



# Introduction to JAVA 3D


Aslan Neishaboori  
CS491B


 Introduction to Java3D 1



## Contents:


- The 3D world
- Basics of Java 3D (Scene Graph)
- Important classes
- Simple recipe
- Example code
- Some General facts
- & Summery


 Introduction to Java3D 2



## How do we see the world?


Left eye
Right eye


 Introduction to Java3D 3



## 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.


 Introduction to Java3D 4



## 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


 Introduction to Java3D 5



## 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

 Introduction to Java3D 6

## 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

Introduction to Java3D 7

## A typical Scene Graph

The diagram illustrates a typical scene graph structure. At the top is the **Virtual Universe** (represented by a hatched rectangle). Below it is a **Locale** (diamond). Two **BranchGroup Nodes** (BG, circles) are connected to the Locale. The left BG node is a **Shape3D node** (S, triangle) which contains **Node Components**: **Appearance** (oval) and **Geometry** (oval). The right BG node is a **TransformGroup Node** (TG, triangle) which contains a **View Platform** (rectangle) and a **View** (rectangle). The View Platform is connected to the View. The View is connected to **Canvas3D** (rectangle), which is connected to **Screen3D** (rectangle). There are also two empty rectangles below the View Platform.

Introduction to Java3D 8

## Scene Graph Symbols

Nodes and Node Components (objects)	Arcs (object relationships)
Virtual Universe	Parent-child link
Locale	Reference
Group	
Leaf	
Node Component	
Other objects	

Introduction to Java3D 9

The diagram is identical to slide 8. It includes two text boxes:
 

- One box points to the Locale and Virtual Universe nodes: "Together a Locale and a Virtual Universe compose a *SceneGraph* superstructure."
- Another box points to the Shape3D node: "A reference associates a Node Component object with a Scene Graph Node."

Introduction to Java3D 10

The diagram is identical to slide 8. Two rounded rectangles highlight specific parts of the graph:
 

- The left rounded rectangle, labeled **Content Branch**, encloses the Shape3D node and its associated Appearance and Geometry node components.
- The right rounded rectangle, labeled **View Branch**, encloses the TransformGroup Node, the View Platform, the View, and the Canvas3D and Screen3D nodes.

Introduction to Java3D 11

## Node

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

The class hierarchy diagram shows a box labeled **Node** at the top. Below it are two boxes: **Group** and **Leaf**. Arrows point from Group and Leaf up to Node, indicating inheritance.

Introduction to Java3D 12

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

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

Introduction to Java3D 13

## 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

Introduction to Java3D 14

## Make things simpler

Introduction to Java3D 15

## Deeper look at Group class

- Used in specifying the location and orientation of visual objects in the virtual universe.
- Two important subclass:
  - BranchGroup
  - TransformGroup

Introduction to Java3D 16

## 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.

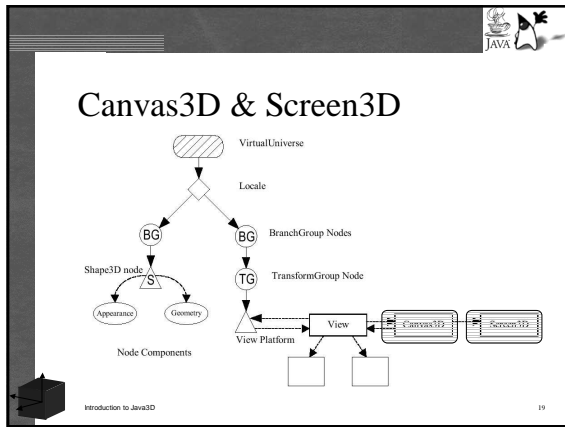
Introduction to Java3D 17

## leaf Class

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

But could reference NodeComponent

Introduction to Java3D 18



## Canvas3D

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

Introduction to Java3D 20

## Screen3D

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

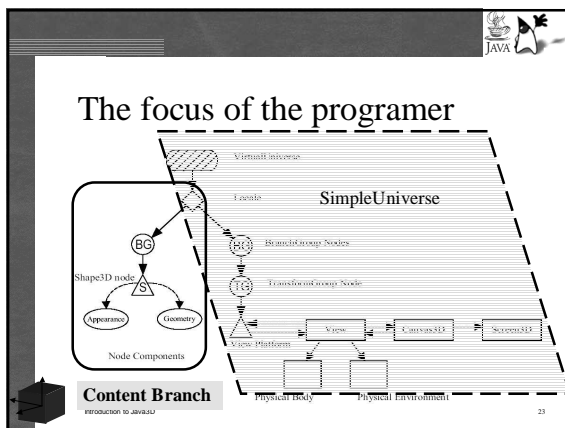
Introduction to Java3D 21

## Java3D API Organization

The API has core classes and utility classes

Core Classes	Utility Classes
■ javax.media.j3d package	■ com.sun.j3d.utils package
■ lowest level classes required for Java3D programming	■ convenient and powerful additions to the core


Introduction to Java3D 22



## Shape3D

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


Introduction to Java3D 24

JAVA 

## 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.

Introduction to Java3D 25


JAVA 

## Building Geometry


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




PointArray



LineArray




TriangleArray

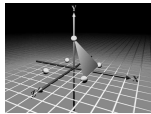
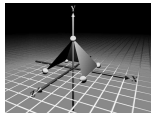
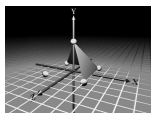
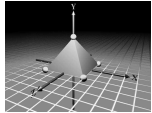


QuadArray


Introduction to Java3D 26

JAVA 

## Pyramid Geometry:


Introduction to Java3D 27

JAVA 

## Easier way to create geometry?

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


Introduction to Java3D 28

JAVA 

## 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	Lighwave 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

Introduction to Java3D 29

JAVA 

## 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`

Introduction to Java3D 30

**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 ( ) )

Introduction to Java3D 31

**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

Introduction to Java3D 32

**HelloJava3D Class**

```

public class HelloJava3Da extends Applet {
    public HelloJava3Da() {
        setLayout(new BorderLayout());
        GraphicsConfiguration config
        = SimpleUniverse.getDefaultConfiguration();
        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 move the ViewPlatform back a bit so the
    // objects in the scene can be viewed .
    SimpleU.getViewingPlatform().setNominalViewingTransform();
    simpleU.addBranchGraph (scene);
    // end of HelloJava 3Da (constructor)
  }
  
```

Annotations in the code:

- 1- Create a canvas3D (points to `Canvas3D canvas3D = new Canvas3D(config);`)
- 2- Create a Simpleuniverse (points to `SimpleUniverse SimpleU = new SimpleUniverse (canvas3D);`)
- 3- Customize Simpleuniverse (points to `SimpleU.getViewingPlatform().setNominalViewingTransform();`)
- 4- Compile ContentBranch Graph (points to `scene.compile();`)
- 5- Insert ContentBranchGraph into the Locale (points to `simpleU.addBranchGraph (scene);`)

Introduction to Java3D 33

**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.

Introduction to Java3D 34


**Compiling**

Introduction to Java3D 35

**OpenGL Vs Java3D**

<u>OpenGL</u>	<u>Java3D</u>
<ul style="list-style-type: none"> <li>low level; procedural</li> <li>immediate mode rendering</li> </ul>	<ul style="list-style-type: none"> <li>high level; OO</li> <li>different rendering modes!</li> </ul>

Introduction to Java3D 36



## 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


Introduction to Java3D 37



## Problems

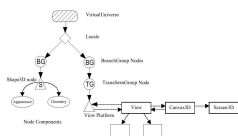
- Very limited literature compared to OpenGL
  - Because of its complexity there is a higher demand for detailed references.
- Not really platform independent
  - Works fine on UNIX and windows

Introduction to Java3D 38




## Summery

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




Introduction to Java3D 39



## 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.

Introduction to Java3D 40



## Thank you!

Introduction to Java3D 41