change the scene graph structure. Your new structure is shown on the right. Introduce another field for `somersaultPivot` (same ways as for `dancePivot`) and change the code in your `SimpleInitApp()` method to:

   ```
   dancePivot.attachChild(somersaultPivot);
   somersaultPivot.attachChild(monkey);
   rootNode.attachChild(dancePivot);
   ```

   Then update the `simpleUpdate(float tpf)` method:

   ```
   // monkey spins
   monkey.rotate(0, 4 * FastMath.PI * tpf, 0);
   // monkey somersaults
   somersaultPivot.rotate(-FastMath.PI * tpf, 0, 0);
   // monkey dances
   dancePivot.rotate(0, tpf, 0);
   ```

5. Make the monkey jump up and down as it rotates around different axes. We will use the trigonometric function $\sin x$ and a timer for that.

   Introduce a floating-point value field in your class

   ```
   private float myTimer = 0;
   ```

   We will progress the timer and use it to compute the vertical offset for the `dancePivot`. Add the following code to your `SimpleInitApp()` method:

   ```
   // monkey jumps
   myTimer += tpf;
   somersaultPivot.setLocalTranslation(1, 0.4f *
          FastMath.sin(FastMath.PI * myTimer) + .8f, 0);
   ```

   You'll notice that the monkey does not touch the table now. Remove `monkey.move` from your `SimpleInitApp()` method. Explore what happens if instead of moving the `somersaultPivot` node you move some other scene graph nodes.

6. Set up the camera.

   By default, the camera is positioned at location (0, 0, 10), which does not give a good view at the table. Add the following line to your `SimpleInitApp()` method:

   ```
   // Setting up the camera
   cam.setLocation(new Vector3f(0, 2, 7));
   ```

7. To make the scene more lively, add a point light by inserting the following

   ```
   PointLight myLight = new PointLight();
   myLight.setColor(ColorRGBA.White);
   myLight.setPosition(new Vector3f(0, 2, 0));
   myLight.setRadius(20);
   rootNode.addLight(myLight);
   ```

   into your `SimpleInitApp()` method. Notice that the radius determines how far the light from the point light travels. You can further experiment with light, see https://jmonkeyengine.github.io/wiki/jme3/advanced/light_and_shadow.html for details.

8. Finally, we add shadows by adding the following lines to your `SimpleInitApp()` method:

   ```
   // The monkey can only cast shadows
   monkey.setShadowMode(RenderQueue.ShadowMode.Cast);
   // The table can both cast and receive
   table.setShadowMode(RenderQueue.ShadowMode.
        CastAndReceive);
   // setting up the shadow renderers,
   //     every kind of light needs a separate one
   PointLightShadowRenderer plsr =
        new PointLightShadowRenderer(assetManager, 512);
   plsr.setLight(myLight);
   plsr.setFlushQueues(false); // should be false for all
                               // but the last renderer
   DirectionalLightShadowRenderer dlsr = new
    DirectionalLightShadowRenderer(assetManager,512,2);
   dlsr.setLight(sun);
   // adding them to the view port (what we see)
   viewPort.addProcessor(plsr);
   viewPort.addProcessor(dlsr);
   ```