**Introducción a Java3D**

**API Java3D**

La API de Java3D es un conjunto de clases que sirven como interfaz a un sistema para desplegar gráficos tri-dimensionales y sonidos. Es similar a la bilioteca gráfica OpenGL, pero Java3D tiene la característica de ser estructurado.

Los programas de Java3D pueden ser escritos para ser corridos com programas independientes o como applets para ser mostrados en browsers.  
  
**Scene Graph**

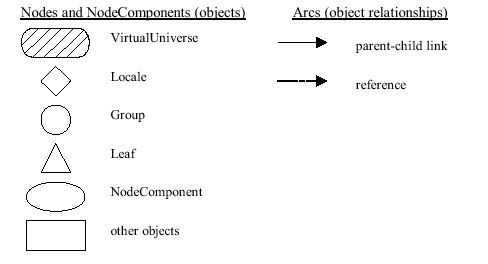
Un universo virtual de Java3D es definido por un grafo, que se conoce como *Scene Graph*. El Scene Graph es un ordenamiento de ojetos 3D en una estructura de árbol, que se crea instanciando ojetos de las clases de Java3D. Con este grafo se define la geometría, sonido, luz, ubicación, orientación y apariencia de ojetos visuales y de sonido.

Los nodos, generalmente se relacionan de la forma 'padre-hijo', pero también existe otro tipo de relación, las 'referencias'. Esta relación asocia un *Nodo Componente* con un nodo del Scene Graph. Un Nodo Componente define la geometría y la apariencia usada para mostrar los ojetos visuales.

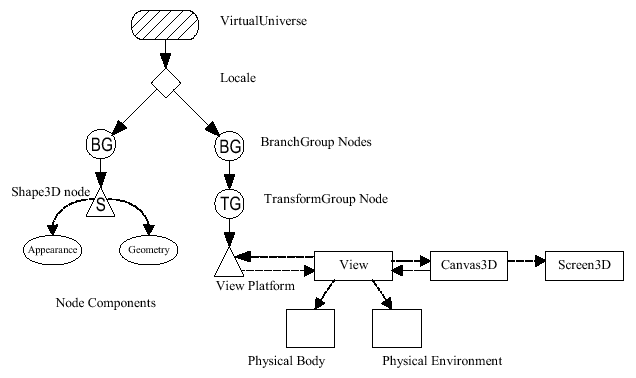
El camino de cada Scene Graph define completamente la información del estado de la hoja, es decir, la uicación, orientación y tamaño del ojeto.

La representación gráfica de un Scene Graph sirve como herramienta de diseño y/o como documentación para programas Java3D.

Los símbolos son:



Un ejemplo de un grafo es el siguiente:

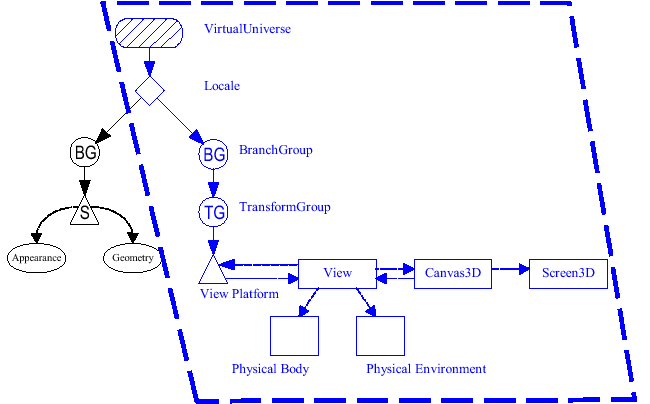


Para construir un grafo, se puede seguir la siguiente receta:

1. Crear un ojeto Canvas3D
2. Crear un objeto VirtualUniverse
3. Crear un objeto Locale, adjuntándolo al objeto VirtualUniverse
4. Construir una rama de vista:
   1. Crear un objeto View
   2. Crear un objeto ViewPlatform
   3. Crear un objeto PhysicalBody
   4. Crear un objeto PhysicalEnvironment
   5. Adjuntar los ojetos ViewPlatform, Physicalody, PhysicalEnvironment y Canvas3D al ojeto View
5. Construir una(s) rama(s) de contenido
6. Compilar las ramas del grafo
7. Insertar su-grafos en el Locale

**Receta Simple**

Los programas de Java3D que usan la receta anterior, tienen grafos con estructuras idénticas.



Por esto, existe la clase SimpleUniverse, que realiza los pasos 2, 3 y 4 de la receta anterior. Esto permite reducir considerablemente el tiempo y esfuerzo necesitado para crear el grafo. Usando esta clase, la receta anterior queda como:

1. Crear un ojeto Canvas3D
2. Crear un ojeto SimpleUniverse que referencia el objeto Canvas3D
   1. Personalizar el objeto SimpleUniverse
3. Construir la rama con el contenido
4. Compilar la rama de contenidp
5. Insertar la rama de contenido en el Locale del SimpleUniverse

**Componentes de un grafp**

Los componentes de un grafo son:

* BranchGroup (BG): permite agrupar un conjunto de nodos bajo una raiz
* TransformGroup (TG): tiene asociada una matriz de transformación que afecta a todos los elementos bajo este nodo

Un BG puede ser *compilado*, de manera de convertir todos sus descendientes a un formato más eficiente para poder desplegarlo en pantalla.  
  
Ejemplo: Mostrar un cubo de colores, rotado según uno de sus ejes.

import java.applet.Applet;

import java.awt.BorderLayout;

import java.awt.Frame;

import java.awt.event.\*;

import com.sun.j3d.utils.applet.MainFrame;

import com.sun.j3d.utils.universe.\*;

import com.sun.j3d.utils.geometry.ColorCube;

import javax.media.j3d.\*;

import javax.vecmath.\*;

public class HelloJava3Da extends Applet {

public HelloJava3Da(){

setLayout(new BorderLayout());

GraphicsConfiguration config =

SimpleUniverse.getPreferredConfiguration();

Canvas3D canvas3D =new Canvas3D(config);

add("Center",canvas3D);

BranchGroup scene=createSceneGraph();

scene.compile();

//SimpleUniverse is a Convenience Utility class

SimpleUniverse simpleU =new SimpleUniverse(canvas3D);

//This moves the ViewPlatform back a bit so the

//objects in the scene can be viewed.

simpleU.getViewingPlatform().setNominalViewingTransform();

simpleU.addBranchGraph(scene);

}//end of HelloJava3Da (constructor)

public BranchGroup createSceneGraph(){

//Create the root of the branch graph

BranchGroup objRoot =new BranchGroup();

//rotate object has composite transformation matrix

Transform3D rotate =new Transform3D();

rotate.rotX(Math.PI/4.0d);

TransformGroup objRotate =new TransformGroup(rotate);

objRotate.addChild(new ColorCube(0.4));

objRoot.addChild(objRotate); 13.

return objRoot;

}//end of createSceneGraph method of HelloJava3Da

//The following allows this to be run as an application

//as well as an applet

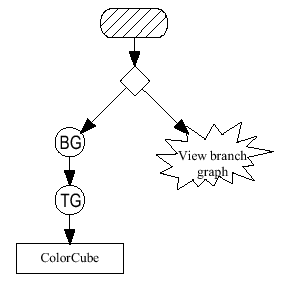
public static void main(String [] args)){

Frame frame =new MainFrame(new HelloJava3Da(),256,256);

}//end of main (methodof HelloJava3Da)

}// end of class HelloJava3Da

El grafo que representa lo anterior es:



y la representación es:

