Java3D

Graficacion

Nos interesa la parte de la graficacion donde se usa la computadora para generar imágenes 3D esta area es conocida actualmente como CGI (*Computer Generated Imagery*)

Una escena tridimensional se compone de luces, cámaras, y objetos 3d.

Luz
No vemos las cosas , lo que vemos es la luz reflejada por las cosas.

Cámara
Lo que vemos depende de la posición y la orientación de la cámara

Objetos 3D
Una escena vacía carece de interés por tanto supondremos que contiene objetos 3d

Sistema de coordenadas
Origen de coordenadas es el punto con coordenadas (0, 0, 0).

En lugar de tomar las coordenadas de un objeto respecto al origen se pueden tomar relativas a otro objeto.

Transformaciones
Un objeto 3d se puede trasladar, rotar, o escalar,

Traslación
Sirve para mover el objeto 3d de lugar es decir sirve para cambiar sus coordenadas.

Rotación
Sirve para rotar el objeto 3d alrededor de un eje (usualmente x, y o z). Es decir sirve para cambiar la orientación del objeto.

Escala
Sirve para cambiar las dimensiones del objeto

Mapeo de textura
por medio de un mapeo matemático es posible mapear una imagen (la textura) en 2d en un objeto 3d. Por ejemplo se puede “pegar” una imagen a un cilindro.

Hay al menos dos librerías importantes para 3D en java: jogl y Java3D.

Java3D
En Java3D se cuenta con objetos 3d primitivos como: cilindros, conos, esferas y cubos

Cuando se crea un objeto 3D Java3D lo coloca en el origen por tanto si se necesita en una posición diferente entonces es necesario trasladarlo.

En java 3D una escena se describe de manera jerárquica (o mediante un grafo).

Java3D no es parte del núcleo de java ni siquiera es una extensión oficial del núcleo de java por tanto cuando

se importa un paquete de Java3D se importa empezando con

import com.

Ejemplo: Hola mundo en Java3D

```
import com.sun.j3d.utils.universe.*;
import com.sun.j3d.utils.geometry.*;
import javax.media.j3d.*;
public class Hola {
public Hola(){
   SimpleUniverse universe = new SimpleUniverse();
   BranchGroup group = new BranchGroup();
   group.addChild(new ColorCube(0.3));
   universe.getViewingPlatform().setNominalViewingTransform();
   universe.addBranchGraph(group);
}
public static void main( String[] args ) { new Hola(); }
}
```

Ejemplo de un mapeo de textura en un cilindro

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

public class Lata {
  public Lata(){
       GraphicsConfiguration config = SimpleUniverse.getPreferredConfiguration();
       SimpleUniverse universe = new SimpleUniverse(canvas);
         BranchGroup group = new BranchGroup();
         Appearance appsol = new Appearance();
        TextureLoader tex=new TextureLoader("lata.jpg", null);
         appsol.setTexture(tex.getTexture());
         Cylinder sol = new Cylinder(0.35f, 1.0f , Primitive.GENERATE_NORMALS |
         Primitive.GENERATE_TEXTURE_COORDS, appsol);
         Canvas3D canvas = new Canvas3D(config); canvas.setSize(400, 400);
         JFrame f = new JFrame("Lata");
         f.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
         f.add(canvas); f.pack(); f.setVisible(true);
         group.addChild(sol);
         universe.getViewingPlatform().setNominalViewingTransform();
         universe.addBranchGraph(group); }
        public static void main(String a[]) { new Lata();}

}
```

-Aplicar texturas a superficies
-Mover la cámara de lugar
-Detectar colisiones
-Agregar de forma dinámica un objeto a la escena
-Remover de forma dinámica un objeto a la escena