Introduction to JAVA 3D

Aslan Neishaboori
CS491B

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

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 Java 3D 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)
- Arcs (object relationships)

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.
- **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

- Specify the
  - shape,
  - sound,
  - behavior of visual objects
- May not have children
- But could reference NodeComponent
In Introduction to Java3D

Canvas3D & Screen3D
- Extends Canvas class from java.awt
- Java Converts Canvas3D size in pixels to physical world size in meters.
- Need at least one.

Canvas3D

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
- java.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

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

- Fbx: 3D Studio
- C3D: Colibri, Xfig
- DEM: Digital Elevation Map
- DSF: AutoCAD Drawing Interchange File
- JDB: Imagine
- LWS: LightWave Scene Format
- NIF: WorldsOfType NIF format
- OBJ: Wavefront
- PDS: Precis Data Bank
- PLAY: PLAY
- SLD: Solid Works (p3 and aus files)
- VRZ: StereoViz
- 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:
     i. Create a View object
     ii. Create a ViewPlatform object
     iii. Create a PhysicalBody object
     iv. Create a PhysicalEnvironment object
     v. Attach ViewPlatform, PhysicalBody, PhysicalEnvironment, and Camera3D objects to View object
  5. Construct content branch graphs
  6. Compile branch graphs
  7. Insert subgraphs into the Locale
In introduction to Java3D

Simple Recipe

- Using the SimpleUniverse class in Java3D program reduces the time and effort needed to create the view branch Graph.
- The steps 1,2,3,4, and 7 create a SimpleUniverse class for creating a SimpleUniverse: SimpleUniverse()

HelloJava3D Class

```java
public class HelloJava3D extends Applet {
    public HelloJava3D() {
        setLayout(new BorderLayout());
        GraphicsConfiguration config = SimpleUniverse.getPreferredConfiguration();
        Canvas3D canvas3D = new Canvas3D(config);
        add("Center", canvas3D);
        BranchGroupsscene = 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);
    }
}
```

Some terminologies in Java3D

- **Become live**: a BranchGraph becomes live as soon as it is attached to a scene graph. Each object of the BranchGraph 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
- **Java3D**: high level; object-oriented rendering

Different rendering modes!
In Introduction to Java3D:

Java 3D vs other API’s
- **Low level APIs**
  - **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

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

Thank you!