Assignment 1

Name: Jason Bishop
Student #: 3042012
Course: COMP382

*Note: All Mile Tracer Extension files for BlueJ are located in the Question1 folder for this project.

1. (required) Perform the following:

a) Create a 3D environment called MyOwn3D that consists of an abstract room with four walls, i.e., the inside of a cube and a floor.
b) Give the floor and walls a solid color.
c) Divide the room into grids such that each side is a 3 x 3 grid.

See the BlueJ project file in *Question1* for the implementation of the MyOwn3D class.

2. (required) Populate the scene from MyOwn3D with the following objects:

a) various inclined planes
b) two spheres
c) two cubes of different sizes
d) a rectangular column
e) two poles

See the BlueJ project file in *Question2* for the MyOwn3D scene with populated objects.

3. In your own words, compare and contrast the two Java 3D variants.

The two Java 3D variants are implemented on top of OpenGL and DirectX Graphics. The DirectX version is only available on Windows platforms, whereas the OpenGL version is supported on a wide variety of platforms. Another difference is that the DirectX version is reported to be slightly faster, but the difference is not large, and many users do not notice a difference. However, it seems that the OpenGL version is the primary focus by Sun in Java 3D. They began Java 3D with it and implemented DirectX later. They have been providing more support, such as bug fixes, that are making the OpenGL version more stable and reliable. Nonetheless, there are very few technical differences between the two versions and the implementation is largely the same. This is because they are based on similar graphics pipeline architecture.

4. In your own words, identify three key advantages and three key disadvantages of Java 3D for game programming.

Three key advantages of Java 3D for game programming are the presence of the scene graph, performance, and of course, Java integration. The scene graph is very useful because it makes the complex task of 3D programming somewhat easier. It hides the low-level aspects of 3D graphics and allows for easier organization of the 3D environment. Additionally, the scene graph can group shapes with common properties. Additionally, multithreading is used to render and traverse the scene graph. Java 3D obtains its performance abilities though scene graph optimizations as well as implementing OpenGL (or DirectX) for low-level performance. Also, optimizations can be further achieved by programmers using capability bits or even bypassing the scene graph itself. This can give greater control over rendering and scene management. Finally, since Java 3D is based in Java, all the advantages of object-oriented programming provided by Java are also available in Java 3D. Other Java APIs can be used as well, allowing for a vast array of options available to the programmer.

Three key disadvantages of Java 3D for game programming are garbage collection, lack of console support, and additional overhead and problems with the scene graph. The garbage collector in Java runs without direct control of the programmer. While running, the rendering performance of Java 3D can be significantly slowed. This slowdown can be reduced with careful programming, the lack of control can sometimes hinder applications. This is especially the case with games, because they can often be processor intensive. If the processor is occupied with garbage collection activities, then it will not be able to render the application as efficiently. The additional overhead of the scene graph in Java 3D needs to be considered. Some programmers proficient in OpenGL and DirectX find this additional overhead intolerable. However, there are optimizations that come with this overhead (see above) that can offset this fact. Yet there is also the problem implementing advanced rendering techniques such and vertex and pixel shading because Java 3D cannot perform these functions. Finally, the lack of console support for games can be a major problem in the area of games. Modern platforms like the Xbox One and PS4 do not support games developed with Java. Neither do older generation systems such as the Xbox 360 and PS3. These are the major leaders in the home videogaming market. However, the emergence of the PC market as a major competitor mitigates this factor somewhat.

5. Explain a method to create a two-cylinder yo-yo only using quad-strip objects and triangle-fan objects. Hint: refer to the Self-Test in Getting Started with the Java™ 3D API, Chapter 2.

N/A

6. What is an external model? Explain its needs and functionality with an example.

An external model is simply an object loaded into Java 3D that is not created with the standard geometry classes available. Instead, the object is created using 3D modeling software and loaded into the program. This allows for much more advanced modeling and use of properties such as geometry and texturing in a more progressive fashion. An example of this is the robot model used in my BlueJ project file in *Question6*.

As can be seen in the example, using the LoadModel class, the model is loaded without many of it's properties. No textures or color is used, and the model may or may not be to scale with the virtual environment. Additionally, certain models do not orient correctly when loaded into the scene graph. For most uses, the model will need to be rotated, scaled, translated, and appearance properties will need to be set. These changes can be seen when using the LoadModelFixed class.

7. How would you examine a model’s scene graph? Explain with an example.

N/A

8. What is a lathe shape? Explain with an example.

N/A