

# Instituto Politecnico Nacional

## Escuela Superior de Cómputo



Bocanegra Heziquio Yesllanezi

Practica 3

Java3D

Sistema solar

30 de abril de 2021

2CM13

Programación Orientada a Objetos







## Contenido

Introduccion.....	3
Qué es .....	3
¿Qué es el API 3D de Java?.....	3
Construir un escenario gráfico .....	3
Desarrollo .....	4
Codigo.....	5
Consola .....	9
Conclusion.....	10
 Código 1 SistemaSolar.java .....	7
Código 2 SistemaSolar0.java .....	7
Código 3 SistemaSolar1.java .....	8
Código 4 Simbolo Del Sistema.....	9
Código 5 Programa ejecutándose en Java3D .....	9





# Introducción

## Qué es

El API Java 3D es un interface para escribir programas que muestran e interactúan con gráficos tridimensionales. Java 3D es una extensión estándar del JDK 2 de Java. El API Java 3D proporciona una colección de constructores de alto-nivel para crear y manipular geometrías 3D y estructuras para dibujar esta geometría. Java 3D proporciona las funciones para creación de imágenes, visualizaciones, animaciones y programas de aplicaciones gráficas 3D interactivas

## ¿Qué es el API 3D de Java?

El API 3D de Java es un árbol de clases Java que sirven como interface para sistemas de renderizado de gráficos tridimensionales y un sistema de sonido. El programador trabaja con constructores de alto nivel para crear y manipular objetos geométricos en 3D. Estos objetos geométricos residen en un universo virtual, que luego es renderizado. El API está diseñado con flexibilidad para crear universos virtuales precisos de una amplia variedad de tamaños, desde astronómicos a subatómicos. A pesar de toda esta funcionalidad, el API es sencillo de usar. Los detalles de renderizado se manejan automáticamente. Aprovechándose de los Threads Java, el renderizador Java 3D es capaz de renderizar en paralelo. El renderizador también puede optimizarse automáticamente para mejorar el rendimiento del renderizado. Un programa Java 3D crea ejemplares de objetos Java 3D y los sitúa en una estructura de datos de escenario gráfico. Este escenario gráfico es una composición de objetos 3D en una estructura de árbol que especifica completamente el contenido de un universo virtual, y cómo va a ser renderizado. Los programas Java 3D pueden escribirse para ser ejecutados como aplicaciones solitarias o como applets en navegadores que hayan sido extendidos para soportar Java 3D, o ámbos.

## Construir un escenario gráfico

Un universo virtual Java 3D se crea desde un escenario gráfico. Un escenario gráfico se crea usando ejemplares de clases Java 3D. El escenario gráfico está ensamblado desde objetos que definen la geometría, los sonidos, las luces, la localización, la orientación y la apariencia de los objetos visuales y sonoros. Una definición común de un escenario gráfico es una estructura de datos compuesta de nodos y arcos. Un nodo es un elemento dato y un arco es una relación entre elementos datos. Los nodos en un escenario gráfico son los ejemplares de las clases Java 3D. Los arcos representan dos tipos de relaciones entre ejemplares Java 3D.





# Desarrollo

## 1.-Sistema Solar (Planetario)

Agregar 2 “planetas” más 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 año).
- agregarla al BranchGroup






```
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.*;

public class SistemaSolar {
public SistemaSolar(){
    BranchGroup group = new BranchGroup();
    Appearance appsol = new Appearance();
    Appearance appmercury = new Appearance();
    Appearance appvenus = new Appearance();
    Appearance appearth = new Appearance();
    Appearance appmars = new Appearance();
    TextureLoader tex=new TextureLoader("TIERRA.JPG", null);
    appearth.setTexture(tex.getTexture());
    tex=new TextureLoader("SOL.JPG", null);
    appsol.setTexture(tex.getTexture());
    tex=new TextureLoader("MERCURIO.JPG", null);
    appmercury.setTexture(tex.getTexture());
    tex=new TextureLoader("VENUS.JPG", null);
    appvenus.setTexture(tex.getTexture());
    tex=new TextureLoader("MARTE.JPG", null);
    appmars.setTexture(tex.getTexture());
    Sphere sol = new Sphere(0.35f, Primitive.GENERATE_NORMALS |
    Primitive.GENERATE_TEXTURE_COORDS, 32, appsol);
    Sphere mercury = new Sphere(0.035f, Primitive.GENERATE_NORMALS |
    Primitive.GENERATE_TEXTURE_COORDS, 32, appmercury);
    Sphere venus = new Sphere(0.040f, Primitive.GENERATE_NORMALS |
    Primitive.GENERATE_TEXTURE_COORDS, 32, appvenus);
    Sphere earth = new Sphere(0.045f, Primitive.GENERATE_NORMALS |
    Primitive.GENERATE_TEXTURE_COORDS, 32, appearth);
    Sphere mars = new Sphere(0.045f, Primitive.GENERATE_NORMALS |
    Primitive.GENERATE_TEXTURE_COORDS, 32, appmars);
    TransformGroup solRotXformGroup = Posi.rotate(sol, new Alpha(-1, 1250));
    TransformGroup mercuryRotXformGroup = Posi.rotate(mercury, new Alpha(-1, 2000));
    TransformGroup venusRotXformGroup = Posi.rotate(venus, new Alpha(-1, 1300));
    TransformGroup earthRotXformGroup = Posi.rotate(earth, new Alpha(-1, 1000));
    TransformGroup marsRotXformGroup = Posi.rotate(mars, new Alpha(-1, 1400));
    TransformGroup mercuryTransXformGroup = Posi.translate(mercuryRotXformGroup,
```



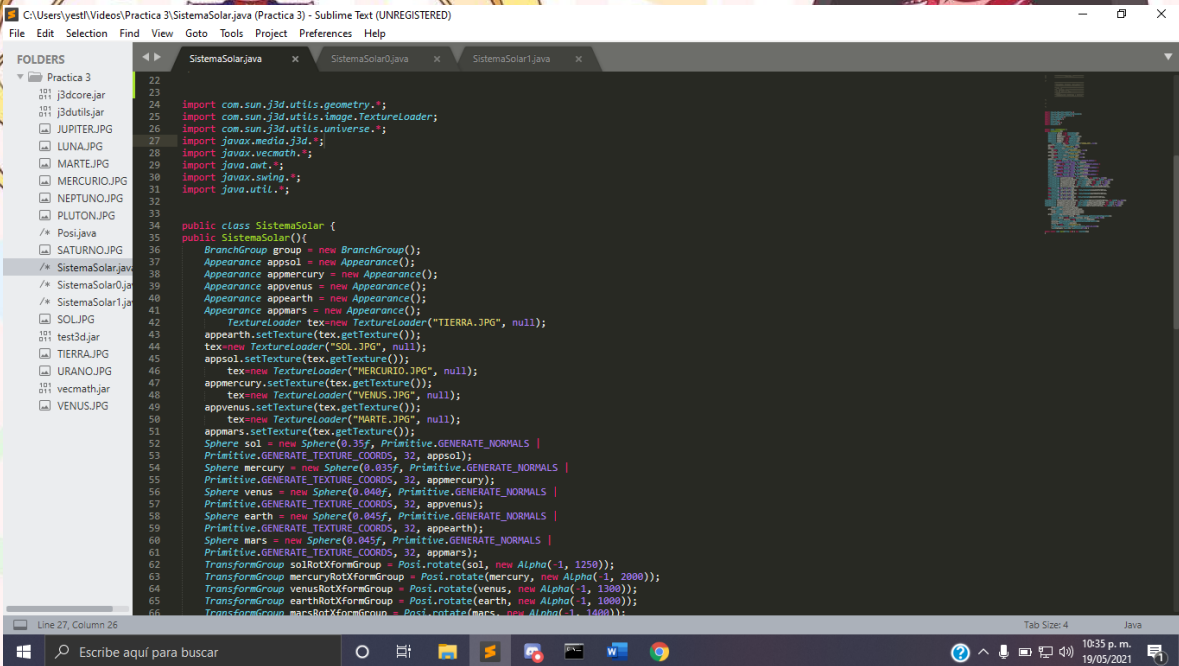


```

new Vector3f(0.0f, 0.0f, 0.4f);
TransformGroup venusTransXformGroup = Posi.translate(venusRotXformGroup,
new Vector3f(0.0f, 0.0f, 0.6f));
TransformGroup earthTransXformGroup = Posi.translate(earthRotXformGroup,
new Vector3f(0.0f, 0.0f, 0.8f));
TransformGroup marsTransXformGroup = Posi.translate(marsRotXformGroup,
new Vector3f(0.0f, 0.0f, 1.0f));
TransformGroup mercuryRotGroupXformGroup = Posi.rotate(mercuryTransXformGroup, new
Alpha(-1, 4000));
TransformGroup venusRotGroupXformGroup = Posi.rotate(venusTransXformGroup, new Alpha(-
1, 4900));
TransformGroup earthRotGroupXformGroup = Posi.rotate(earthTransXformGroup, new Alpha(-1,
5000));
TransformGroup marsRotGroupXformGroup = Posi.rotate(marsTransXformGroup, new Alpha(-1,
4600));
group.addChild(solRotXformGroup);
group.addChild(mercuryRotGroupXformGroup);
group.addChild(venusRotGroupXformGroup);
group.addChild(earthRotGroupXformGroup);
group.addChild(marsRotGroupXformGroup);
GraphicsConfiguration config = SimpleUniverse.getPreferredConfiguration();
Canvas3D canvas = new Canvas3D(config); canvas.setSize(500, 500);
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); }

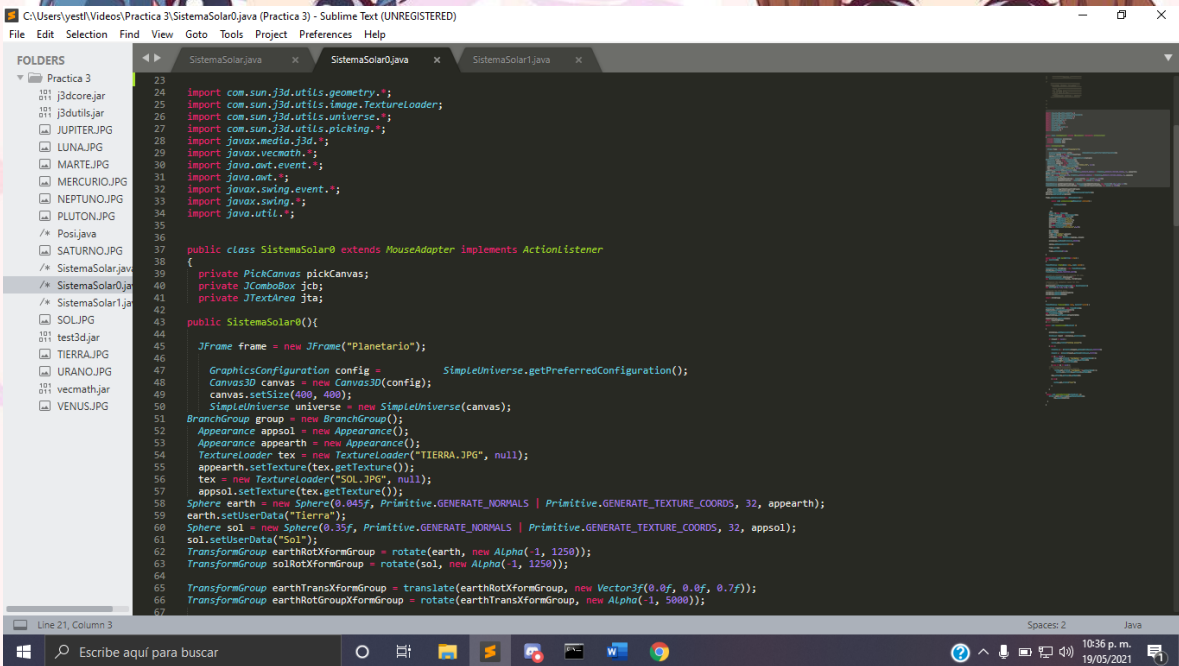
public static void main(String a[]) { new SolarSis(); }

```



```
22
23
24 import com.sun.j3d.utils.geometry.*;
25 import com.sun.j3d.utils.image.TextureLoader;
26 import com.sun.j3d.utils.universe.*;
27 import javax.media.j3d.*;
28 import javax.vecmath.*;
29 import java.awt.*;
30 import javax.swing.*;
31 import java.util.*;
32
33
34 public class SistemaSolar {
35     public SistemaSolar() {
36         BranchGroup group = new BranchGroup();
37         Appearance appsol = new Appearance();
38         Appearance appmercury = new Appearance();
39         Appearance appvenus = new Appearance();
40         Appearance appearth = new Appearance();
41         Appearance appmars = new Appearance();
42         TextureLoader tex = new TextureLoader("TIERRA.JPG", null);
43         appearth.setTexture(tex.getTexture());
44         tex = new TextureLoader("SOL.JPG", null);
45         appsol.setTexture(tex.getTexture());
46         tex = new TextureLoader("MERCURIO.JPG", null);
47         appmercury.setTexture(tex.getTexture());
48         tex = new TextureLoader("VENUS.JPG", null);
49         appvenus.setTexture(tex.getTexture());
50         tex = new TextureLoader("MARTE.JPG", null);
51         appmars.setTexture(tex.getTexture());
52         Sphere sol = new Sphere(0.35f, Primitive.GENERATE_NORMALS |
53             Primitive.GENERATE_TEXTURE_COORDS, 32, appsol);
54         Sphere mercury = new Sphere(0.035f, Primitive.GENERATE_NORMALS |
55             Primitive.GENERATE_TEXTURE_COORDS, 32, appmercury);
56         Sphere venus = new Sphere(0.040f, Primitive.GENERATE_NORMALS |
57             Primitive.GENERATE_TEXTURE_COORDS, 32, appvenus);
58         Sphere earth = new Sphere(0.045f, Primitive.GENERATE_NORMALS |
59             Primitive.GENERATE_TEXTURE_COORDS, 32, appearth);
60         Sphere mars = new Sphere(0.045f, Primitive.GENERATE_NORMALS |
61             Primitive.GENERATE_TEXTURE_COORDS, 32, appmars);
62         TransformGroup solRotXformGroup = Posi.rotate(sol, new Alpha(1, 1250));
63         TransformGroup mercuryRotXformGroup = Posi.rotate(mercury, new Alpha(1, 2000));
64         TransformGroup venusRotXformGroup = Posi.rotate(venus, new Alpha(1, 1300));
65         TransformGroup earthRotXformGroup = Posi.rotate(earth, new Alpha(1, 1000));
66         TransformGroup marsRotXformGroup = Posi.rotate(mars, new Alpha(1, 1000));
67     }
68 }
```

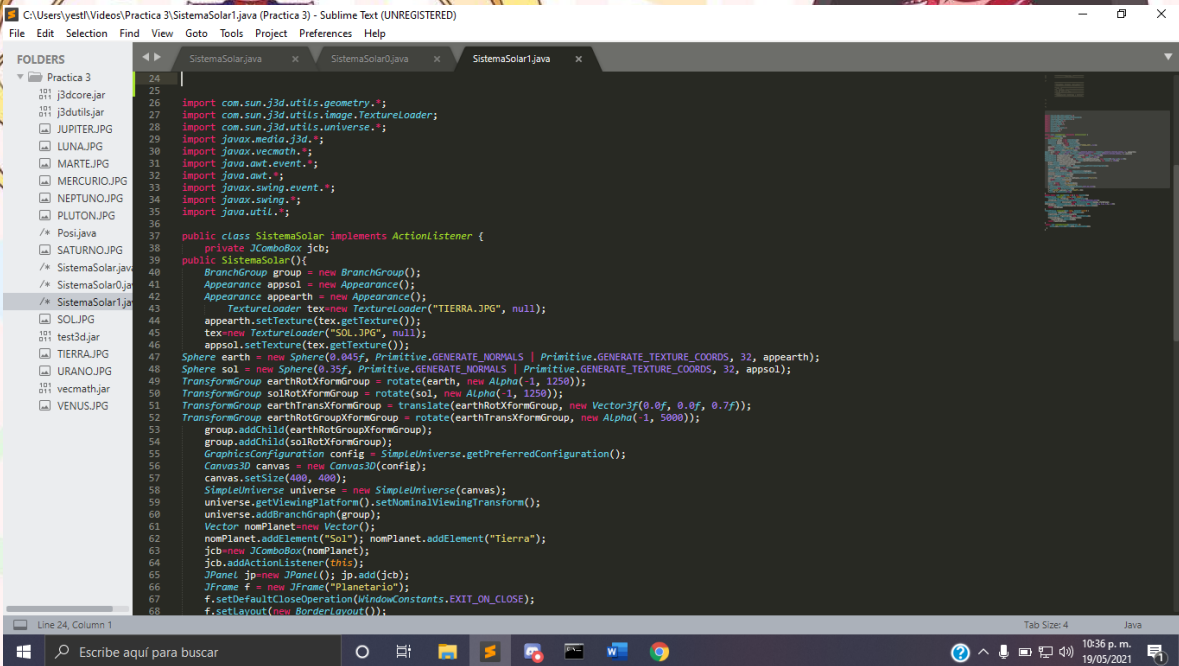
Código 1 SistemaSolar.java



```
23
24
25 import com.sun.j3d.utils.geometry.*;
26 import com.sun.j3d.utils.image.TextureLoader;
27 import com.sun.j3d.utils.universe.*;
28 import javax.media.j3d.*;
29 import javax.vecmath.*;
30 import java.awt.event.*;
31 import java.awt.*;
32 import javax.swing.event.*;
33 import javax.swing.*;
34 import java.util.*;
35
36
37 public class SistemaSolar0 extends MouseAdapter implements ActionListener {
38     private PickCanvas pickCanvas;
39     private JComboBox jcb;
40     private JTextArea jta;
41
42     public SistemaSolar0() {
43         JFrame frame = new JFrame("Planetario");
44
45         GraphicsConfiguration config = SimpleUniverse.getPreferredConfiguration();
46         Canvas3D canvas = new Canvas3D(config);
47         canvas.setSize(400, 400);
48         SimpleUniverse universe = new SimpleUniverse(canvas);
49         BranchGroup group = new BranchGroup();
50         Appearance appsol = new Appearance();
51         Appearance appearth = new Appearance();
52         TextureLoader tex = new TextureLoader("TIERRA.JPG", null);
53         appearth.setTexture(tex.getTexture());
54         tex = new TextureLoader("SOL.JPG", null);
55         appsol.setTexture(tex.getTexture());
56         Sphere earth = new Sphere(0.045f, Primitive.GENERATE_NORMALS | Primitive.GENERATE_TEXTURE_COORDS, 32, appearth);
57         earth.setUserData("Tierra");
58         Sphere sol = new Sphere(0.35f, Primitive.GENERATE_NORMALS | Primitive.GENERATE_TEXTURE_COORDS, 32, appsol);
59         sol.setUserData("Sol");
60         TransformGroup earthRotXformGroup = rotate(earth, new Alpha(1, 1250));
61         TransformGroup solRotXformGroup = rotate(sol, new Alpha(1, 1250));
62         TransformGroup earthTransXformGroup = translate(earthRotXformGroup, new Vector3f(0.0f, 0.0f, 0.7f));
63         TransformGroup earthRotXformGroup = rotate(earthTransXformGroup, new Alpha(1, 5000));
64     }
65 }
```

Código 2 SistemaSolar0.java





```
24
25
26 import com.sun.j3d.utils.geometry.*;
27 import com.sun.j3d.utils.image.TextureLoader;
28 import com.sun.j3d.utils.universe.*;
29 import javax.media.j3d.*;
30 import javax.vecmath.*;
31 import java.awt.event.*;
32 import java.awt.*;
33 import javax.swing.event.*;
34 import javax.swing.*;
35 import java.util.*;
36
37 public class SistemaSolar implements ActionListener {
38     private JComboBox jcb;
39     public SistemaSolar() {
40         BranchGroup group = new BranchGroup();
41         Appearance appsol = new Appearance();
42         Appearance appearth = new Appearance();
43         TextureLoader tex = new TextureLoader("TIERRA.JPG", null);
44         appearth.setTexture(tex.getTexture());
45         tex = new TextureLoader("SOL.JPG", null);
46         appsol.setTexture(tex.getTexture());
47         Sphere earth = new Sphere(0.045f, Primitive.GENERATE_NORMALS | Primitive.GENERATE_TEXTURE_COORDS, 32, appearth);
48         Sphere sol = new Sphere(0.35f, Primitive.GENERATE_NORMALS | Primitive.GENERATE_TEXTURE_COORDS, 32, appsol);
49         TransformGroup earthRotXformGroup = rotate(earth, new Alpha(1, 1250));
50         TransformGroup solRotXformGroup = rotate(sol, new Alpha(1, 1250));
51         TransformGroup earthTransXformGroup = translate(earthRotXformGroup, new Vector3f(0.0f, 0.0f, 0.7f));
52         TransformGroup earthRotXformGroup = rotate(earthTransXformGroup, new Alpha(1, 5000));
53         group.addChild(earthRotXformGroup);
54         group.addChild(solRotXformGroup);
55         GraphicsConfiguration config = SimpleUniverse.getPreferredConfiguration();
56         Canvas3D canvas = new Canvas3D(config);
57         canvas.setSize(400, 400);
58         SimpleUniverse universe = new SimpleUniverse(canvas);
59         universe.getViewingPlatform().setNominalViewingTransform();
60         universe.addBranchGraph(group);
61         Vector nomPlanet = new Vector();
62         nomPlanet.addElement("SOL");
63         jcb = new JComboBox(nomPlanet);
64         jcb.addActionListener(this);
65         JPanel jp = new JPanel();
66         JFrame f = new JFrame("Planetaario");
67         f.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
68         f.setLayout(new BorderLayout());
```

Código 3 SistemaSolar1.java

### Compilación en Windows

javac -cp .;j3dcore.jar;j3dutils.jar;vecmath.jar SistemaSolar.java

### ejecución

javac -cp .;j3dcore.jar;j3dutils.jar;vecmath.jar SistemaSolar



# Consola

(desde CMD) en Windows 10  
Versión de java 8

```
Simbolo del sistema
C:\Users\yestl\Videos\Practica 3>javac -cp .;j3dcore.jar;j3dutils.jar;vecmath.jar SistemaSolar.java
C:\Users\yestl\Videos\Practica 3>java -cp .;j3dcore.jar;j3dutils.jar;vecmath.jar SistemaSolar
C:\Users\yestl\Videos\Practica 3>
```

Código 4 Simbolo Del Sistema



Código 5 Programa ejecutándose en Java3D



## Conclusion

Las complicaciones para esta práctica comenzaron desde el momento en que intentaba instalar las paqueterías para poder ocupar java3D en mi computadora, ya que anteriormente tuve problemas con Java por las versiones, después de varios intentos descubrí que se debía exactamente a eso y tuve que quedarme con la versión 8 de java, solo así era como se podía ejecutar java3D.

Esta práctica fue compleja debido a que no había tenido la oportunidad de trabajar con Java3D, lo que procedió fue buscar las librerías y todas las funciones que se utilizaban, las imágenes que se necesitaban para la práctica el profesor las proporciono y eso lo hizo un poco menos complejo, ya que las especificaciones ya se encontraban, realizar la práctica fue algo muy complejo y aun no se si esta realizada de la manera adecuada o de la mas optima posible.

Java3D fue muy complejo para mi, espero poder aprender a manejarlo de una mejor forma, ya que su interfaz me agrado bastante.

