INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

ALUMNO: Meza Vargas Brandon David.

PRÁCTICA: Práctica No. 3.

TEMA: Java 3D.

OPCIÓN: Opción 1, Sistema Solar (Planetario).

FECHA: 27-nov-2020

GRUPO: 2CM1

MATERIA: Programación Orientada a Objetos

INTRODUCCIÓN

Java 3D es un API de gráficos 3D desarrollada por Sun como una extensión del JDK 2 del lenguaje de programación Java. Es una colección de clases que tienen como objetivo principal facilitar la creación y representación de escenas tridimensionales en la computadora, así como la animación e interacción con las mismas.

Para construir un mundo virtual en Java 3D, debemos crear y manipular objetos geométricos tridimensionales que se encuentran en un universo virtual que después es renderizado. El renderizado se hace en paralelo gracias al aprovechamiento de los Threads de Java. Los programas pueden escribirse para ejecutarse como aplicaciones o como applets en navegadores.

DESARROLLO

En esta practica se escogió la opción numero 1, la cual es un planetario que consta del sol, la tierra y dos planetas más, en este caso venus y mercurio.

Primeramente, importamos las librerías necesarias para realizar la practica sin problemas, ver figura 1;

```
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 java.io.File;
import javax.swing.*;
import java.util.*;
```

Figura 1. librerías.

Posteriormente creamos las apariencias para cada planeta y el sol, esto gracias a ***Appearance***. Una vez creada la apariencia, con una variable tex de tipo TextureLoader cargaremos una textura con una imagen y se la pondremos a la apariencia de cada planeta con setTexture, ver figura 2;

```
Appearance appsol = new Appearance();
Appearance appearth = new Appearance();
Appearance appvenus = new Appearance();
Appearance appmercurio = new Appearance();

TextureLoader tex=new TextureLoader("C:\\Users\\PC\\Documents\\NetBeansProjects\\Planet\\src\\images\\TIERRA.JPG", null);
appearth.setTexture(tex.getTexture());
tex=new TextureLoader("C:\\Users\\PC\\Documents\\NetBeansProjects\\Planet\\src\\images\\SOL.JPG", null);
appsol.setTexture(tex.getTexture());
tex=new TextureLoader("C:\\Users\\PC\\Documents\\NetBeansProjects\\planetario\\src\\images\\venus.jpg",null);
appvenus.setTexture(tex.getTexture());
tex=new TextureLoader("C:\\Users\\PC\\Documents\\NetBeansProjects\\planetario\\src\\images\\mercurio.jpg",null);
appmercurio.setTexture(tex.getTexture());
```

Figura 2. Apariencias y textura.

Antes de seguir, en nuestro programa contamos con métodos, que se encargan del movimiento de rotación y traslación de los planetas, ver figura 3;

```
    TransformGroup rotate(Node node, Alpha alpha) {
        TransformGroup xformGroup = new TransformGroup();
        xformGroup.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
RotationInterpolator interpolator =new RotationInterpolator(alpha, xformGroup);
interpolator.setSchedulingBounds(new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 1.0));
        xformGroup.addChild(interpolator); xformGroup.addChild(node);
        return xformGroup;
}
TransformGroup translate(Node node, Vector3f vector) {
        Transform3D transform3D = new Transform3D();
        transform3D.setTranslation(vector);
        TransformGroup transformGroup =new TransformGroup();
        transformGroup.setTransform(transform3D);
        transformGroup.addChild(node);
        return transformGroup;
```

Figura 3. Código encargado de la rotación y translación.

Una vez sabido esto, podemos continuar, la siguiente figura nos muestra la parte del código que hace el movimiento de rotación y translación cada planeta.

```
TransformGroup earthRotXformGroup = rotate(earth, new Alpha(-1, 1250));
TransformGroup solRotXformGroup = rotate(sol, new Alpha(-1, 1250));
TransformGroup venusRotXformGroup = rotate(venus, new Alpha(-1,1250));
TransformGroup mercurioRotXformGroup = rotate(mercurio, new Alpha(-1,1250));

TransformGroup earthTransXformGroup = translate(earthRotXformGroup, new Vector3f(0.0f, 0.0f, 0.7f));
TransformGroup earthRotGroupXformGroup = rotate(earthTransXformGroup, new Alpha(-1, 5000));

TransformGroup venusTransXformGroup= translate(venusRotXformGroup, new Vector3f(0.0f,0.0f,.7f));
TransformGroup venusRotGroupXformGroup = rotate(venusTransXformGroup, new Alpha(-1, 2000));

TransformGroup mercurioTransXformGroup= translate(mercurioRotXformGroup, new Vector3f(0.0f,0.0f,0.7f));
TransformGroup mercurioRotGroupXformGroup = rotate(mercurioTransXformGroup, new Alpha(-1, 1000));
```

Figura 4. Movimientos de rotación y translación.

Posteriormente agregamos esto al BranchGroup, ver figura 5;

```
group.addChild(earthRotGroupXformGroup);
group.addChild(venusRotGroupXformGroup);
group.addChild(mercurioRotGroupXformGroup);
group.addChild(solRotXformGroup);
```

Figura 5. Agregando al BranchGroup.

Finalmente creamos un lienzo en donde se verá lo creado, este se agrega al universo, además de crearse un frame para la visualización del programa.

En el main solo creamos un objeto de tipo Planetario.

```
    GraphicsConfiguration config = SimpleUniverse.getPreferredConfiguration();
    Canvas3D canvas = new Canvas3D(config);
    canvas.setSize(400, 400);
    SimpleUniverse universe = new SimpleUniverse(canvas);
    universe.getViewingPlatform().setNominalViewingTransform();
    universe.addBranchGraph(group);
    Vector nomPlanet=new Vector();
    nomPlanet.addElement("Sol"); nomPlanet.addElement("Tierra"); nomPlanet.addElement("Venus");nomPlanet.addElement("Mercurio");
    jcb=new JComboBox(nomPlanet);
    jcb.addActionListener(this);
    JPanel jp=new JPanel(); jp.add(jcb);
    JFrame f = new JFrame("Planetario");
    f.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    f.setLayout(new BorderLayout());
    f.add("Center",canvas); f.add("South", jp);
    f.pack(); f.setVisible(true);


public static void main(String[] args) {
    new Planet();
}
```

Figura 6.creando universo, frame y main

Cuando compilamos y corremos el programa este se ve de la siguiente manera:

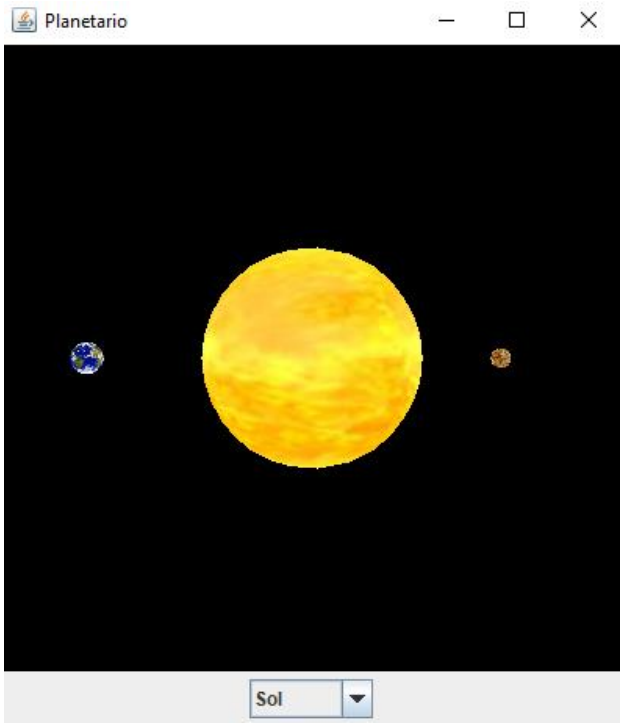

Figura 7. Ejecución del programa.

El funcionamiento del programa se ve mas a detalle en el video adjunto a este reporte.

CONCLUSIÓN

Como vimos en la introducción y desarrollo de esta práctica, java 3D nos ayuda a construir mundos virtuales, permitiendo al usuario crear y manipular objetos tridimensionales. Es una forma, algo laboriosa a mi parecer, de poder crear mundos tridimensionales, donde podemos hacer uso de muchas herramientas del API 3D, podemos explicar desde modelos, recrear escenas o simplemente para un entretenimiento realizando animaciones.

Fue una práctica donde tuve un poco de confusión ya que este tema fue visto de manera muy rápida.

Con esta practica adquirí conocimientos, al menos básicos, de java3D ya que nunca había tocado este tema.