Práctica 3

Java3D

Programación Orientada a Objetos

Humberto Alejandro Ortega Alcocer

Opción 1: Sistema Solar

15 de enero de 2021

Grupo: 2CM1

Introducción

En computación no hay nada más interesante que poder crear visualizaciones de información interesantes. A lo largo de la historia de las computadoras, distintos científicos y grandes personalidades dentro del mundo de la computación han creado programas y aplicaciones que permiten visualizar datos de formas cada vez mas realistas. En primer lugar, para poder comunicar más apropiadamente lo que está sucediendo, pero en segundo lugar porque, conforme estas visualizaciones se han vuelto más y más complejas, se presta para poder añadir interacciones nunca antes vistas por los seres humanos.

Al inicio, los programas debían trabajar directamente con una terminal (o línea de comandos), por lo que todas las representaciones gráficas de información debían responder a un modelo de dos dimensiones (2D), sin embargo, al introducirse el concepto de "gráficos computacionales", se ha dado poco a poco el paso hacia una visualización en tres dimensiones (3D) con la cual es posible no solamente tener una idea más acertada sobre los objetos representados en pantalla, sino que se permitió el desarrollo de juegos de video, simulaciones espaciales y de prototipos, así como una infinidad de otras aplicaciones dónde una visualización en tres dimensiones es requerida. La práctica actual, consiste en mostrar una visualización del sistema solar, y para esto añadiremos dos planetas nuevos a la visualización.

El planteamiento de la práctica, en palabras del profesor Roberto Tecla, es:

1.-**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

Desarrollo

Para el desarrollo de esta práctica se utilizaron los archivos proporcionados previamente por el profesor: SisPos.java y SolarSis.java. En éstos archivos, se incluye la implementación de un modelo del sistema solar simple y nuestra tarea a realizar para la práctica es añadir dos planetas nuevos y mostrarlos.

En el caso del archivo SisPos.java, aquí se incluye una función que permite calcular la posición de un planeta dentro de nuestra escena en función de la órbita descrita, su masa, y su velocidad de desplazamiento. Veámoslo como una función de ayuda para poder realizar el cálculo físico para poder tener certeza en nuestra visualización.

La parte relevante de la práctica se encuentra en SolarSis.java, y he hecho una refactorización del archivo únicamente para darle más legibilidad y denotar donde he realizado los cambios necesarios para la práctica.

Los planetas que se añadieron fueron:

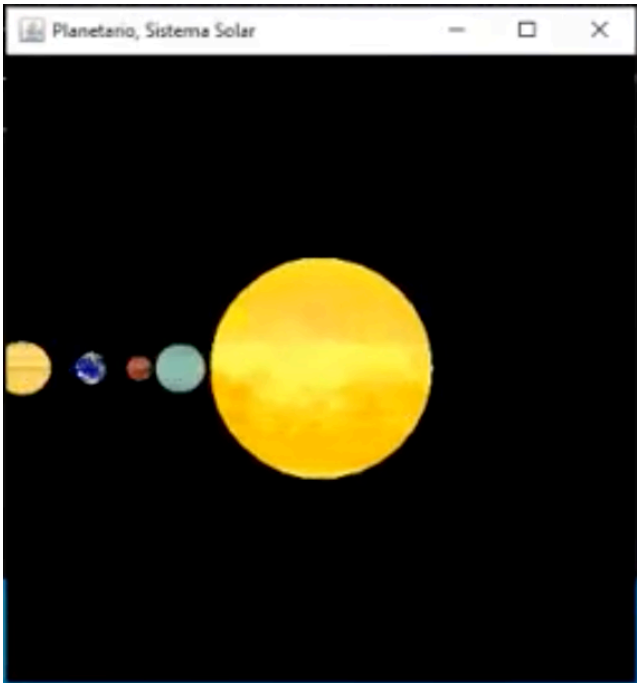
- Saturno
- Urano

Los cuales cuentan con su información disponible en Wikipedia para poder realizar los ajustes necesarios en el código que los representa. Es importante notar que las traslaciones y proyecciones realizadas dentro del programa no corresponden a una escala temporal lineal, por lo que, si intentáramos colocar los datos “tal cual” los encontramos, veremos que las órbitas trazadas por dichos planetas en nuestra simulación son mucho más cortas que lo que esperamos. Para solucionar este problema, he ajustado los valores a la escala en la que se encuentran el resto de los planetas que ya se encontraban previamente definidos dentro de la práctica.

En la siguiente hoja, se incluye el código de la clase SolarSis dónde se pueden apreciar los cambios realizados.

```
public class SolarSis {
  public SolarSis() {
    // creamos el Grupo
    BranchGroup group = new BranchGroup();

    // crear Apariencias
    Appearance appsol = new Appearance();
    Appearance appearth = new Appearance();
    Appearance appMarte = new Appearance();
    Appearance appJupiter = new Appearance();
    Appearance appSaturno = new Appearance();
    Appearance appUrano = new Appearance();

    // Cargar una textura a partir del archivo de una imagen y poner la textura en
    // la aparienica
    TextureLoader tex = new TextureLoader("TIERRA.JPG", null);
    appearth.setTexture(tex.getTexture());
    tex = new TextureLoader("SOL.JPG", null);
    appsol.setTexture(tex.getTexture());
    tex = new TextureLoader("MARTE.JPG", null);
    appMarte.setTexture(tex.getTexture());
    tex = new TextureLoader("JUPITER.JPG", null);
    appJupiter.setTexture(tex.getTexture());
    tex = new TextureLoader("Saturno.JPG", null);
    appSaturno.setTexture(tex.getTexture());
    tex = new TextureLoader("Urano.JPG", null);
    appUrano.setTexture(tex.getTexture());

    // crear una esfera con el radio y la apariencia correspondiente al planeta
    Sphere earth = new Sphere(0.045f, Primitive.GENERATE_NORMALS | Primitive.GENERATE_TEXTURE_COORDS, 32, appearth);
    Sphere sol = new Sphere(0.35f, Primitive.GENERATE_NORMALS | Primitive.GENERATE_TEXTURE_COORDS, 32, appsol);
    Sphere Marte = new Sphere(0.023f, Primitive.GENERATE_NORMALS | Primitive.GENERATE_TEXTURE_COORDS, 32, appMarte);
    Sphere Jupiter = new Sphere(0.18f, Primitive.GENERATE_NORMALS | Primitive.GENERATE_TEXTURE_COORDS, 32, appJupiter);
    Sphere Saturno = new Sphere(0.20f, Primitive.GENERATE_NORMALS | Primitive.GENERATE_TEXTURE_COORDS, 32, appSaturno);
    Sphere Urano = new Sphere(0.22f, Primitive.GENERATE_NORMALS | Primitive.GENERATE_TEXTURE_COORDS, 32, appUrano);

    // rotar la esfera sobre su propio eje a la velocidad correspondiente al planeta
    // (duración del día)
    TransformGroup earthRotXformGroup = Posi.rotate(earth, new Alpha(-1, 1250));
    TransformGroup MarteRotXformGroup = Posi.rotate(Marte, new Alpha(-1, 1292));
    TransformGroup JupiterRotXformGroup = Posi.rotate(Jupiter, new Alpha(-1, 507));
    TransformGroup solRotXformGroup = Posi.rotate(sol, new Alpha(-1, 1250));
    TransformGroup SaturnoRotXformGroup = Posi.rotate(Saturno, new Alpha(-1, 1250));
    TransformGroup UranoRotXformGroup = Posi.rotate(Urano, new Alpha(-1, 708));
    // alejar la esfera del sol (la posición del sol es el origen)
    TransformGroup earthTransXformGroup = Posi.translate(earthRotXformGroup, new Vector3f(0.0f, 0.0f, 0.7f));
    TransformGroup MarteTransXformGroup = Posi.translate(MarteRotXformGroup, new Vector3f(0.0f, 0.0f, 1.06f));
    TransformGroup JupiterTransXformGroup = Posi.translate(JupiterRotXformGroup, new Vector3f(0.0f, 0.0f, 3.64f));
    TransformGroup SaturnoTransXformGroup = Posi.translate(SaturnoRotXformGroup, new Vector3f(0.0f, 0.0f, 4.00f));
    TransformGroup UranoTransXformGroup = Posi.translate(UranoRotXformGroup, new Vector3f(0.0f, 0.0f, 4.64f));
    // rotar la esfera alrededor del sol a la velocidad correspondiente al planeta
    // (duración del año).
    TransformGroup earthRotGroupXformGroup = Posi.rotate(earthTransXformGroup, new Alpha(-1, 2500));
    TransformGroup MarteRotGroupXformGroup = Posi.rotate(MarteTransXformGroup, new Alpha(-1, 4705));
    TransformGroup JupiterRotGroupXformGroup = Posi.rotate(JupiterTransXformGroup, new Alpha(-1, 6125));
    TransformGroup SaturnoRotGroupXformGroup = Posi.rotate(SaturnoTransXformGroup, new Alpha(-1, 7500));
    TransformGroup UranoRotGroupXformGroup = Posi.rotate(UranoTransXformGroup, new Alpha(-1, 8200));
    // agregarla al BranchGroup
    group.addChild(earthRotGroupXformGroup);
    group.addChild(MarteRotGroupXformGroup);
    group.addChild(JupiterRotGroupXformGroup);
    group.addChild(solRotXformGroup);
    group.addChild(SaturnoRotGroupXformGroup);
    group.addChild(UranoRotGroupXformGroup);

    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, Sistema Solar");

    f.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    f.add(canvas);
    f.pack();
    f.setVisible(true);
  }

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

El programa, una vez compilado y ejecutado correctamente, se visualiza de la siguiente forma:



Conclusión

El uso de Java3D es importante para poder comprender las dificultades y consideraciones a tener presentes cuando trabajamos en un ambiente gráfico de tres dimensiones. Pude notar que, en esencia, Java3D es un wrapper de OpenGL para Java, por lo que muchas de las funciones que se emplean, así como el orden en que se emplean, corresponden a su símil utilizando C/C++ o C# directamente con OpenGL.

Esta práctica me costó mucho trabajo ya que mi sistema operativo (MacOS) en su versión más reciente no admite el uso de Java3D, por lo que tuve que esperar a conseguir una computadora con Windows como sistema operativo principal para poder realizar la instalación de las librerías y componentes adecuados.

Me parece sumamente interesante lo simple que puede volverse el código para implementar una escena de este tipo, y lo rápido que se ejecuta una vez terminado. Además, la visualización me parece una herramienta educativa excelente y, si bien puede no corresponder con los valores reales de nuestro sistema solar a detalle, nos permite ver una escena dónde se realizan múltiples cálculos y derivaciones de forma continua para poder mostrar los elementos de forma adecuada en nuestro programa.