

# SNS COLLEGE OF TECHNOLOGY

Coimbatore-35  
An Autonomous Institution



Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade  
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

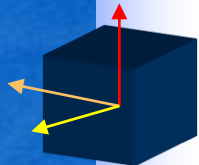
## DEPARTMENT OF INFORMATION TECHNOLOGY

### 16IT AUGMENTED REALITY AND VIRTUAL REALITY

III YEAR – V SEM

UNIT 5 – VR PROGRAMMING

TOPIC 3 –Java 3D Scene Graph - Sensors and Behaviours



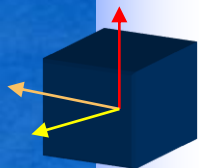
# How do we see the world?



Left eye

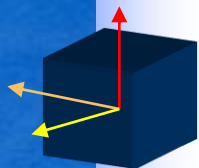


Right eye



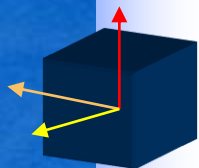
# Rendering a 3D object is hard!

- Rendering in Java 3D
  - Geometric models
  - Color
  - Shading
  - Texture
  - Light
  - movement
  
- Realism can range from opaque, shaded polygons to images approximating photographs in their complexity.



# Java 3D Overview

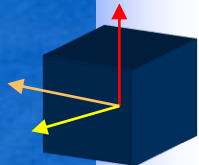
- A high-level (Object Oriented) API for building interactive 3D applications and applets
  - It enables authors to build shapes and control animation and interaction.
  - uses a *scene graph* to model/control the 3D scene
- Fast and efficient implementation on a variety of platforms
- Areas of Application
  - applets for spicing up web sites
  - Complex 3D graphics
  - Advance Scientific simulations



# What is java 3D made of ?

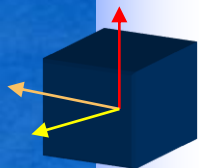
- What is java 3D made of ?
  - Instance of Java 3D Classes
- How are these classes related to each other?
  - By using a graph data structure called:

## The Scene Graph

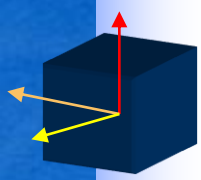
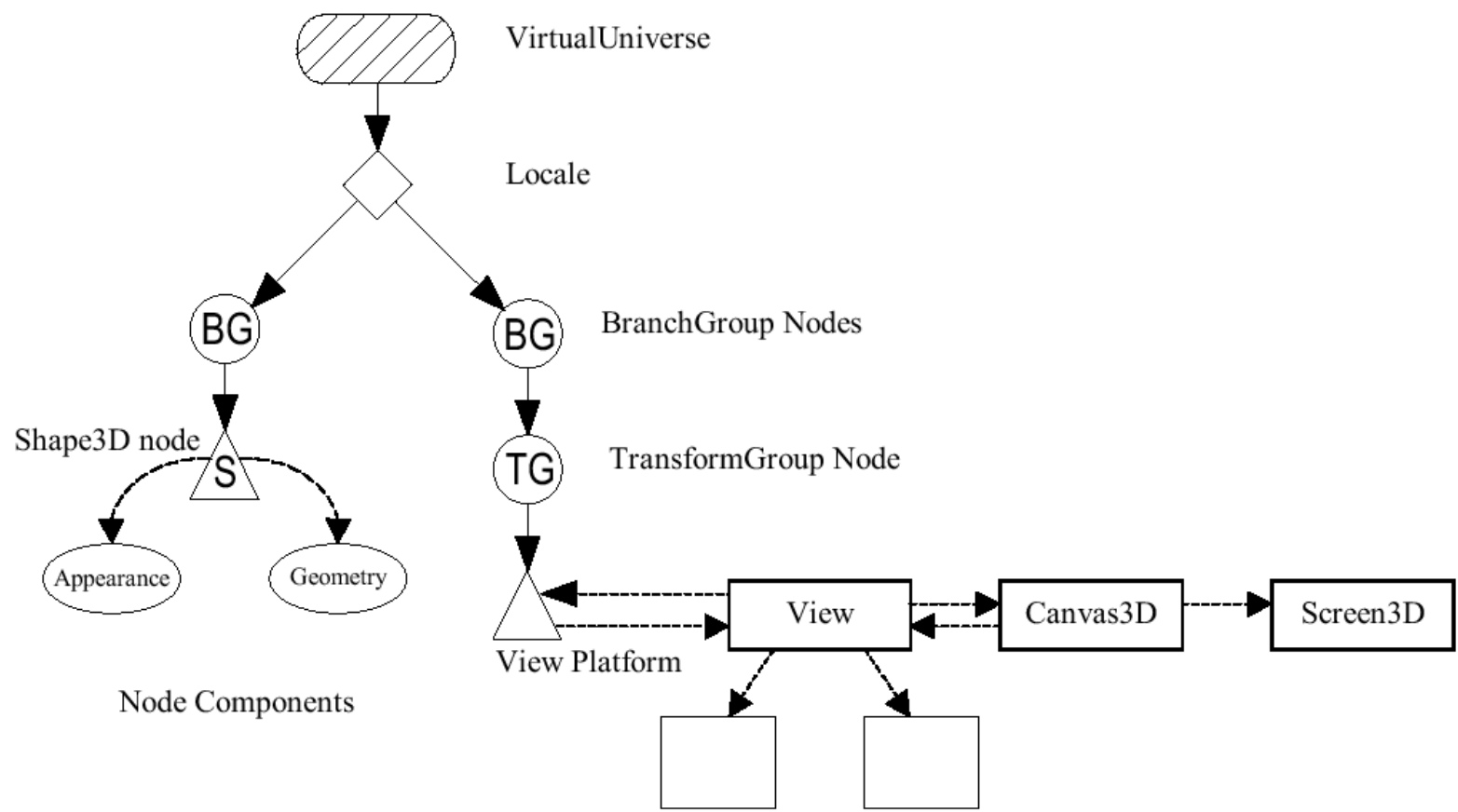


# What is a Scene Graph?

- A **scene graph** is a tree-like data structure that stores, organizes, and renders 3D scene information (3D objects, materials, lights, behaviours ...).
  - It is not a tree
  - It has nodes and arcs (connects the nodes)
  - Nodes are java classes


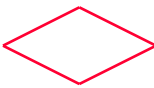
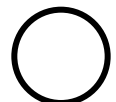
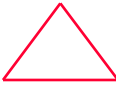




# A typical Scene Graph



# Scene Graph Symbols

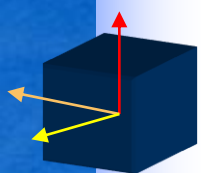
## Nodes and Node Components (objects)

	<b>VirtualUniverse</b>
	<b>Locale</b>
	<b>Group</b>
	<b>Leaf</b>
	<b>Node Component</b>
	<b>Other objects</b>

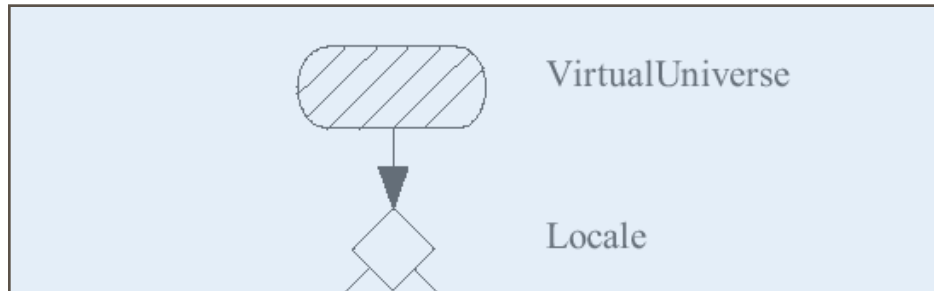
## Arcs (object relationships)

  
**Parent-child link**

  
**Reference**

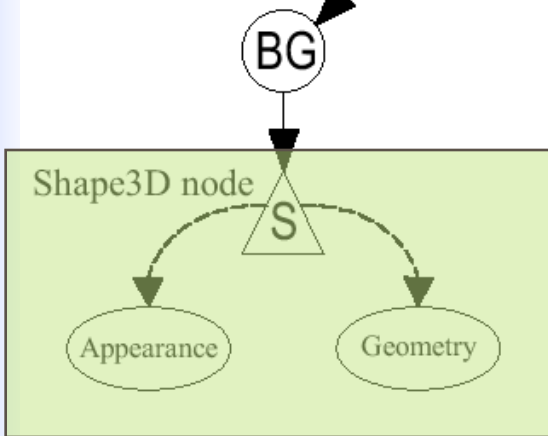




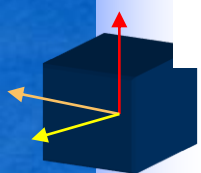
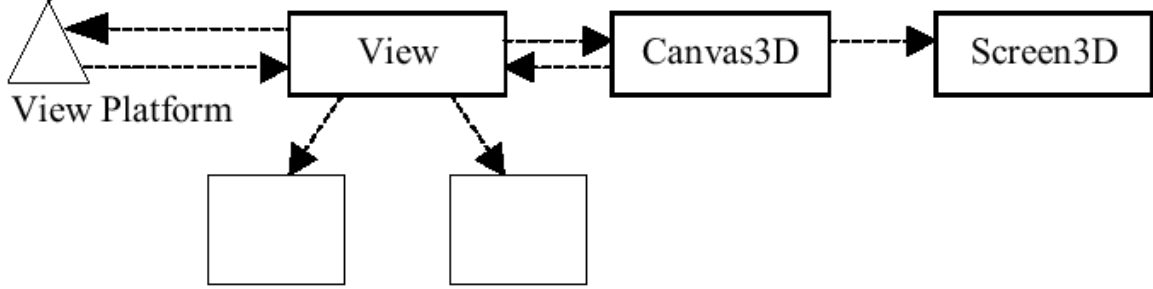
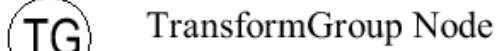


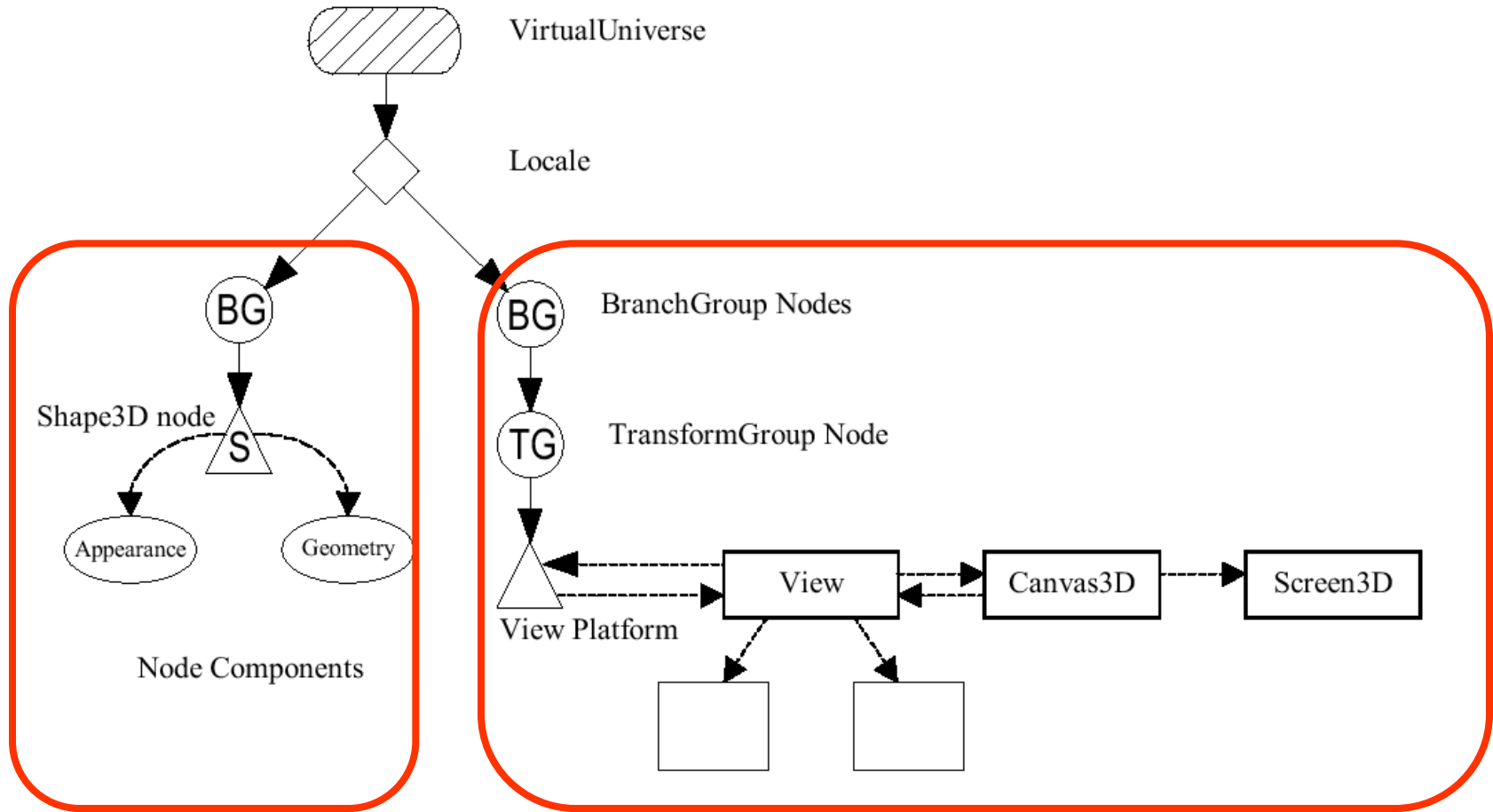
Together a Locale and a Virtual Universe compose a *SceneGraph* superstructure.

A reference associates a Node Component object with a Scene Graph Node .



Node Components

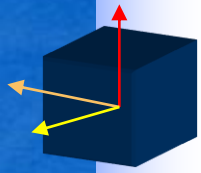




## Content Branch

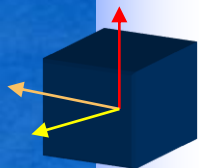
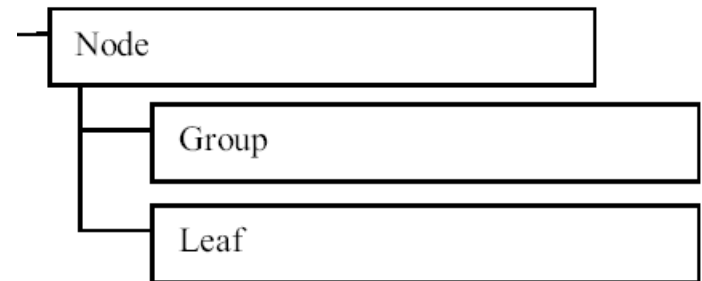
Introduction to Java3D

## View Branch



# Node

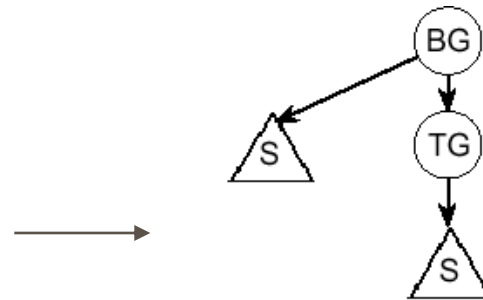
- Node Class is an abstract supper class of
  - Group
  - Leaf



# Scenegraph nodes

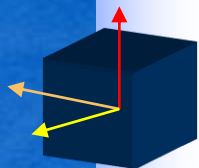
There are two types of nodes:

- **Group** : the primary role of a Group is to act as the parent of the other nodes, specially other Group nodes and Leaf nodes.



Groups may have children which are Leaf nodes or other Group nodes

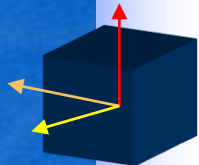
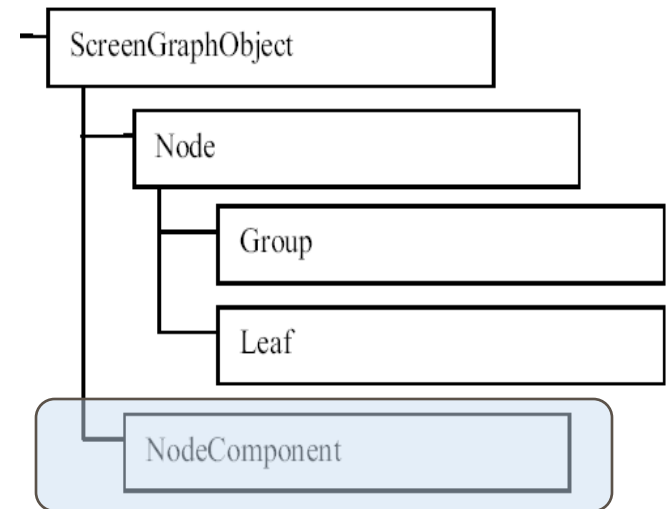
- **Leaf** : leaf nodes specify the shape, sound, and behavior of a scene graph object.



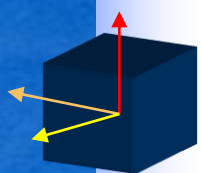
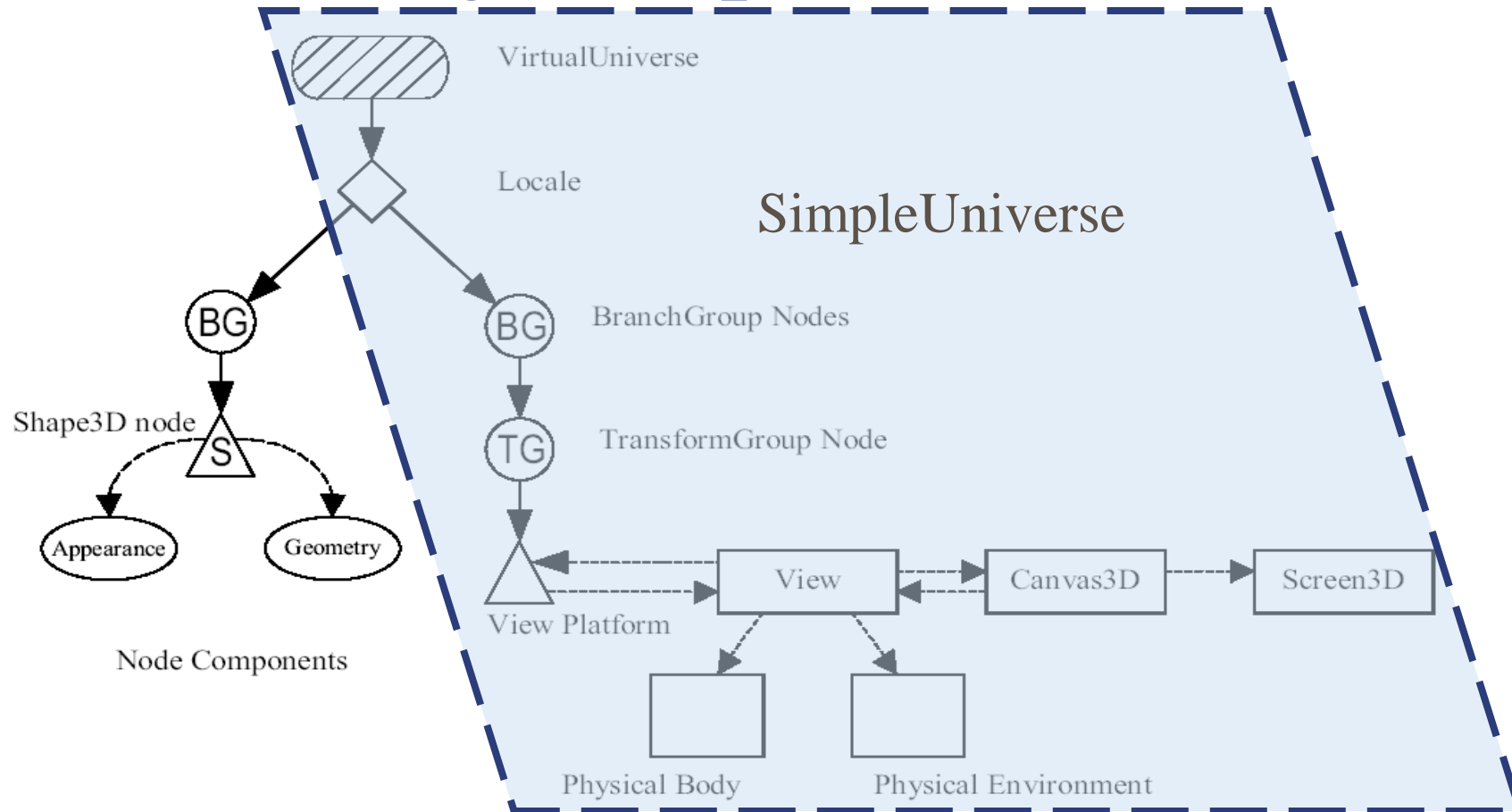
# NodeComponent

- NodeComponent is
  - not part of Scene Graph
  - It is referenced by it
- Used to specify
  - Geometry
  - Appearance
  - Texture
  - Material

Which are Properties of  
Shape3D leaf



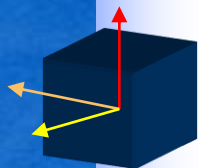
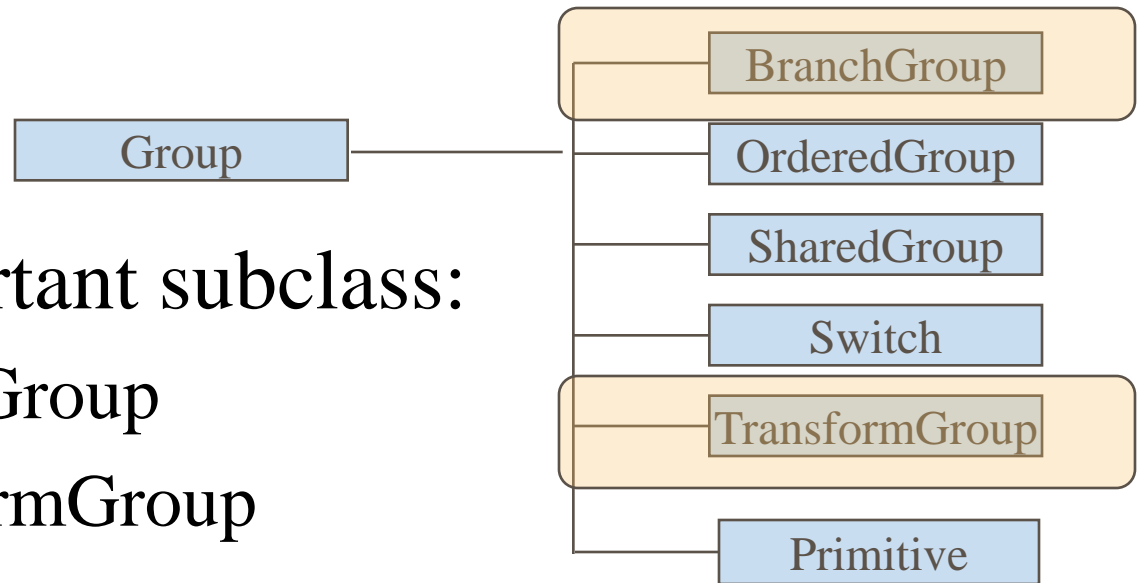
# Make things simpler



# Deeper look at Group class

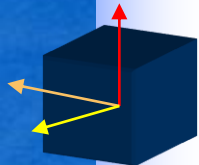
- Used in specifying the location and orientation of visual objects in the virtual universe.

- Two important subclass:
  - BranchGroup
  - TransformGroup



# Deeper look at Group class

- **BranchGroup Class**
  - The only object allowed to be children of Locale objects.
- **TransformGroup Class**
  - hold geometric transformations such as translation and rotation.

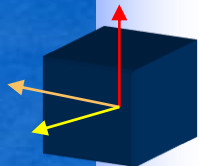
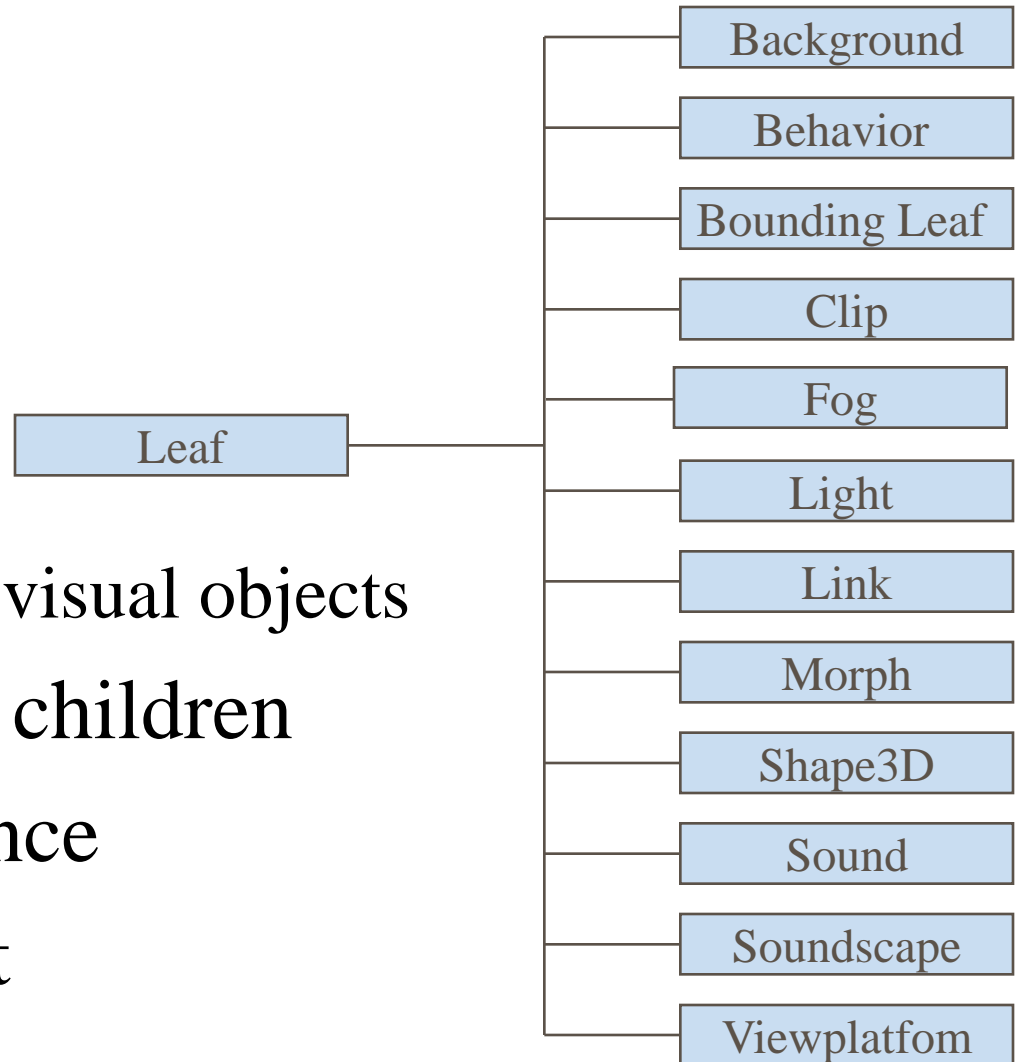




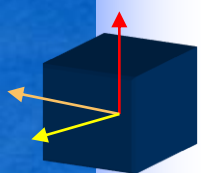
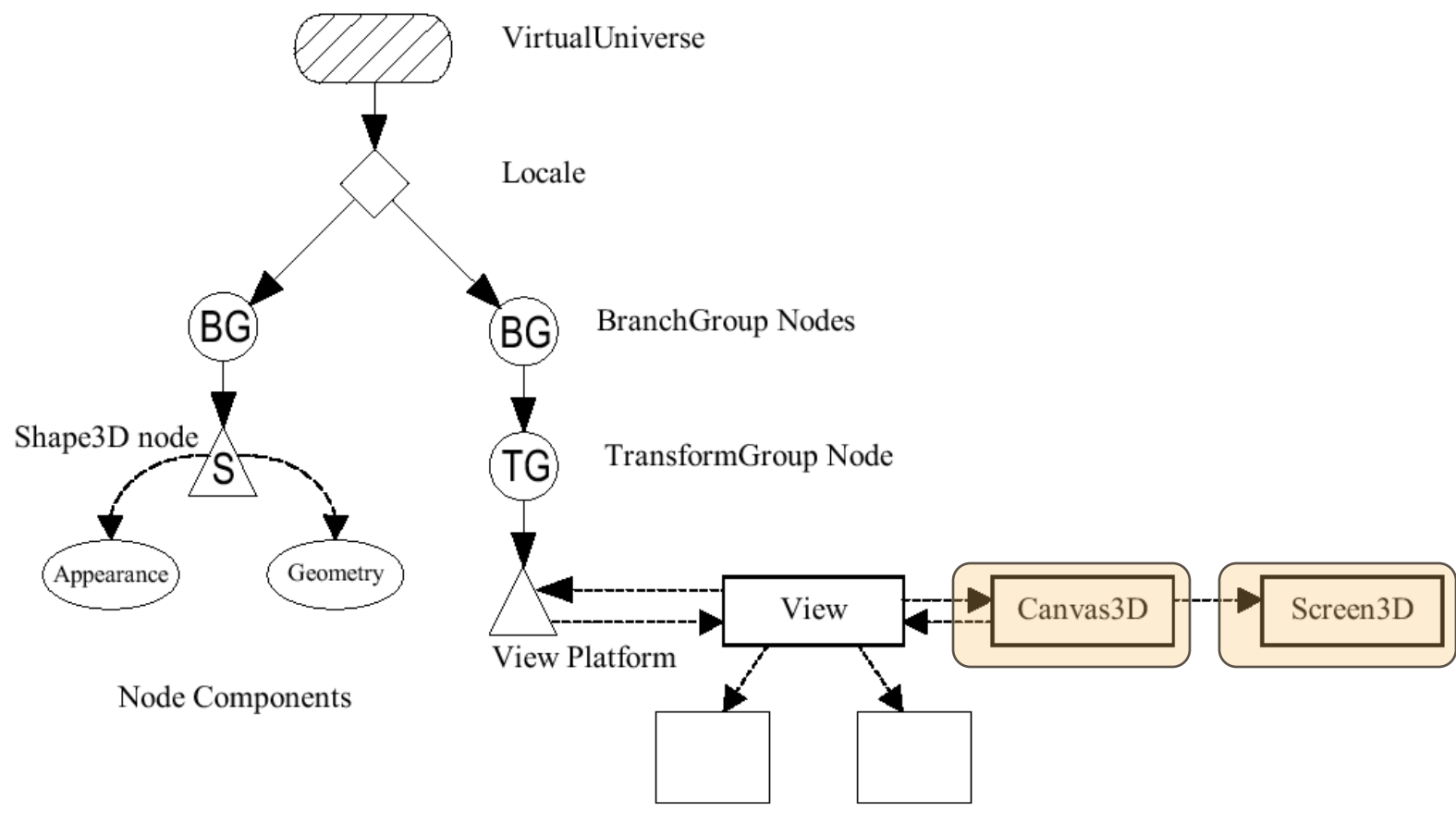
# leaf Class

- Specifyse the
  - shape,
  - sound,
  - behavior of visual objects
- May not have children

But could reference  
NodeComponent

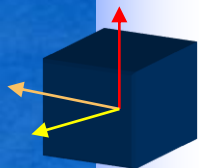
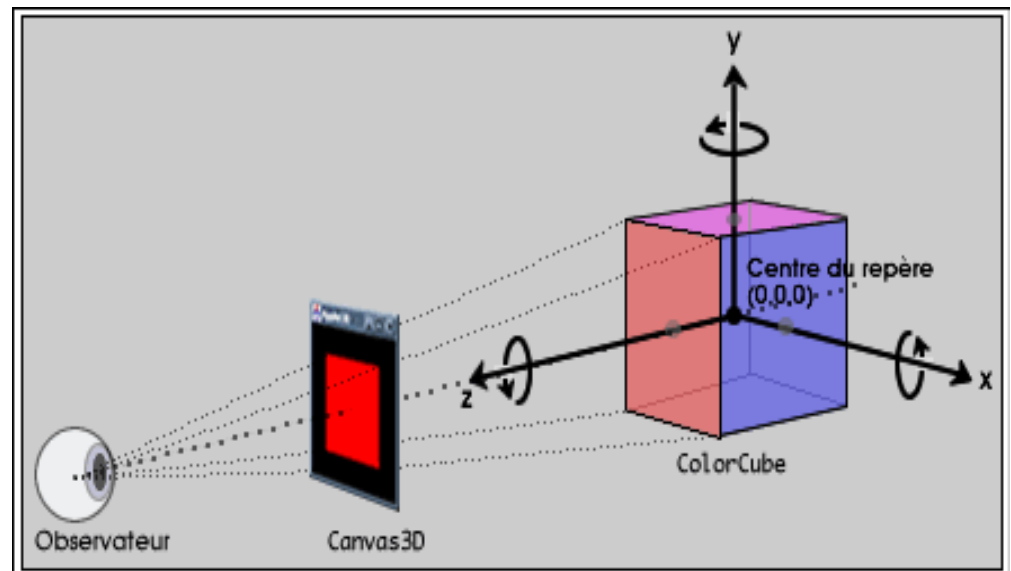


# Canvas3D & Screen3D



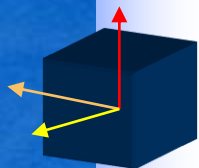
# Canvas3D

- Extends Canvas class from java.awt
- Java Converts Canvas3D size in pixels to physical world size in meters.
- Need at least one.



# Screen3D

- Works hand in hand with Canvas3D
- Provides a 3D version of AWT
- Java 3D supports more than one view at a time.



# Java3D API Organization

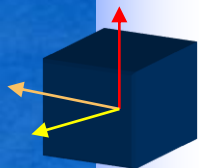
The API has core classes and utility classes

## Core Classes

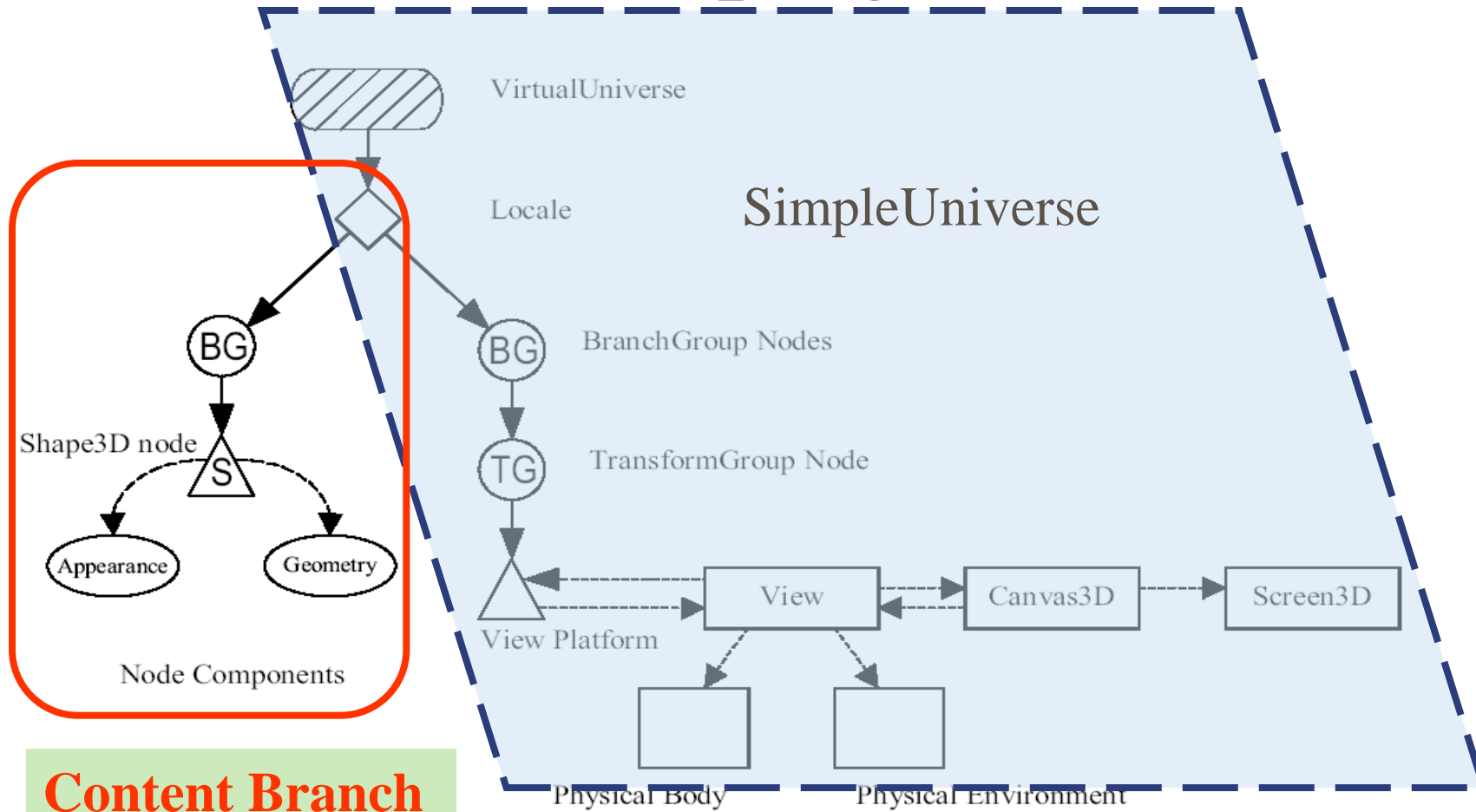
- `javax.media.j3d` package
- lowest level classes required for Java3D programming

## Utility Classes

- `com.sun.j3d.utils` package
- convenient and powerful additions to the core



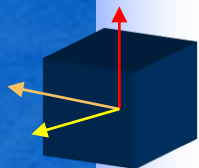
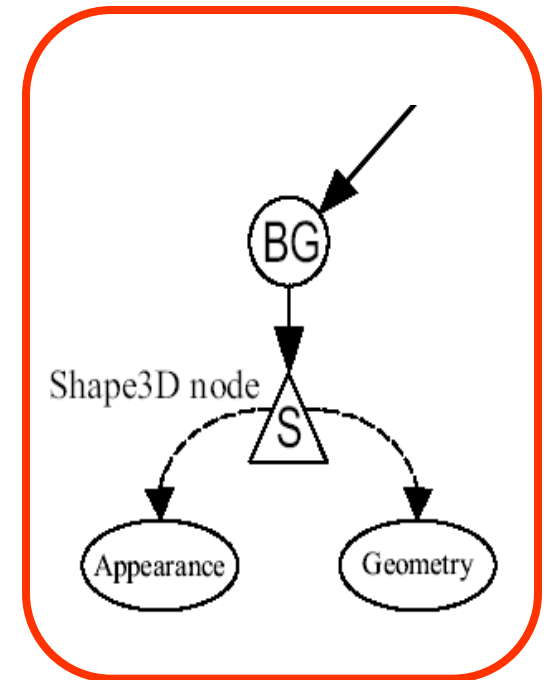
# The focus of the programmer



## Content Branch

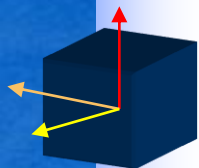
# Shape3D

- Has two NodeComponent
  - Geometry
    - made up of coordinates (vertices)
  - Appearance
    - e.g. color, texture, transparency, material



# Geometry

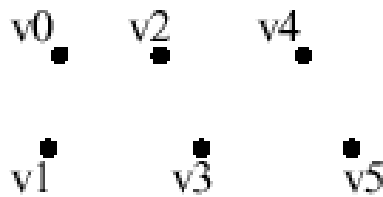
- There are several predefined shape classes in `com.sun.j3d.utils.geometry`:
  - Box, Sphere, Cone, Cylinder
- Usually these classes are insufficient, and a shape's geometry must be built by connecting vertices.



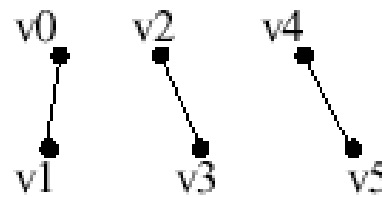


# Building Geometry

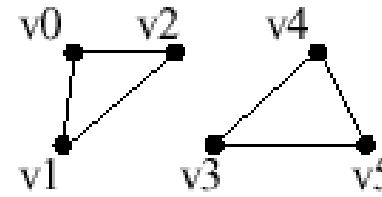
- The `GeometryArray` class is the parent for several useful geometry subclasses



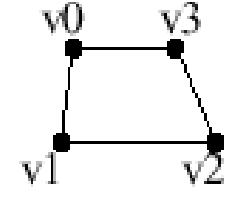
PointArray



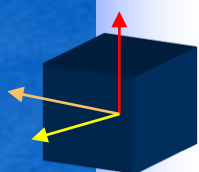
LineArray



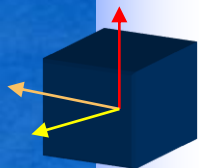
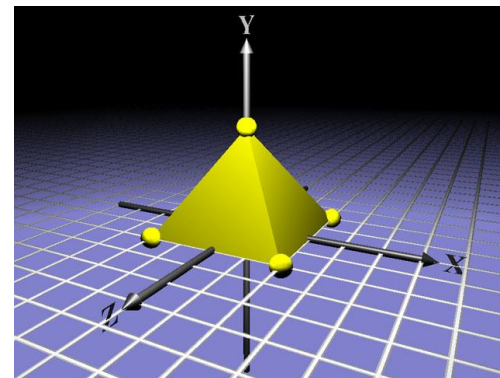
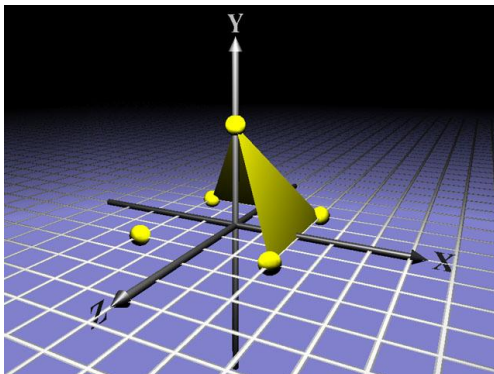
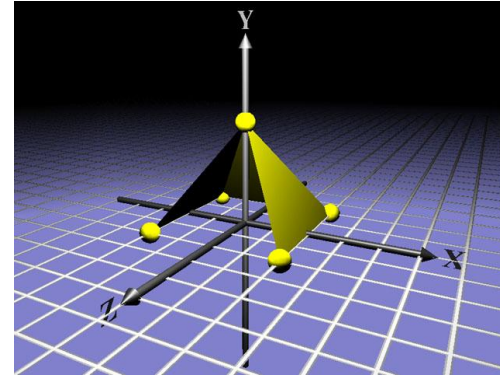
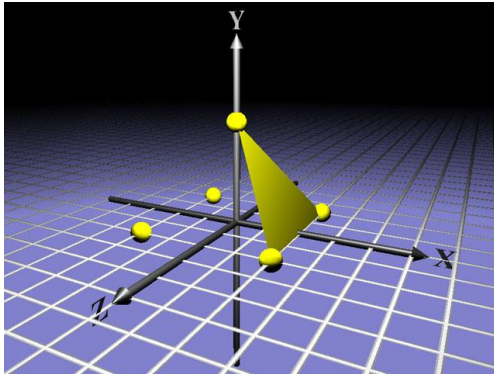
TriangleArray



QuadArray

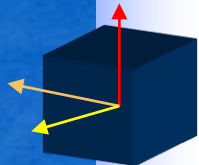


# Pyramid Geometry:



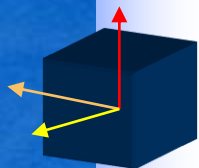
# Easier way to create geometry?

- Use other applications
- Load into Java 3D
  - Benefits
    - It takes far less time
  - Problem
    - Lose some factuality



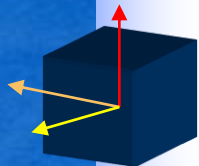
# Publicly Available Java 3D loaders

■ File	Format Description
■ 3DS	3D-Studio
■ COB	Caligari trueSpace
■ DEM	Digital Elevation Map
■ DXF	AutoCAD Drawing Interchange File
■ IOB	Imagine
■ LWS	Lightwave Scene Format
■ NFF	WorldToolKit NFF format
■ OBJ	Wavefront
■ PDB	Protein Data Bank
■ PLAY	PLAY
■ SLD	Solid Works (prt and asm files)
■ VRT	Superscape VRT
■ VTK	Visual Toolkit
■ WRL	Virtual Reality Modeling Language



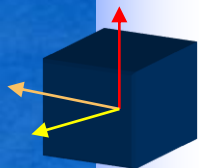
# Recipe for writing a Java3D program

- The basic outline of Java 3D program development consists of seven steps :
  1. Create a Canvas3D object
  2. Create a VirtualUniverse object
  3. Create a Locale object, attaching it to the VirtualUniverse object
  4. Construct a view branch graph
    1. Create a View object
    2. Create a ViewPlatform object
    3. Create a PhysicalBody object
    4. Create a PhysicalEnvironment object
    5. Attach ViewPlatform, PhysicalBody, PhysicalEnvironment, and Canvas3D objects to View object
  5. Construct content branch graph(s)
  6. Compile branch graph(s)
  7. Insert subgraphs into the Locale



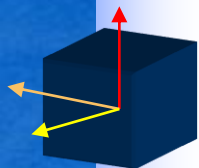
# Simple Recipe

- Using the SimpleUniverse class in Java 3D program reduces the time and effort needed to create the view branch Graph.
- The steps 1,2,3,4,and 7 create a Simple Universe( code for creating a SimpleUniverse: `SimpleUniverse ( )` )



# Simple Recipe

- 1. Create a Canvas3D Object
- 2. Create a SimpleUniverse object which references the earlier Canvas3D object
  - a. Customize the SimpleUniverse object
- 3. Construct content branch
- 4. Compile content branch graph
- 5. Insert content branch graph into the Locale of the SimpleUniverse



# HelloJava3D Class

```
public class HelloJava3Da extends Applet {
```

```
    public HelloJava3Da() {
```

```
        setLayout(new BorderLayout());
```

```
        GraphicsConfiguration config =
```

```
            SimpleUniverse .getPreferredConfiguraton( );
```

```
        Canvas3D canvas3D = new Canvas3D (config) ;
```

```
        add("Center", canvas3D);
```

```
        BranchGroup scene = createSceneGraph();
```

```
        scene.compile();
```

```
// SimpleUniverse is a convenience Utility class
```

```
SimpleUniverse SimpleU = new SimpleUniverse (canvas 3D) ;
```

```
// This move the ViewPlatform back a bit so the
```

```
// objects in the scene can be viewed .
```

```
SimpleU.getViewingPatform().setNominalViewingTransform();
```

```
simpleU.addBranchGraph (scene);
```

```
} // end of HelloJava 3Da (constuctor)
```

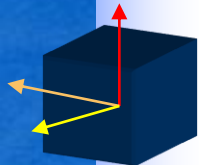
1- Create a canvas3D

4- Compile ContentBranch Graph

2- Create a Simpleuniverse

3- Customize Simpleuniverse

5- Insert ContentBranchGraph into the Locale





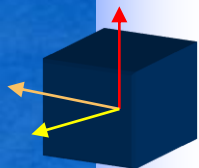
# Some terminologies in Java3D

## ■ *Become live* :

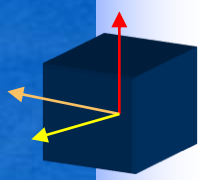
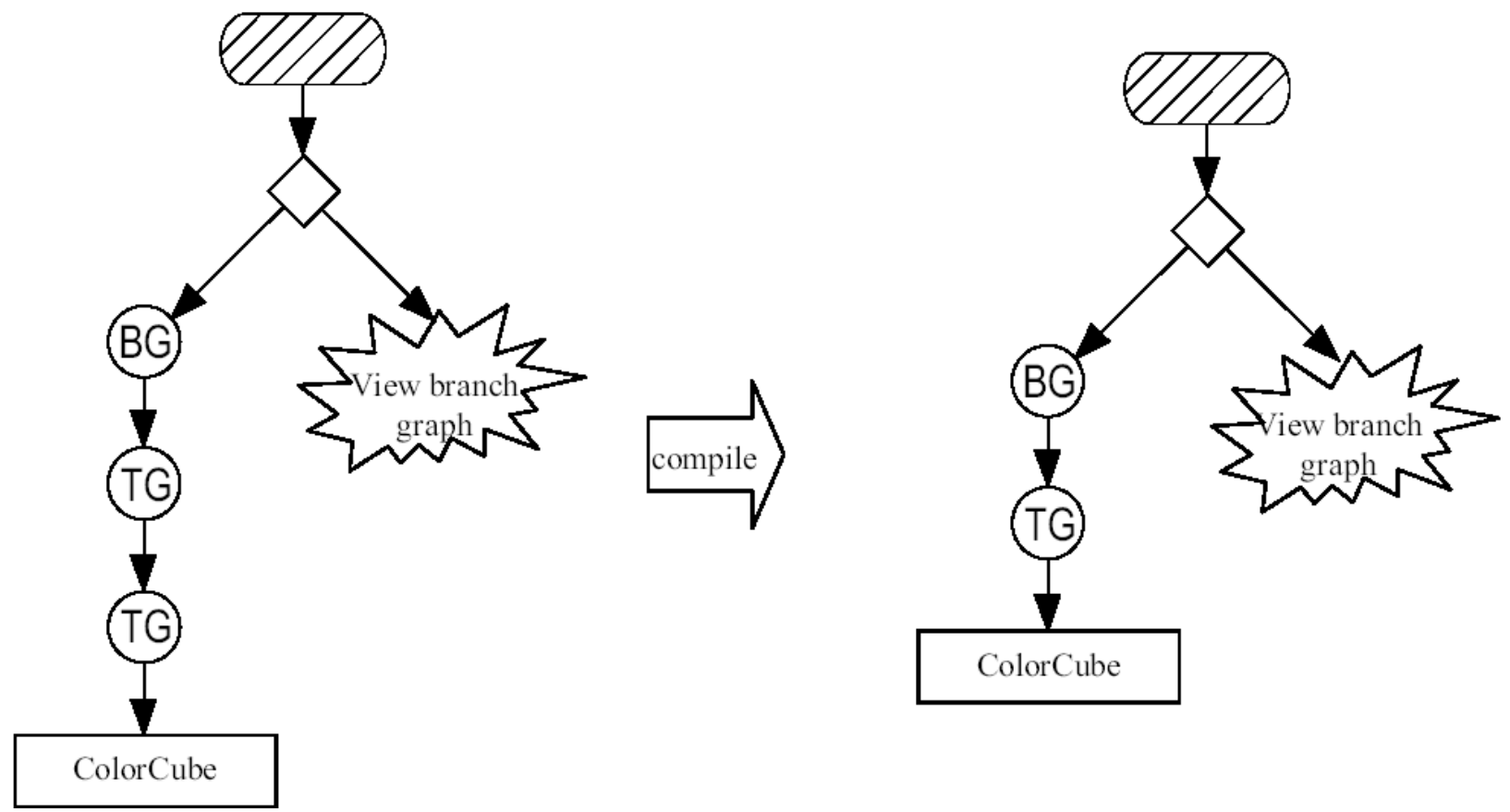
a Branch Graph becomes live as soon as it is attached to a scene graph. Each object of the Branch Graph are subject to being rendered.

## ■ *Compiling*

a BranchGroup converts an object and all its ancestors to a more efficient one from the rendered.



# Compiling



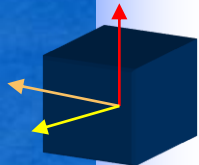
# OpenGL Vs Java3D

## OpenGL

- low level; procedural
- immediate mode rendering

## Java3D

- high level; OO
- different rendering modes!

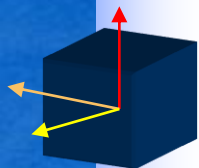


# Java 3D vs other API's

## Low level APIs

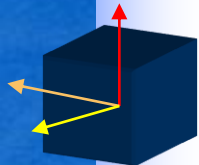
- Fast
- less memory required

- Java3D
- Java: language of the Internet
- portability
  - write once “render” everywhere
- application programmer can concentrate on objects
- web-based, powerful tool, runs inside the browser



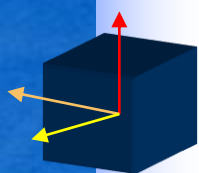
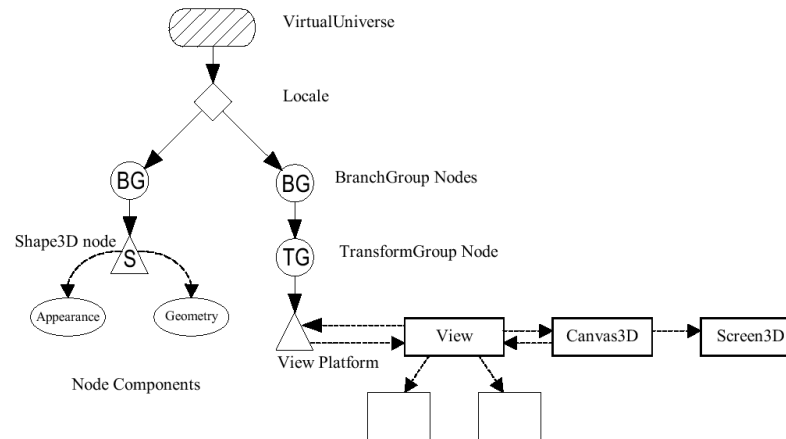
# Problems

- Very limited literature compared to OpenGL
  - Because of its complexity there is a higher demand for detailed references.
- Not really platform independent
  - Works fine on UNIX and windows



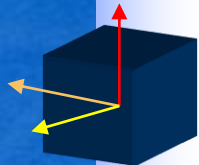
# Summery

- A high-level (Object Oriented) API for building interactive 3D applications and applets
- The Scene Graph is the skeleton of Java 3D



# Summery

- ***Virtual Universe und Locale Class:***  
are derived respectively from Universe and Locale Object in a scene Graph.
- ***SceneGraph Object Class:***  
is the base class for nearly every object that appears in a Java3D scene Graph. This abstract class contains node and node component objects.
- ***Scene Graph Viewing Object Classes:***  
included five classes that are used in the viewing parameters of scene graphs (Canvas3D, Screen3D, View, PhysicalBody, PhysicalEnvironment).
- ***Simple Universe utility class:***  
is used by the Java3D developers to create a Java3D program without dealing with Viewing object Classes.



Thank you!

