Instituto Politécnico Nacional

Escuela Superior de Cómputo

Programación Orientada a Objetos

Reporte de Práctica #3

Java 3D

Profesor: Roberto Tecla Parra

Alumno: Calva Hernández José Manuel

Grupo: 2CM3

Sistema Solar (Planetario)

Agregar 2 “planetas” mas y para cada "planeta"

- crear una apariencia
- cargar una textura a partir del archivo de una imagen
- poner la textura en la apariencia
- crear una esfera con el radio y la apariencia correspondiente al planeta
- rotar la esfera sobre su propio eje a la velocidad correspondiente al planeta (duración del día)
- alejar la esfera del sol (la posición del sol es el origen)
- rotar la esfera alrededor del sol a la velocidad correspondiente al planeta (duración del anio).
- agregarla al BranchGroup

Objetivos

- ➢ Aprender a crear objetos en Java 3d.
- ➢ Entender el concepto de marco de referencia, estableciendo un origen (que puede ser un objeto) para el uso de objetos en Java 3D.
- ➢ Aprender a poner una imagen sobre un objeto (como una etiqueta).
- ➢ Hacer uso del manejo de objetos como alejarlo del origen, rotarlo sobre su propio eje, rotarlo alrededor del origen (traslación), cambiar el tamaño del mismo.
- ➢ Crear un Universo, en el cual serán “seteados” todos los objetos.

Desarrollo

Para el inicio de esta práctica, se hizo uso de un código que venía en la carpeta de prácticas de Java, en la cual venían importadas librerías de java y de Java 3D.

```
import com.sun.j3d.utils.geometry.*;
import com.sun.j3d.utils.image.TextureLoader;
import com.sun.j3d.utils.universe.*;
import javax.media.j3d.*;
import javax.vecmath.*;
import java.awt.*;
import javax.swing.*;
import java.util.*;
```

Se creó una clase nueva llamada SolarSis, también lo que fue el BranchGroup que permite agrupar un conjunto de nodos bajo una raíz y las apariencias del Sol, Mercurio, La Tierra y Marte.

```
BranchGroup group = new BranchGroup();
Appearance appsol = new Appearance();
Appearance appmercury = new Appearance();
Appearance appearth = new Appearance();
Appearance appmars = new Appearance();
```

Se crearon las texturas para cada uno de nuestros objetos y a su vez se le puso a cada uno una imagen que los cubriría (la textura).

```
TextureLoader tex=new TextureLoader("TIERRA.JPG", null);
appearth.setTexture(tex.getTexture());
tex=new TextureLoader("MERCURIO.JPG", null);
appmercury.setTexture(tex.getTexture());
tex=new TextureLoader("MARTE.JPG", null);
appmars.setTexture(tex.getTexture());
tex=new TextureLoader("SOL.JPG", null);
appsol.setTexture(tex.getTexture());
```

Se estableció el tipo de figura (de tercera dimensión), que en este caso fue una esfera para todos nuestros objetos (planetas y Sol), además se fijó el tamaño del radio para cada uno de ellos.

```
Sphere mercury = new Sphere(0.030f, Primitive.GENERATE_NORMALS |
Primitive.GENERATE_TEXTURE_COORDS, 32, appmercury);
Sphere earth = new Sphere(0.045f, Primitive.GENERATE_NORMALS |
Primitive.GENERATE_TEXTURE_COORDS, 32, appearth);
Sphere mars = new Sphere(0.050f, Primitive.GENERATE_NORMALS |
Primitive.GENERATE_TEXTURE_COORDS, 32, appmars);
Sphere sol = new Sphere(0.35f, Primitive.GENERATE_NORMALS |
Primitive.GENERATE_TEXTURE_COORDS, 32, appsol);
```

A cada uno de nuestros objetos se les aplicó una transformación o “cambio” que fue el de hacerlos sobre su propio a cierta velocidad distinta en cada uno de ellos.

```
TransformGroup mercuryRotXformGroup = Posi.rotate(mercury, new Alpha(-1, 750));
TransformGroup earthRotXformGroup = Posi.rotate(earth, new Alpha(-1, 1250));
TransformGroup marsRotXformGroup = Posi.rotate(mars, new Alpha(-1, 1750));
TransformGroup solRotXformGroup = Posi.rotate(sol, new Alpha(-1, 1250));
```

Después se le aplicó otra transformación a cada uno de nuestros objetos, en este caso fue alejarlos del origen, cuya posición ocupa el Sol, y para ello fue necesario de ubicarlos acorde su posición de la Vía Láctea (solo a los planetas).

```
TransformGroup mercuryTransXformGroup = Posi.translate(mercuryRotXformGroup,
new Vector3f(0.0f, 0.0f, 0.4f));
TransformGroup earthTransXformGroup = Posi.translate(earthRotXformGroup,
new Vector3f(0.0f, 0.0f, 0.7f));
TransformGroup marsTransXformGroup = Posi.translate(marsRotXformGroup,
new Vector3f(0.0f, 0.0f, 1.9f));
```

La última transformación que se les aplicó fue la de rotar, pero ahora alrededor del origen, para generar la traslación, cada uno con distinta velocidad de acuerdo con la duración de año.

```
TransformGroup mercuryRotGroupXformGroup = Posi.rotate(mercuryTransXformGroup, new Alpha(-1, 2000));
group.addChild(mercuryRotGroupXformGroup);
TransformGroup earthRotGroupXformGroup = Posi.rotate(earthTransXformGroup, new Alpha(-1, 5000));
group.addChild(earthRotGroupXformGroup);
TransformGroup marsRotGroupXformGroup = Posi.rotate(marsTransXformGroup, new Alpha(-1, 8000));
group.addChild(marsRotGroupXformGroup);
group.addChild(solRotXformGroup);
```

Por último, se agregó cada uno de nuestros objetos al BranchGroup.

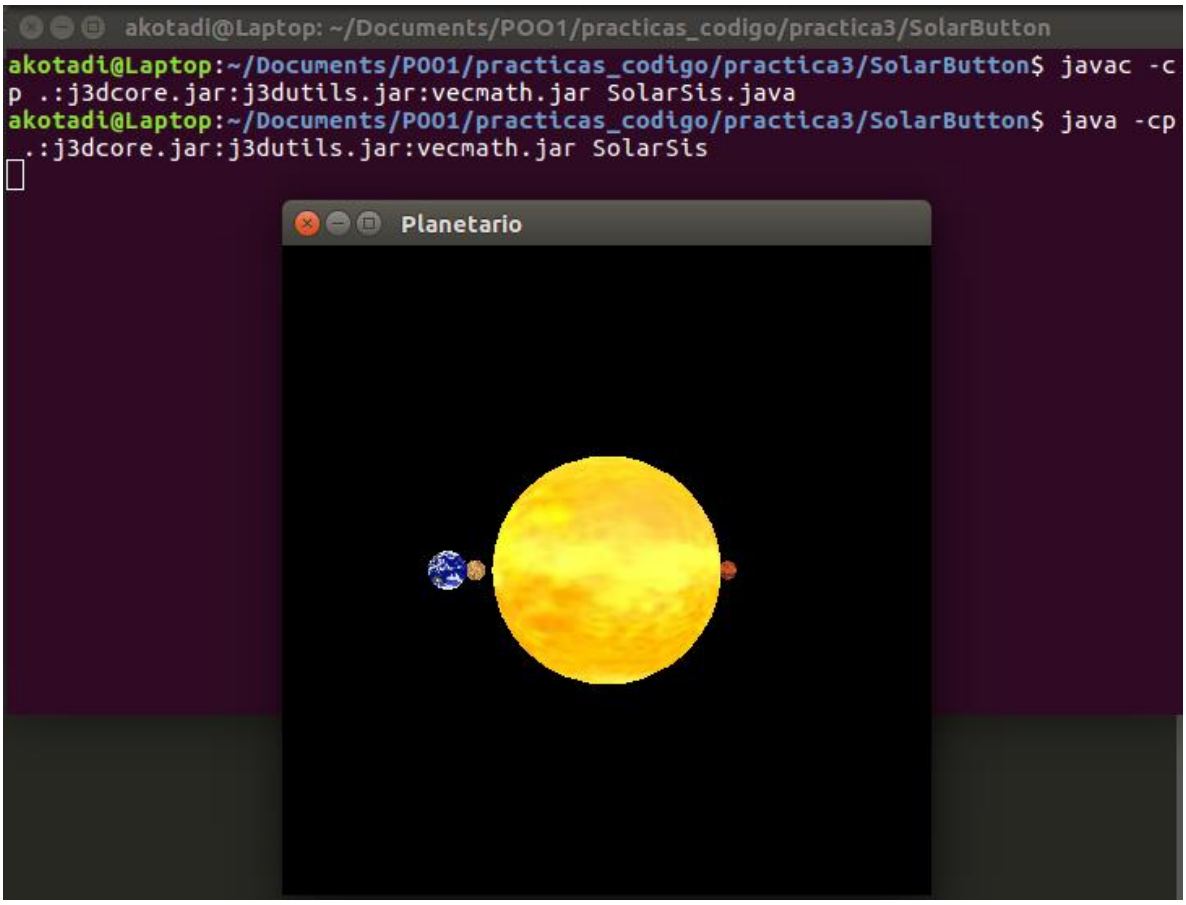
Para terminar, se establecieron las condiciones (líneas de código) necesarias para crear al SimpleUniverse, el cuál sería el escenario donde se encontrarían nuestros objetos y realizarías todas las acciones descritas previamente, y así crearíamos nuestro nuevo SolarSis.

```
    GraphicsConfiguration config = SimpleUniverse.getPreferredConfiguration();
    Canvas3D canvas = new Canvas3D(config); canvas.setSize(400, 400);
    SimpleUniverse universe = new SimpleUniverse(canvas);
    universe.getViewingPlatform().setNominalViewingTransform();
    universe.addBranchGraph(group);
    JFrame f = new JFrame("Planetario");
    f.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    f.add(canvas); f.pack(); f.setVisible(true);

}

public static void main(String a[]) {
    new SolarSis();
}
```

Para ejecutar el programa debe compilarse y ejecutarse con paquetes externos como se muestra a continuación:



Conclusiones

Es importante tener en cuenta que todos nuestro objetos en Java 3D serán puestos en un “escenario” en el cual realizarán todas las acciones que nosotros les programemos, para ponerle una imagen o textura a un objeto es necesario primero crearla y luego ponérsela (estas dos líneas de código van seguidas respectivamente una de la otra) para cada uno de nuestros objetos, a simple vista si no se le aplica una acción a los objetos de Java 3D parecería que son simples objetos en 2D, ya que carecen de acciones y por ende nuestro marco de referencia con respecto es distinto.