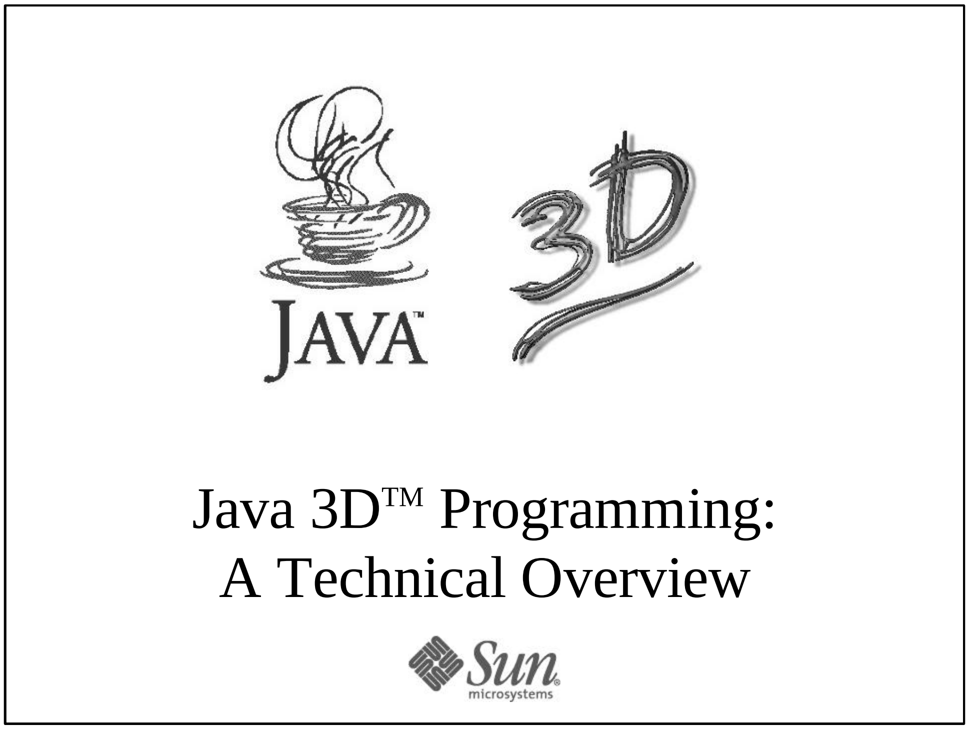
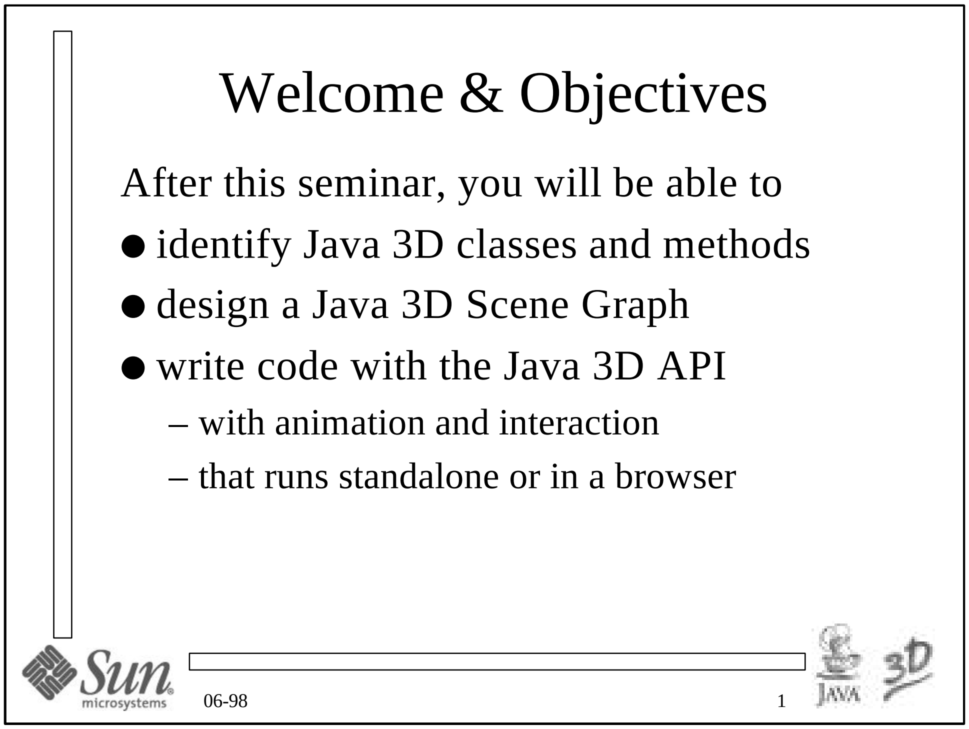


Java 3D™ Programming: A Technical Overview

Student Notes

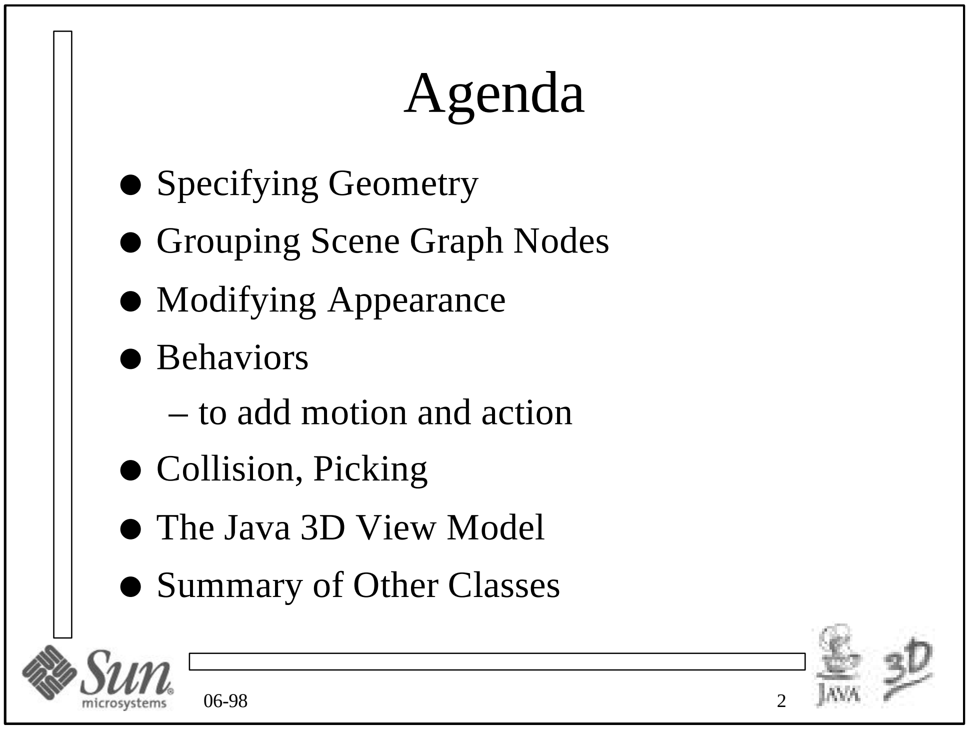


Welcome & Objectives

After this seminar, you will be able to

- identify Java 3D classes and methods
- design a Java 3D Scene Graph
- write code with the Java 3D API
  - with animation and interaction
  - that runs standalone or in a browser

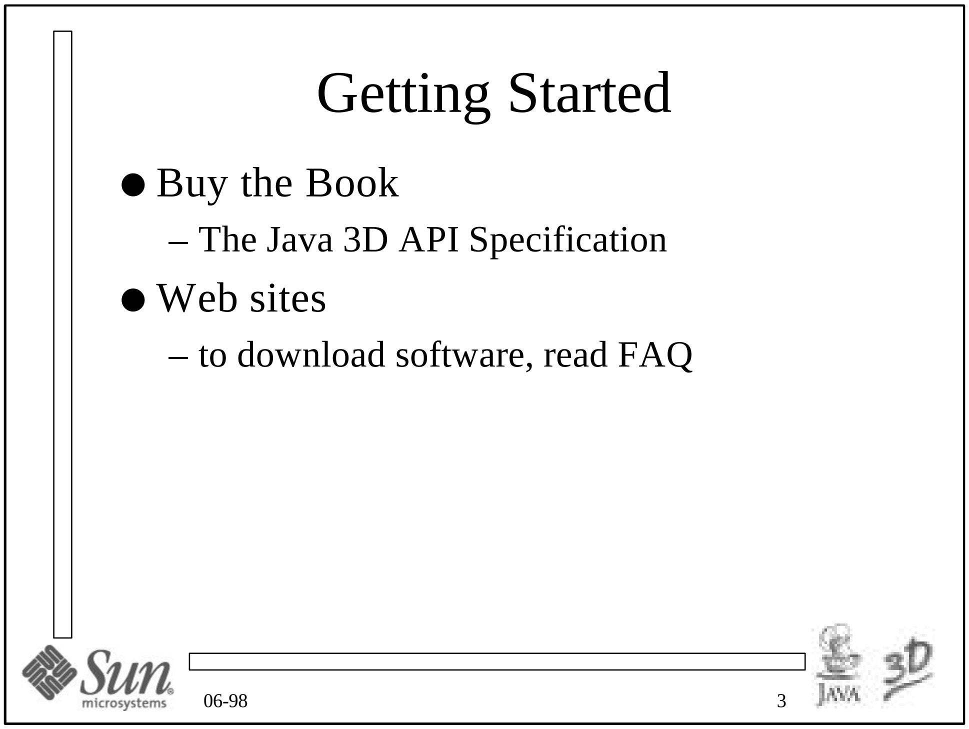
Student Notes



Agenda

- Specifying Geometry
- Grouping Scene Graph Nodes
- Modifying Appearance
- Behaviors
  - to add motion and action
- Collision, Picking
- The Java 3D View Model
- Summary of Other Classes

Student Notes



Getting Started

- Buy the Book
  - The Java 3D API Specification
- Web sites
  - to download software, read FAQ

Student Notes

Sowizral, Rushforth, Deering, The Java 3D API Specification (Addison-Wesley, 1998; ISBN 0-201-32576-4)

Java 3D API pages

http://java.sun.com/products/java-media/3D

http://java.sun.com/products/java-media/3D/forDevelopers/java3dfaq.html

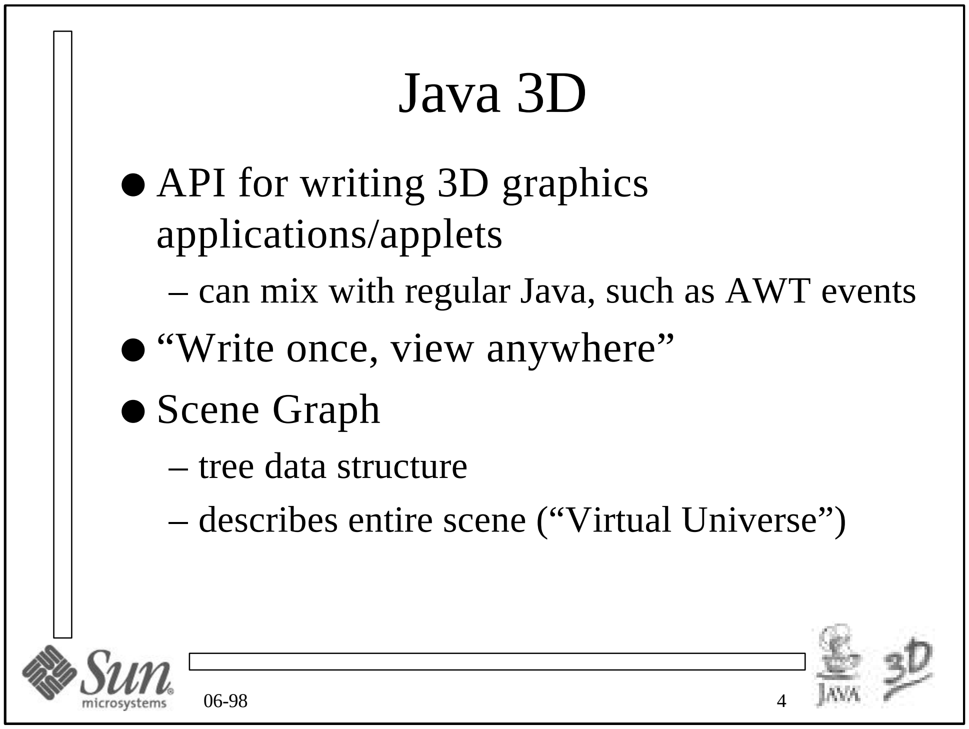
http://www.sun.com/desktop/java3d

The Java 3D Repository

http://java3d.sdsc.edu/

Java 3D Land

http://www.tomco.net/~raf/java3d.html

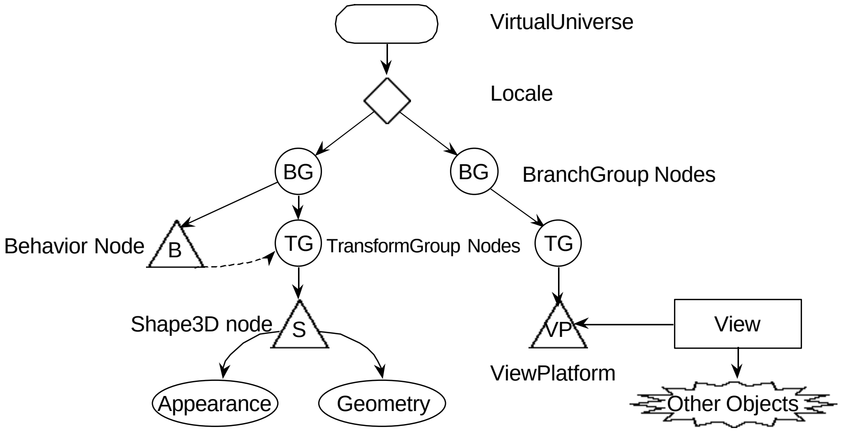


Java 3D

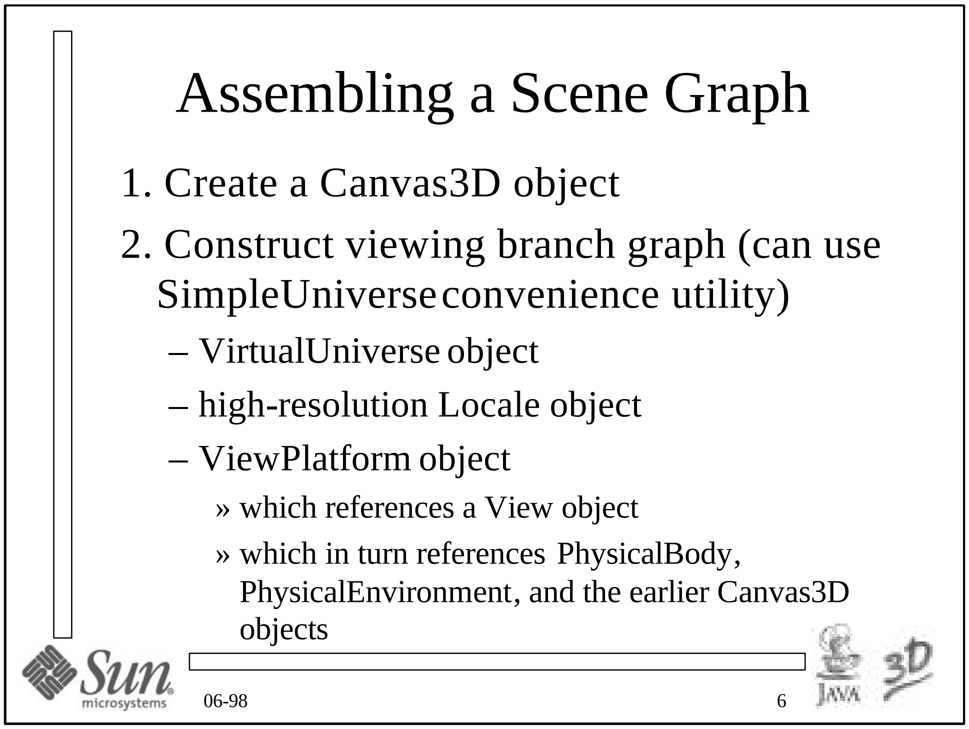
- API for writing 3D graphics applications/applets
  - – can mix with regular Java, such as AWT events
- “Write once, view anywhere”
- Scene Graph
  - – tree data structure
  - – describes entire scene (“Virtual Universe”)

Student Notes

Simple Scene Graph Example



Student Notes



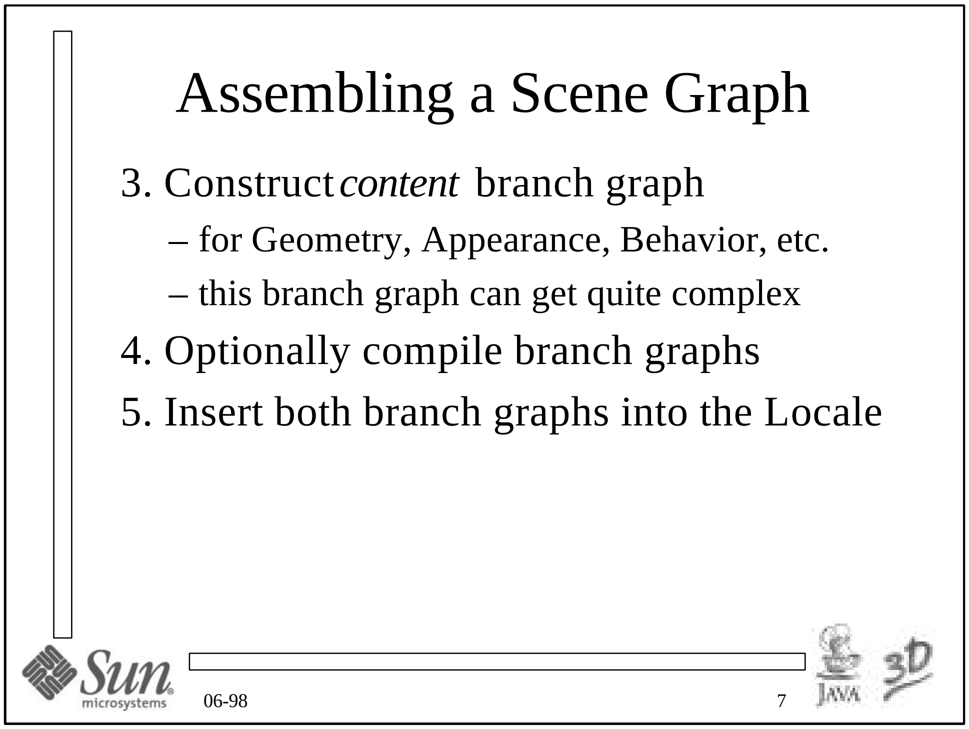
Assembling a Scene Graph

1. Create a Canvas3D object
2. Construct viewing branch graph (can use SimpleUniverse convenience utility)
   - – VirtualUniverse object
   - – high-resolution Locale object
   - – ViewPlatform object
     - » which references a View object
     - » which in turn references PhysicalBody, PhysicalEnvironment, and the earlier Canvas3D objects

Student Notes

VirtualUniverse, Locale, ViewPlatform, View, PhysicalBody, PhysicalEnvironment, and Canvas3D are all Java 3D classes (in the package javax.media.j3d.*).

The SimpleUniverse convenience class (in the package com.sun.j3d.utils.universe.*) performs steps 2 & 3 for you. SimpleUniverse is all you'll need for the vast majority of your applications.

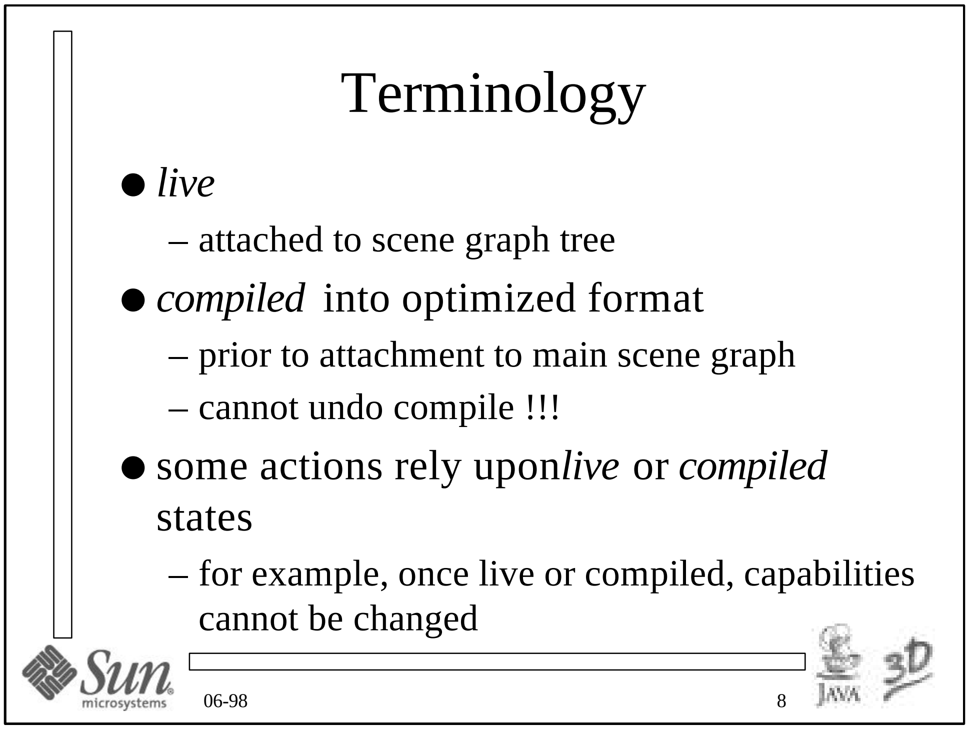


Assembling a Scene Graph

3. Construct *content* branch graph
   - for Geometry, Appearance, Behavior, etc.
   - this branch graph can get quite complex
4. Optionally compile branch graphs
5. Insert both branch graphs into the Locale

Student Notes

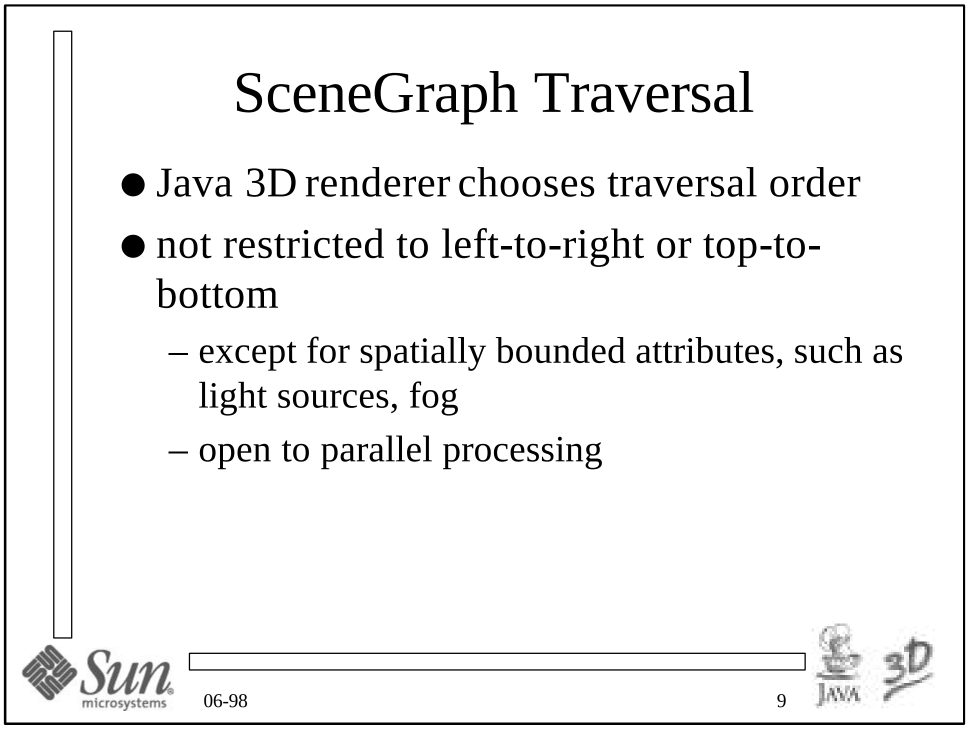
This course will initially focus on Step #3: how you can define 3D objects, their appearance, and their actions. Later, we will discuss the viewing platform, locales, and other aspects which comprise the SimpleUniverse branch graph.



Terminology

- *live*
  - – attached to scene graph tree
- *compiled* into optimized format
  - – prior to attachment to main scene graph
  - – cannot undo compile !!!
- some actions rely upon*live* or *compiled* states
  - – for example, once live or compiled, capabilities cannot be changed

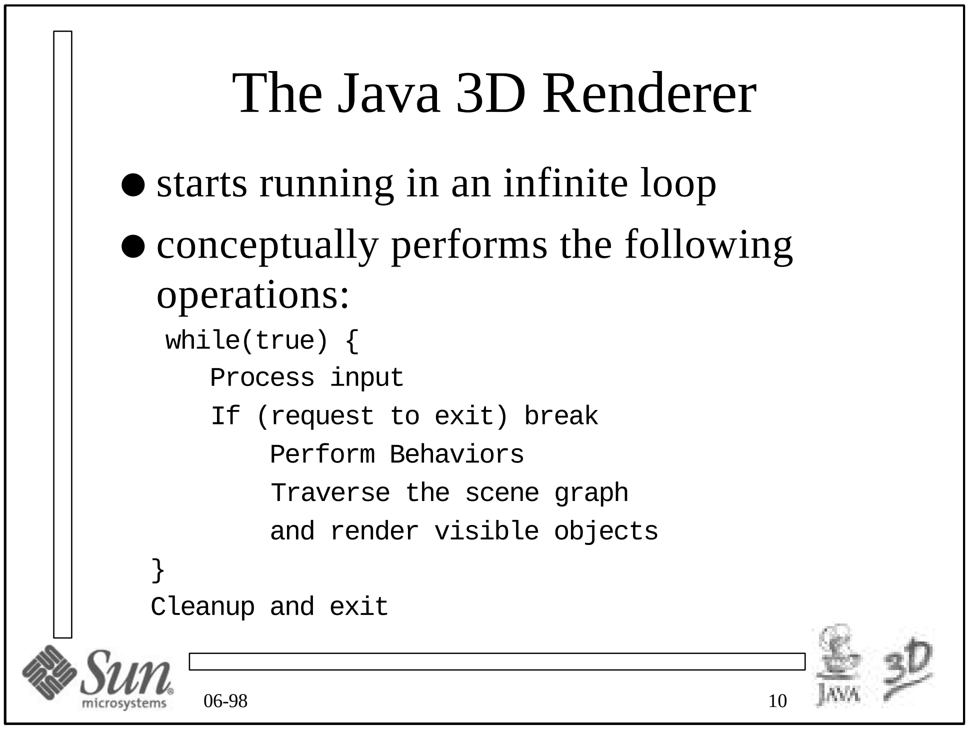
Student Notes



SceneGraph Traversal

- Java 3D renderer chooses traversal order
- not restricted to left-to-right or top-to-bottom
  - except for spatially bounded attributes, such as light sources, fog
  - open to parallel processing

Student Notes

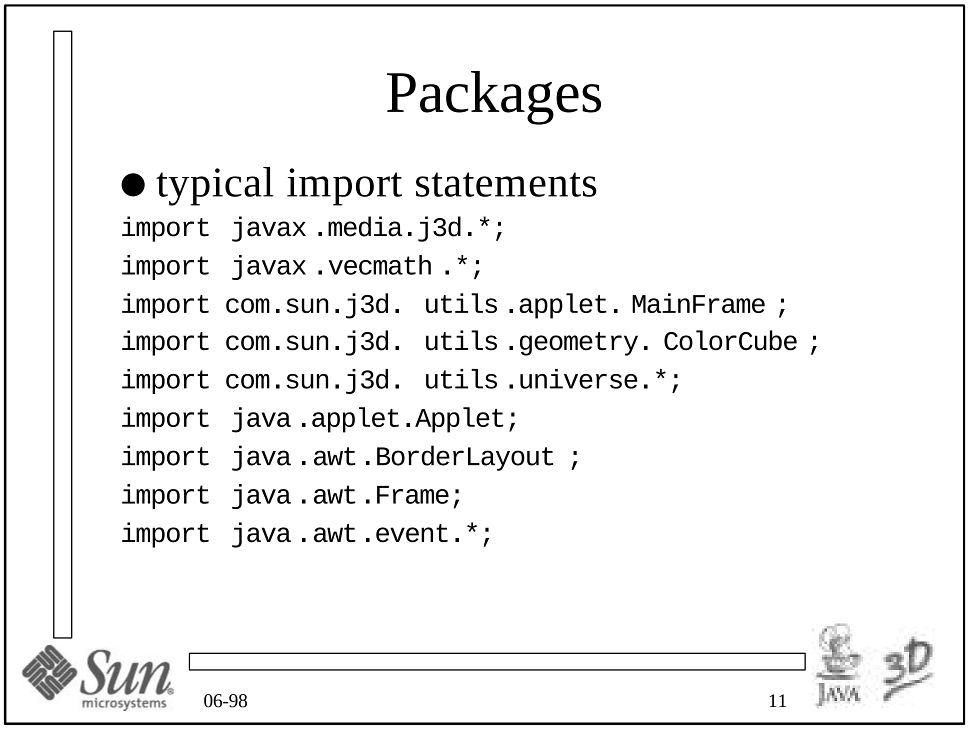


The Java 3D Renderer

- starts running in an infinite loop
- conceptually performs the following operations:

```
 while(true) {
    Process input
    If (request to exit) break
        Perform Behaviors
        Traverse the scene graph
        and render visible objects
}
Cleanup and exit
```

Student Notes



Packages

- typical import statements

```
import javax.media.j3d.*;
import javax.vecmath.*;
import com.sun.j3d.utils.applet.MainFrame;
import com.sun.j3d.utils.geometry.ColorCube;
import com.sun.j3d.utils.universe.*;
import java.applet.Applet;
import java.awt.BorderLayout;
import java.awt.Frame;
import java.awt.event.*;
```

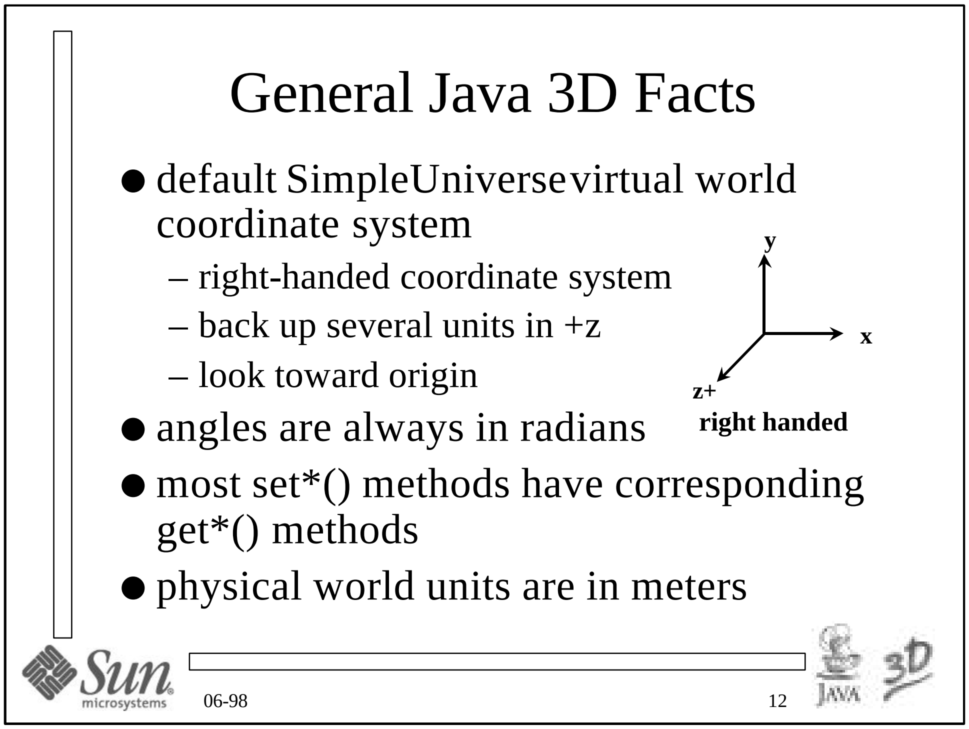
Student Notes

javax.media.j3d.* is the package which contains the entire Java 3D object hierarchy, including VirtualUniverse, Locale, ViewPlatform, View, PhysicalBody, PhysicalEnvironment, and Canvas3D.

javax.vecmath.* is the package which contains low-level mathematical constructs, such as vectors and matrices. This package is separated from javax.media.j3d.*, because they can be widely used outside of Java 3D. Although they are in a different package, the vecmath classes are frequently used in Java 3D classes and their methods.

javax.vecmath.* classes are identified by data type (float, double, etc.) and number of components (2D, 3D, or 4D vectors). Classes include Vector2f, Vector3f, Vector3d, Vector4d, Point3d, and Matrix3f. There are also classes for colors, texture coordinates, and quaternions.

com.sun.j3d.utils.*.* is the Convenience Utility library. There are several subdirectories here: including applet, geometry, ui, and universe. Classes here include SimpleUniverse and AWT helpers (to use input devices for picking or general manipulation). MainFrame allows Java classes to be run as either an applet or a standalone application. Also MainFrame adds an ActionListener, so the window closes gracefully from the window system pop-up menu.



General Java 3D Facts

- default SimpleUniverse virtual world coordinate system
  - right-handed coordinate system
  - back up several units in +z
  - look toward origin
- angles are always in radians
- most set*() methods have corresponding get*() methods
- physical world units are in meters

y
x
z+
right handed

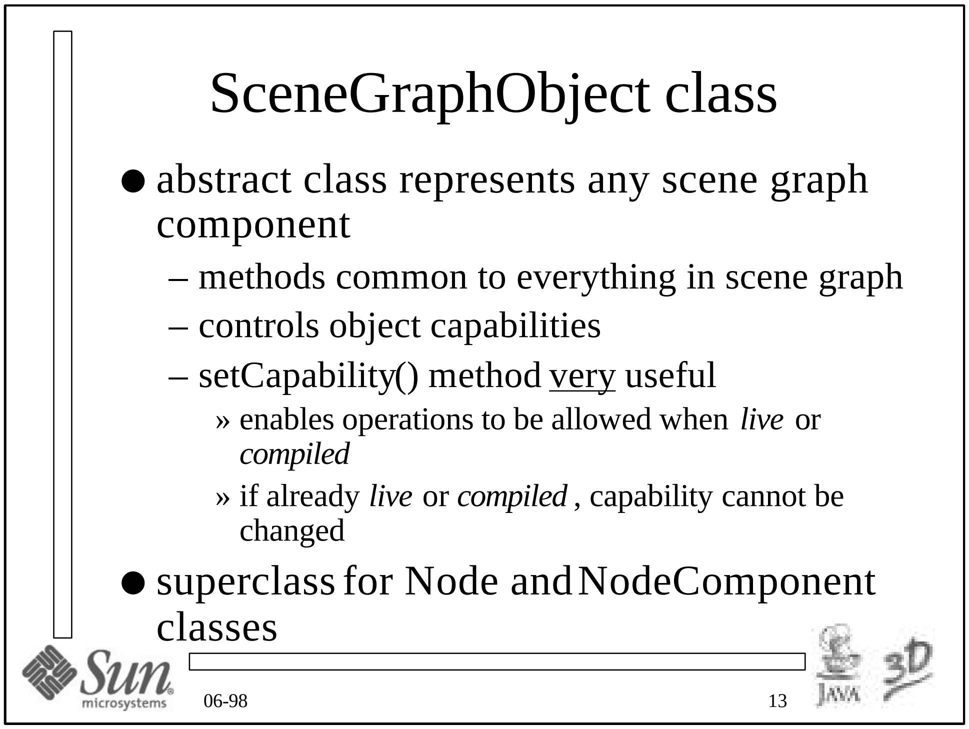
06-98

12

Student Notes

Java 3D also assumes:

- RGB color mode only; not color index
- double buffering exists and is enabled by default
- depth (z) buffering exists and is enabled by default



SceneGraphObject class

- abstract class represents any scene graph component
  - methods common to everything in scene graph
  - controls object capabilities
  - setCapability() method <u>very</u> useful
    - enables operations to be allowed when *live* or *compiled*
    - if already *live* or *compiled* , capability cannot be changed
- superclass for Node and NodeComponent classes

Student Notes

Java 3D Object Hierarchy

- SceneGraphObject
  - Node
    - Group
    - Leaf
  - NodeComponent

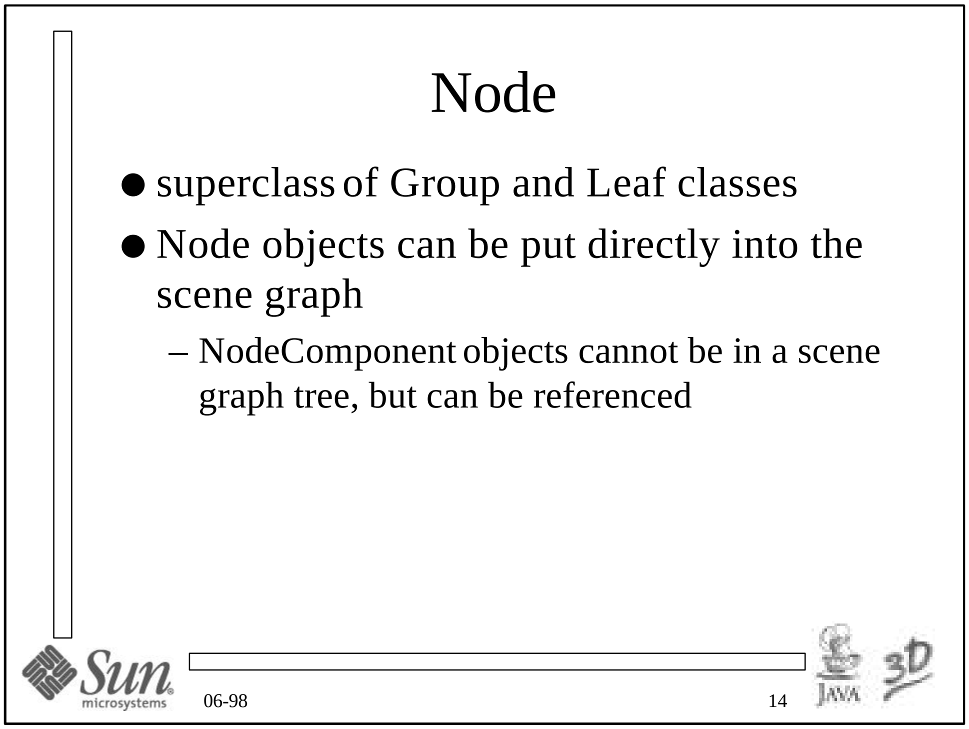
SceneGraphObject methods

```
final boolean getCapability(int bit)
final void setCapability(int bit)
final void clearCapability(int bit)
final boolean isCompiled()
final boolean isLive()
void setUserData(Object userData)
Object getUserData(Object userData)
```

By default, all capabilities are turned off.

From now on, to reduce space, get*() methods which correspond to set*() methods will not be listed here.

Note: documented methods of Java 3D classes are public



Node

- superclass of Group and Leaf classes
- Node objects can be put directly into the scene graph
  - NodeComponent objects cannot be in a scene graph tree, but can be referenced

Student Notes

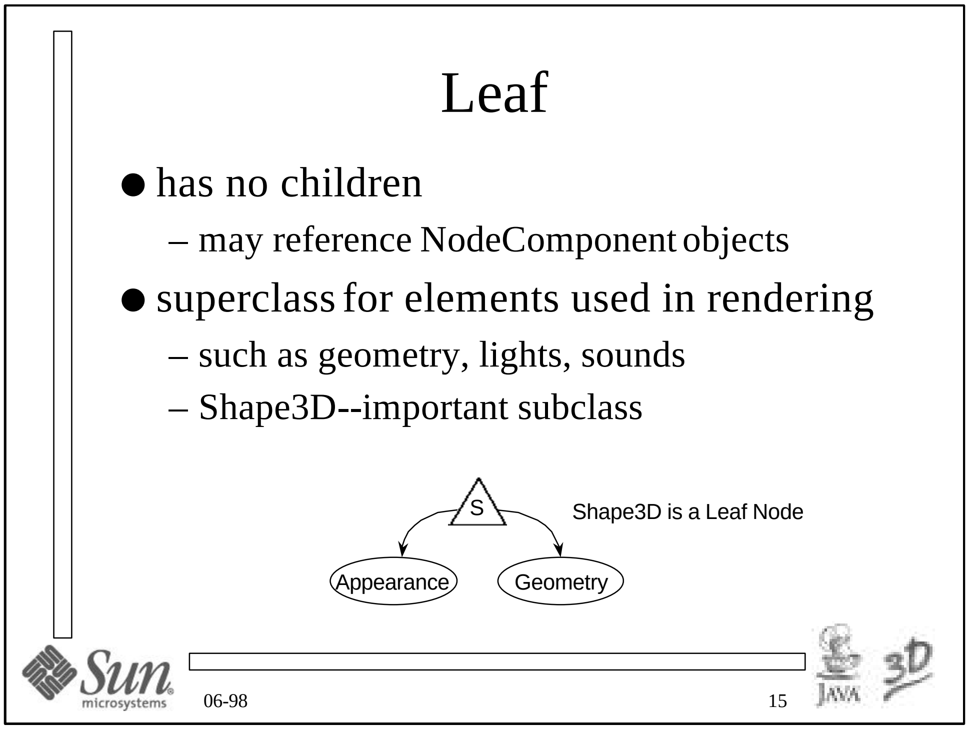
Node methods (partial list)

```
final void setBounds(Bounds region)
final void setBoundsAutoCompute(boolean autoCompute)
final void getLocalToVworld(...)
Node cloneTree(...)
Node cloneNode(boolean forceDuplicate)
void duplicateNode(Node originalNode, boolean forceDuplicate)
void setPickable(boolean pickable)
```

Node capabilities (partial list)

```
ALLOW_BOUNDS_READ, ALLOW_BOUNDS_WRITE
ALLOW_AUTO_COMPUTE_BOUNDS_READ
ALLOW_AUTO_COMPUTE_BOUNDS_WRITE
ENABLE_PICK_REPORTING
ALLOW_PICKABLE_READ, ALLOW_PICKABLE_WRITE
ENABLE_COLLISION_REPORTING
ALLOW_COLLIDABLE_READ, ALLOW_COLLIDABLE_WRITE
ALLOW_LOCAL_TO_VWORLD_READ
```

Read capability usually has a corresponding Write capability. To reduce space, they will be represented together with the shorthand READ | WRITE.



Leaf

- has no children
  - – may reference NodeComponent objects
- superclass for elements used in rendering
  - – such as geometry, lights, sounds
  - – Shape3D--important subclass

S
Appearance
Geometry

Shape3D is a Leaf Node

06-98 15

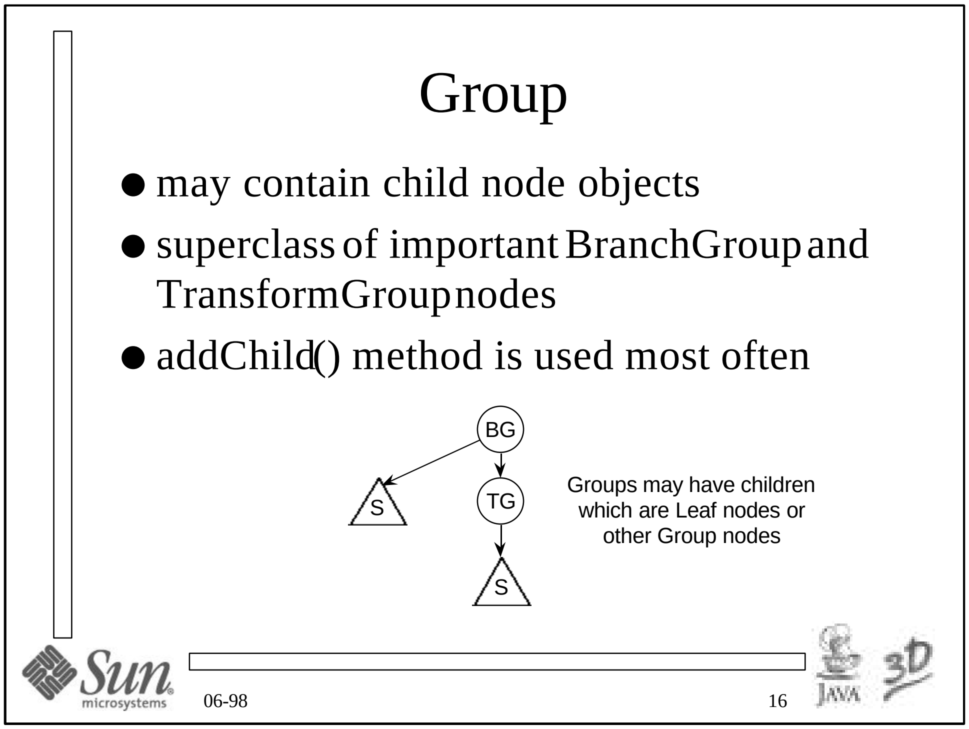
Student Notes

Java 3D Object Hierarchy

- SceneGraphObject
  - Node
    - Leaf
      - Background
      - Behavior
      - BoundingLeaf
      - Clip
      - Fog
      - Light
      - Link
      - Morph
      - Shape3D
      - Sound
      - Soundscape
      - ViewPlatform

Leaf method

```
void updateNodeReferences(NodeReferenceTable referenceTable)
```



Group

- may contain child node objects
- superclass of important BranchGroup and TransformGroup nodes
- addChild() method is used most often

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

06-98

16

Student Notes

Group subclass hierarchy

- SceneGraphObject
  - Node
    - Group
      - BranchGroup
      - OrderedGroup
      - SharedGroup
      - Switch
      - TransformGroup

Group methods and capabilities (partial list)

When live or compiled, ALLOW_CHILDREN_READ enables the methods

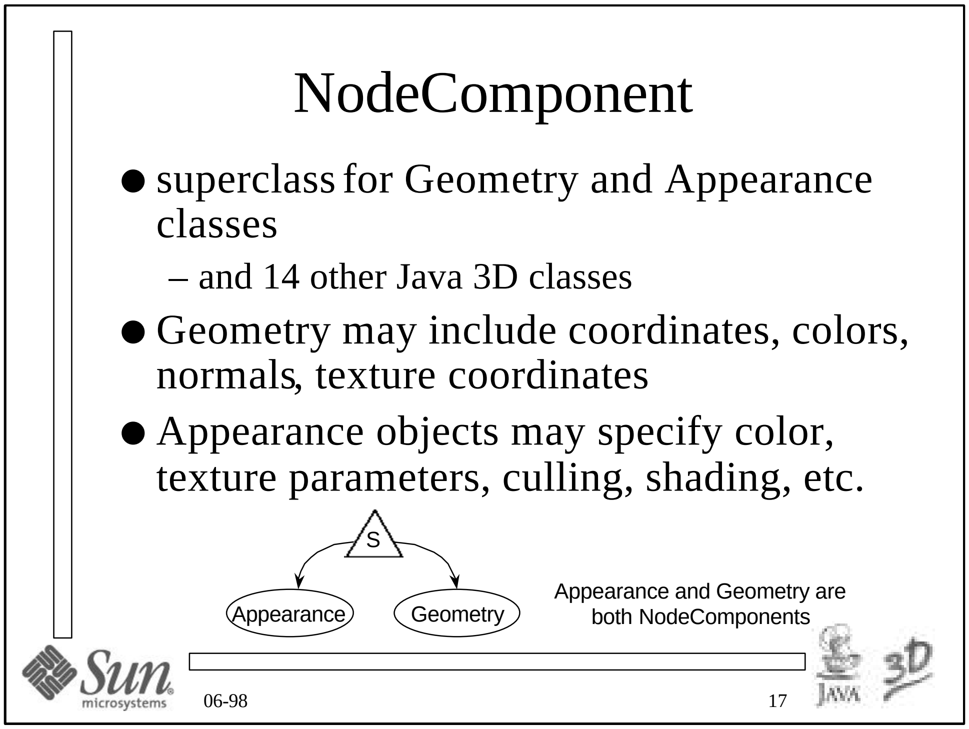
```
final Node getChild (int index)
final int numChildren ()
```

Similarly, ALLOW_CHILDREN_WRITE enables

```
final void setChild (Node child, int index)
final void insertChild (Node child, int index)
final void removeChild (int index)
```

And ALLOW_CHILDREN_EXTEND enables

```
final void addChild (Node child)
final void moveTo (BranchGroup branchGroup)
```



NodeComponent

- superclass for Geometry and Appearance classes
  - – and 14 other Java 3D classes
- Geometry may include coordinates, colors, normals, texture coordinates
- Appearance objects may specify color, texture parameters, culling, shading, etc.

S

Appearance

Geometry

Appearance and Geometry are both NodeComponents

06-98

17

Student Notes

A Shape3D leaf node references Appearance and Geometry objects, which are both NodeComponents.

NodeComponent subclass hierarchy (partial list)

SceneGraphObject

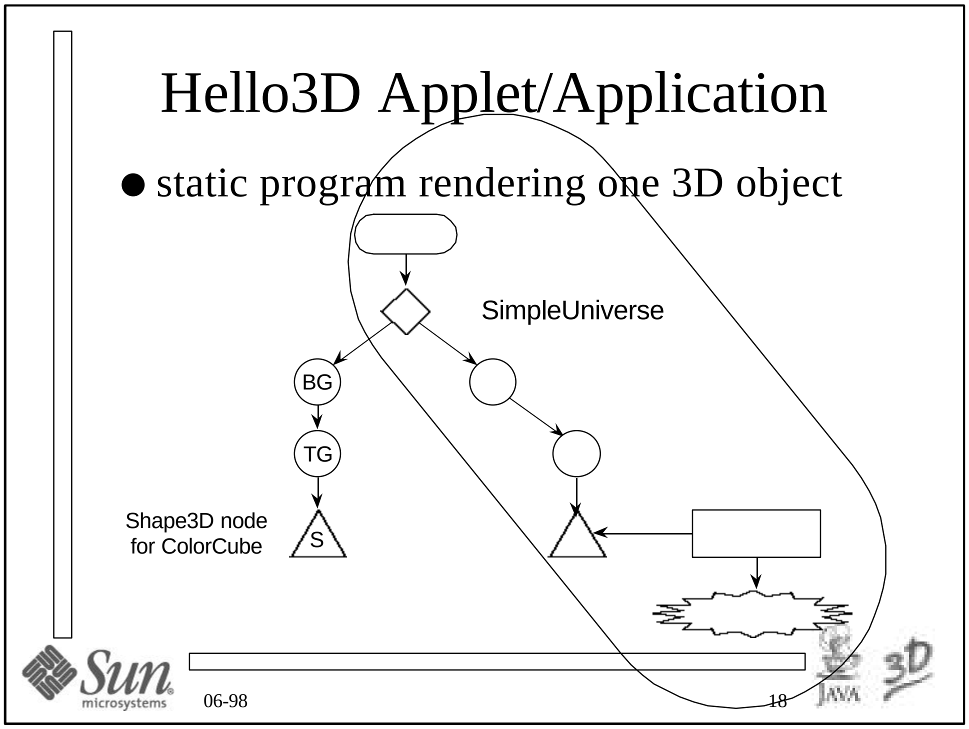
- NodeComponent
  - Geometry
  - Appearance
  - ColoringAttributes
  - LineAttributes
  - PointAttributes
  - PolygonAttributes
  - RenderingAttributes
  - TextureAttributes
  - TransparencyAttributes
  - Material
  - Texture

NodeComponent methods (partial list)

```
final void setDuplicateOnCloneTree(boolean duplicate)
```

```
NodeComponent cloneNodeComponent()
```

```
void duplicateNodeComponent(NodeComponent
originalNodeComponent)
```



Hello3D Applet/Application

- static program rendering one 3D object

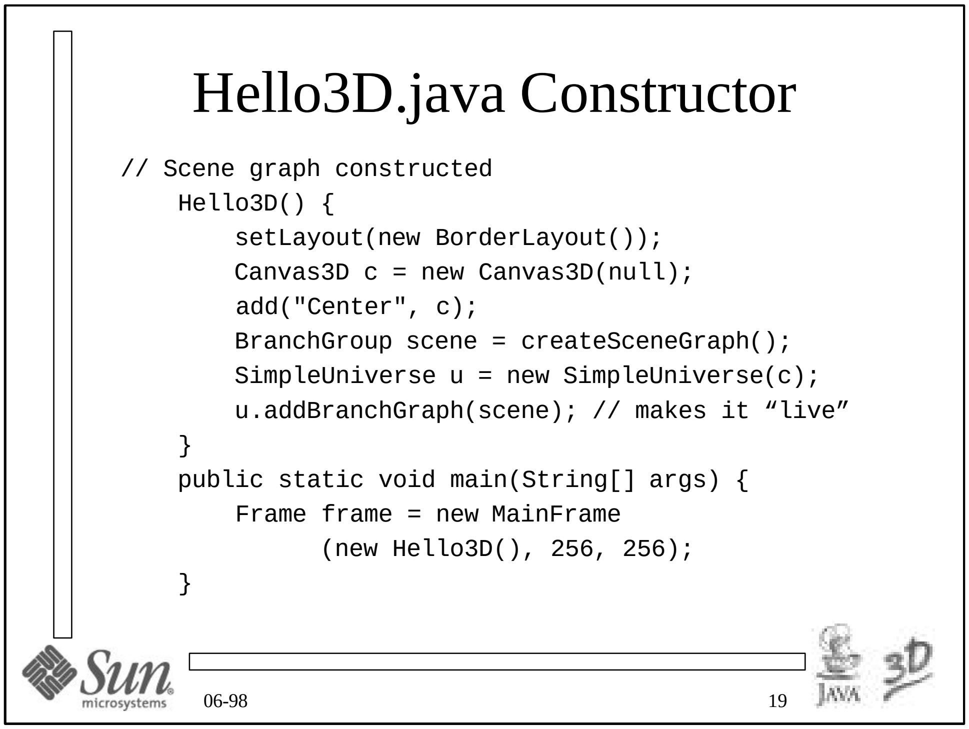
06-98 18

Student Notes

SimpleUniverse is a convenience utility in the package com.sun.j3d.utils.universe.* It creates a branch graph with a VirtualUniverse, Locale, BranchGroup, MultiTransformGroup, and ViewPlatform objects. It also creates other objects which are referenced by the ViewPlatform, such as a PhysicalBody and PhysicalEnvironment.

The MultiTransformGroup is not a standard Java 3D class. It is a convenience utility class that supports several TransformGroup objects.

The entire Java 3D View Model, including the convenience classes SimpleUniverse and MultiTransformGroup are discussed much later.



Hello3D.java Constructor

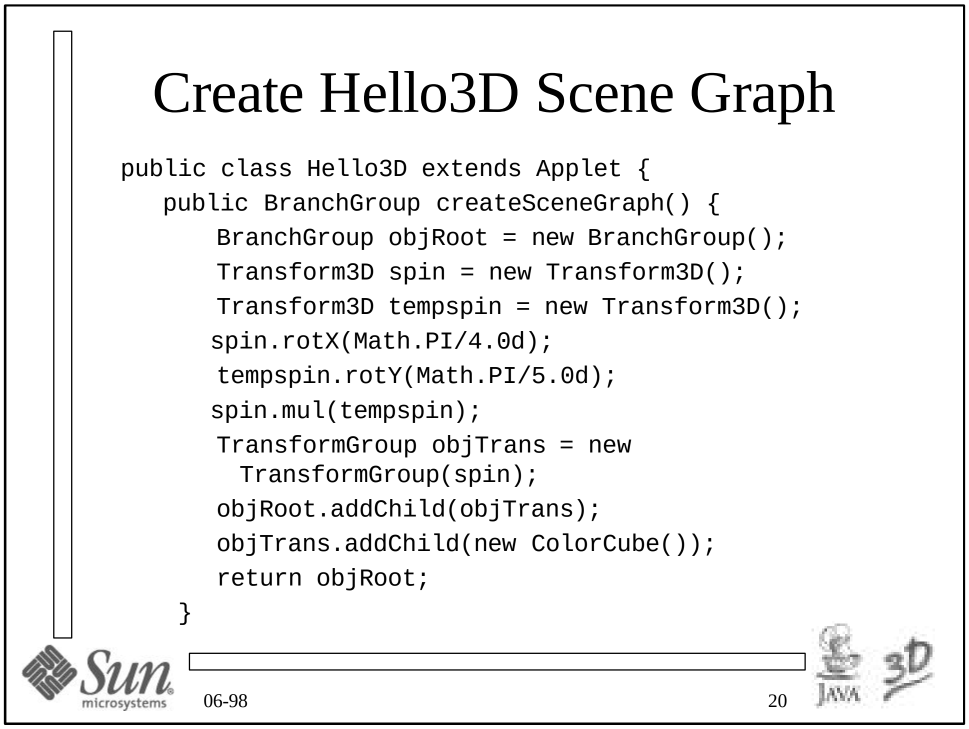
```
// Scene graph constructed
    Hello3D() {
        setLayout(new BorderLayout());
        Canvas3D c = new Canvas3D(null);
        add("Center", c);
        BranchGroup scene = createSceneGraph();
        SimpleUniverse u = new SimpleUniverse(c);
        u.addBranchGraph(scene); // makes it “live”
    }
    public static void main(String[] args) {
        Frame frame = new MainFrame
              (new Hello3D(), 256, 256);
    }
```

Student Notes

Note the Java 3D Canvas3D object is placed within a standard Java AWT container (with the specified LayoutManager).

The MainFrame object allows the class to be run as either a standalone application or as an applet in a web browser. The MainFrame class is Copyright (C) 1996-1998 by Jef Poskanzer <jef@acme.com>. All rights reserved.

Visit the ACME Labs Java page for up-to-date versions of this and other fine Java utilities: http://www.acme.com/java/



Create Hello3D Scene Graph

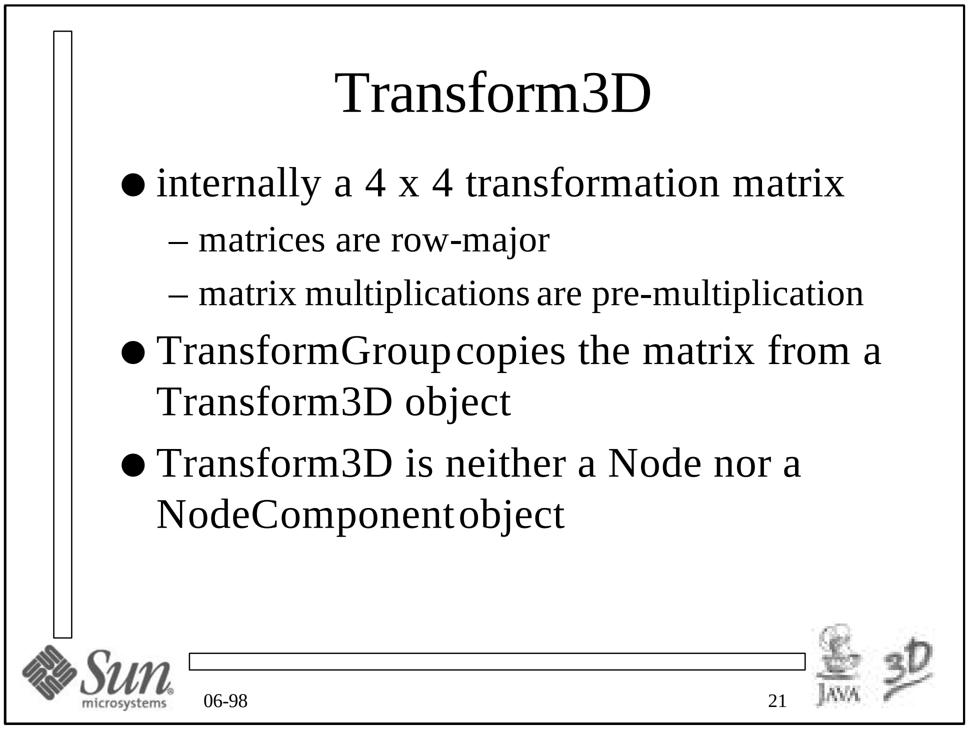
```
public class Hello3D extends Applet {
   public BranchGroup createSceneGraph() {
       BranchGroup objRoot = new BranchGroup();
       Transform3D spin = new Transform3D();
       Transform3D tempspin = new Transform3D();
      spin.rotX(Math.PI/4.0d);
       tempspin.rotY(Math.PI/5.0d);
      spin.mul(tempspin);
       TransformGroup objTrans = new
         TransformGroup(spin);
       objRoot.addChild(objTrans);
       objTrans.addChild(new ColorCube());
       return objRoot;
    }
```

Student Notes

createSceneGraph() creates several objects, such as a BranchGroup, TransformGroup, and Leaf node.

The TransformGroup references a Transform3D class, which represents the transformation matrix. Note the operations to generate the appropriate matrix. In this example, the cube is rotated slightly, so that it looks more 3D.

The ColorCube class is in the Convenience Utility library: com.sun.j3d.utils.geometry.ColorCube. The getShape() method retrieves the Shape3D node of the ColorCube object.



Transform3D

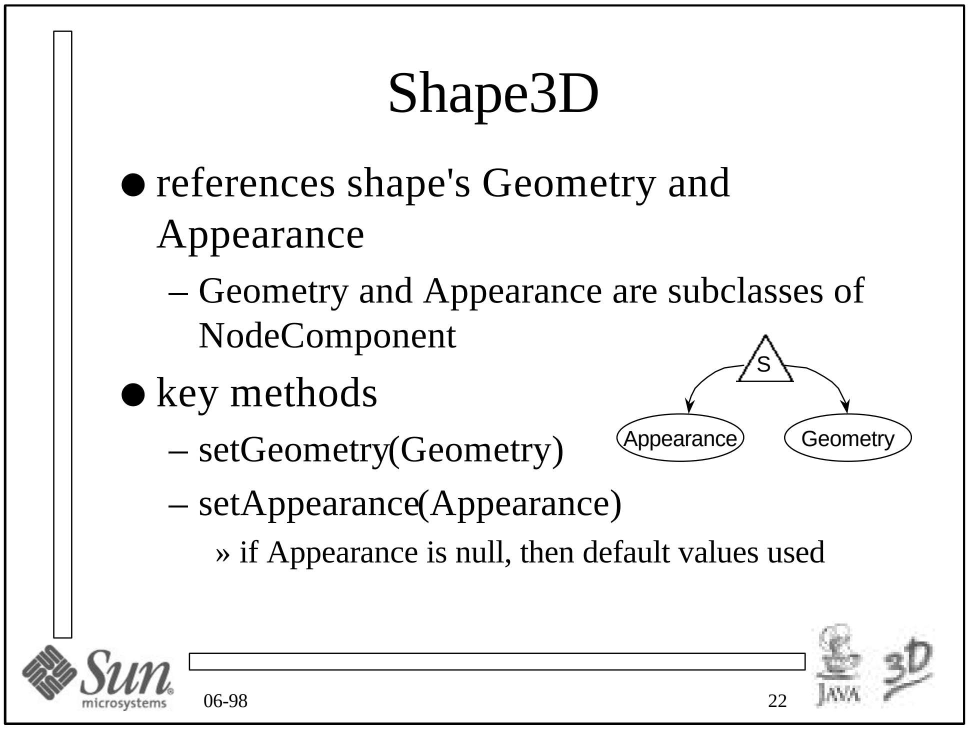
- internally a 4 x 4 transformation matrix
  - – matrices are row-major
  - – matrix multiplications are pre-multiplication
- TransformGroup copies the matrix from a Transform3D object
- Transform3D is neither a Node nor a NodeComponent object

Student Notes

Transform3D methods (partial list)

```
final void set(...)
final void setIdentity()
final void setTranslation(Vector3f or Vector3d)
final void setRotation(...)
final void setScale(...)
void rotX(double angle)
void rotY(double angle)
void rotZ(double angle)
final void mul(...)
final void transpose(...)
final void invert()
final double determinant()
final void transform(Vector4d vec, vector4d vecOut)
```

Avoid the use of the View Model Compatibility Mode methods (described in Appendix C.11), which make it hard to use stereo or head-tracking input.



Shape3D

- references shape's Geometry and Appearance
  - Geometry and Appearance are subclasses of NodeComponent
- key methods
  - setGeometry(Geometry)
  - setAppearance(Appearance)
    - » if Appearance is null, then default values used

S
Appearance
Geometry

06-98 22

Student Notes

Shape3D capabilities:

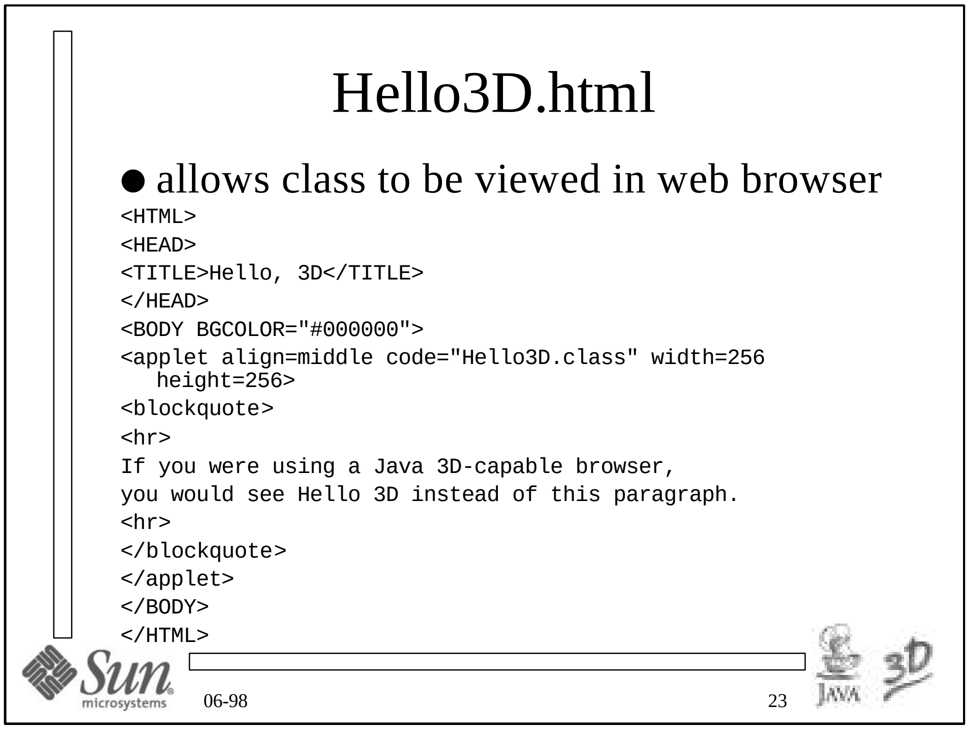
ALLOW_GEOMETRY_READ | WRITE

ALLOW_APPEARANCE_READ | WRITE

ALLOW_COLLISION_BOUNDS_READ | WRITE

Shape3D methods (partial list)

```
final void setGeometry(Geometry geometry)
final void setAppearance(Appearance appearance)
final void setCollisionBounds(Bounds bounds)
Node cloneNode(boolean forceDuplicate)
void duplicateNode(Node originalNode, boolean  forceDuplicate)
void updateNodeReferences(NodeReferenceTable referenceTable)
```

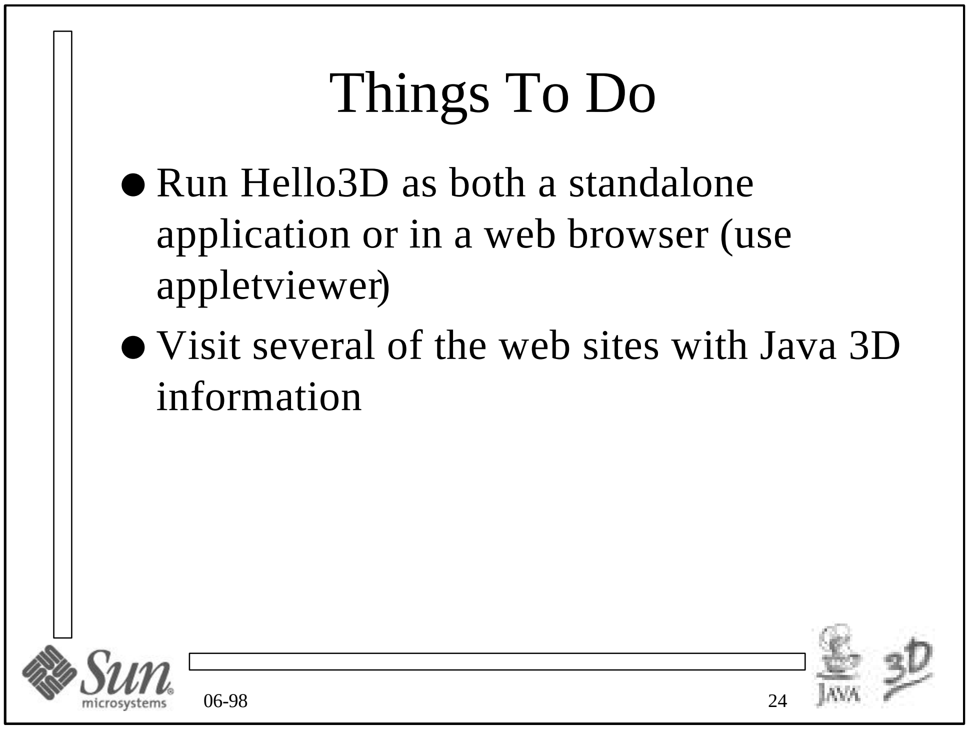


Hello3D.html

- allows class to be viewed in web browser

```
<HTML>
<HEAD>
<TITLE>Hello, 3D</TITLE>
</HEAD>
<BODY BGCOLOR="#000000">
<applet align=middle code="Hello3D.class" width=256
   height=256>
<blockquote>
<hr>
If you were using a Java 3D-capable browser,
you would see Hello 3D instead of this paragraph.
<hr>
</blockquote>
</applet>
</BODY>
</HTML>
```

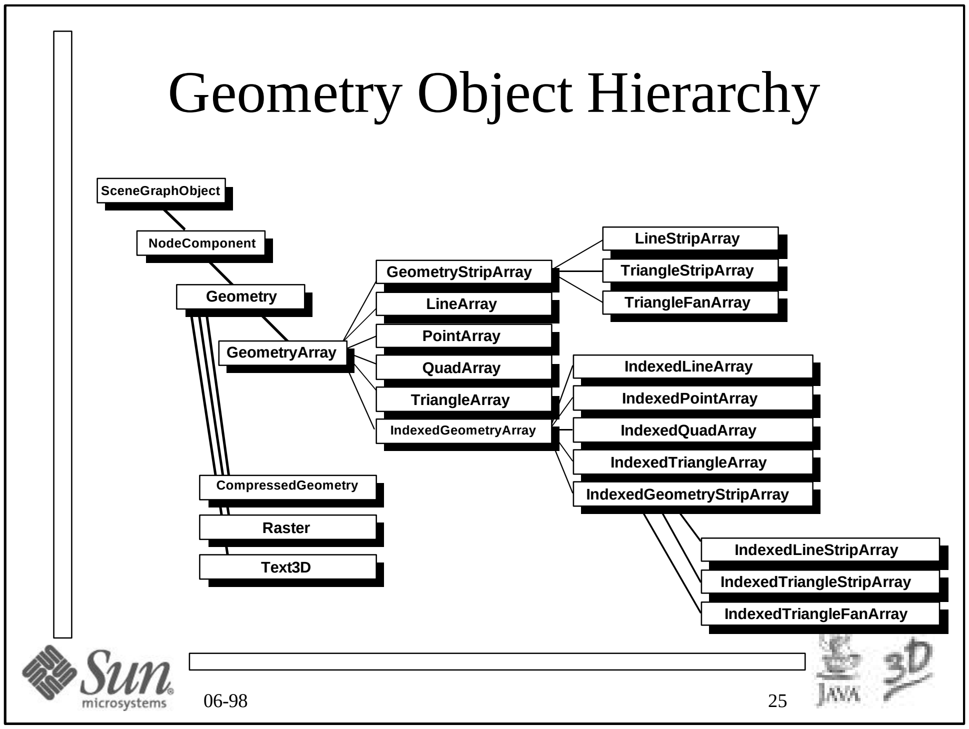
Student Notes



Things To Do

- Run Hello3D as both a standalone application or in a web browser (use appletviewer)
- Visit several of the web sites with Java 3D information

Student Notes



Geometry Object Hierarchy

SceneGraphObject

NodeComponent

Geometry

GeometryArray

GeometryStripArray

LineArray

PointArray

QuadArray

TriangleArray

IndexedGeometryArray

LineStripArray

TriangleStripArray

TriangleFanArray

IndexedLineArray

IndexedPointArray

IndexedQuadArray

IndexedTriangleArray

IndexedGeometryStripArray

IndexedLineStripArray

IndexedTriangleStripArray

IndexedTriangleFanArray

CompressedGeometry

Raster

Text3D

06-98

25

Student Notes

GeometryArray (and subclasses) store coordinate and related information for each vertex in one or more arrays. A Shape3D object references one Geometry object for its data.

GeometryArray capabilities:

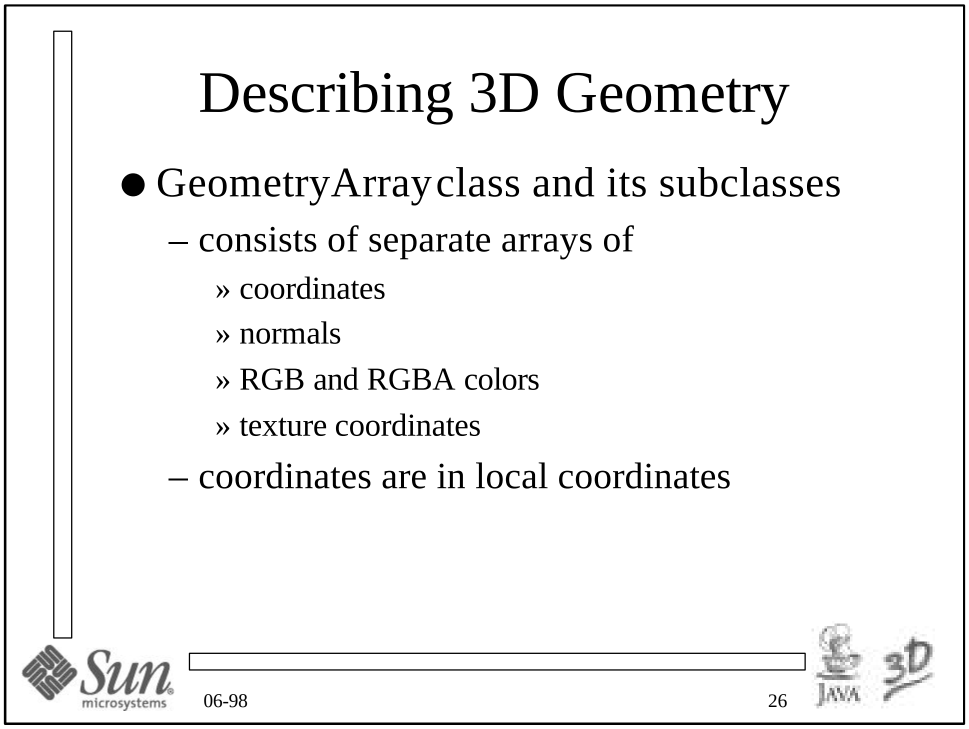
ALLOW_COORDINATE_READ | WRITE

ALLOW_COLOR_READ | WRITE

ALLOW_NORMAL_READ | WRITE

ALLOW_TEXCOORD_READ | WRITE

ALLOW_COUNT_READ



Describing 3D Geometry

- GeometryArray class and its subclasses
  - consists of separate arrays of
    - coordinates
    - normals
    - RGB and RGBA colors
    - texture coordinates
  - coordinates are in local coordinates

Student Notes

GeometryArray constructor:

```
GeometryArray (int vertexCount, int vertexFormat)
```

`vertexFormat` is a mask indicating what is present in each vertex:

COORDINATES

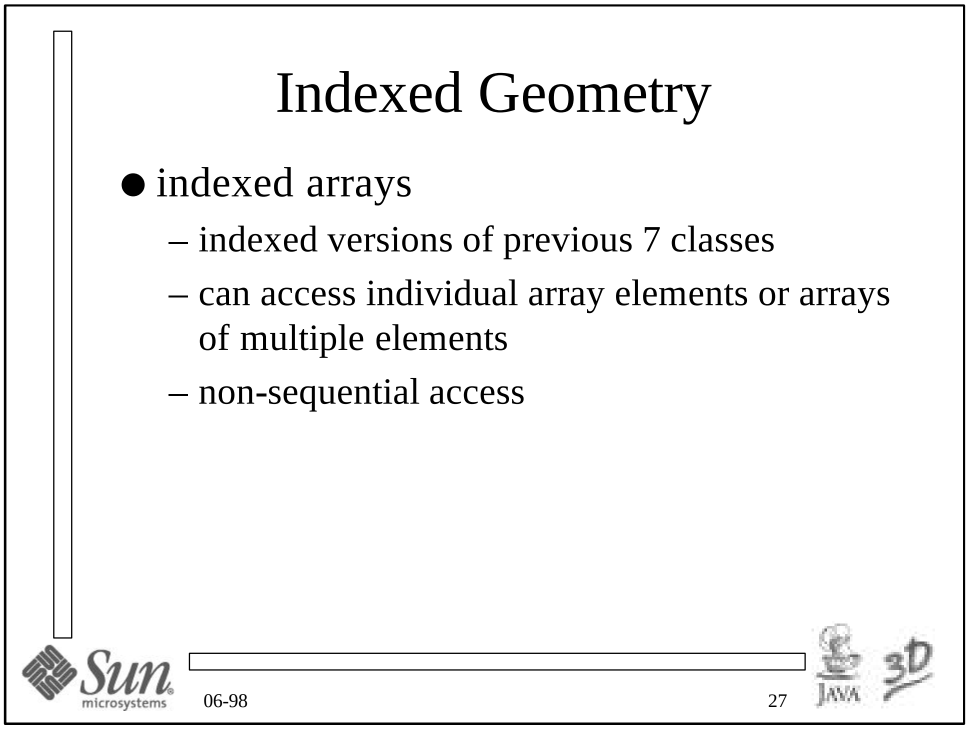
NORMALS

COLOR_3 or COLOR_4

TEXTURE_COORDINATE_2 or TEXTURE_COORDINATE_3

GeometryArray methods (partial list):

```
final int getVertexCount()
final int getVertexFormat()
final void setCoordinate(...)
final void setCoordinates(...)
final void setColor(...)
final void setColors(...)
final void setNormal(...)
final void setNormals(...)
final void setTextureCoordinates(...)
```



Indexed Geometry

- indexed arrays
  - indexed versions of previous 7 classes
  - can access individual array elements or arrays of multiple elements
  - non-sequential access

Student Notes

IndexedGeometryArray capabilities:

ALLOW_COORDINATE_INDEX_READ | WRITE

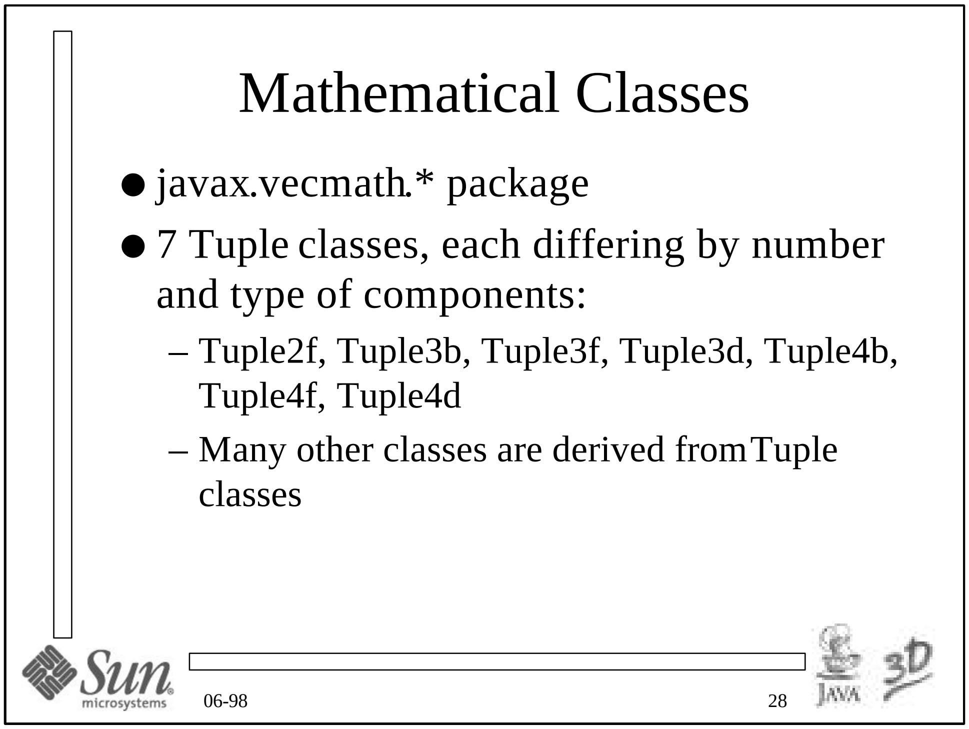
ALLOW_COLOR_INDEX_READ | WRITE

ALLOW_NORMAL_INDEX_READ | WRITE

ALLOW_TEXCOORD_INDEX_READ | WRITE

IndexedGeometryArray methods (partial list):

```
final void setCoordinateIndex(int index, int  coordinateIndex)
final void setCoordinateIndices(int index, int  coordinateIndices[])
final void setColorIndex(int index, int colorIndex)
final void setColorIndices(int index, int colorIndices[])
final void setNormalIndex(int index, int normalIndex)
final void setNormalIndices(int index, int  normalIndices[])
final void setTextureCoordinateIndex(int index, int  texCoordIndex)
final void setTextureCoordinateIndices(int index, int
texCoordIndices[])
final int getIndexCount()
```



Mathematical Classes

- javax.vecmath.* package
- 7 Tuple classes, each differing by number and type of components:
  - – Tuple2f, Tuple3b, Tuple3f, Tuple3d, Tuple4b, Tuple4f, Tuple4d
  - – Many other classes are derived from Tuple classes

Student Notes

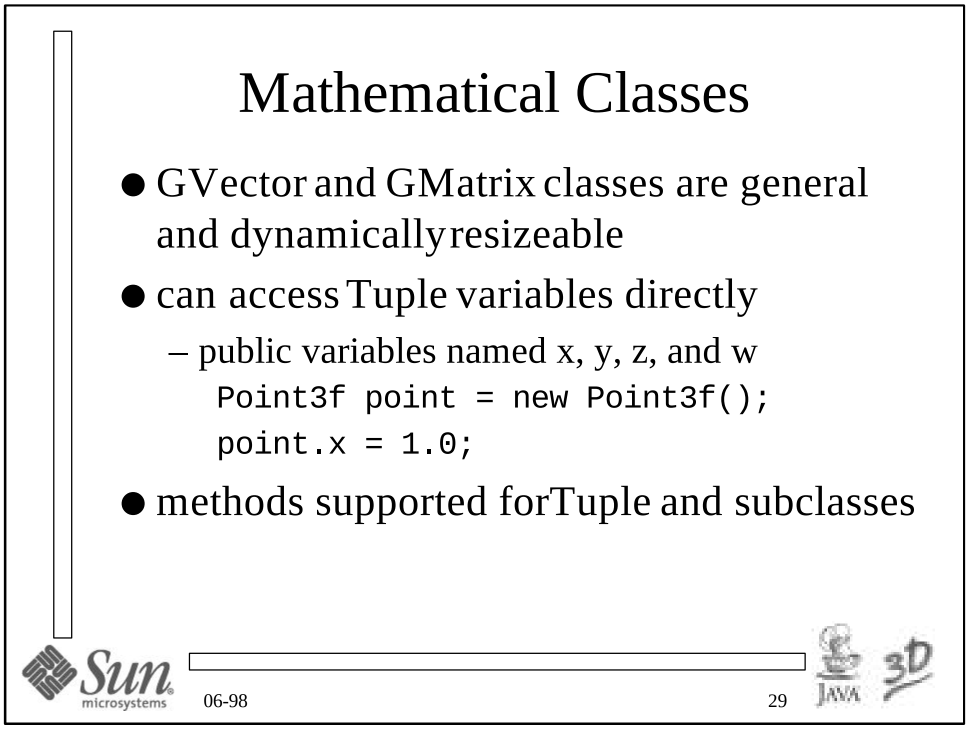
Tuple Object Hierarchies:

- Tuple2f
  - Point2f
  - TexCoord2f
  - Vector2f

- Tuple3f
  - Point3f
  - TexCoord3f
  - Vector3f
  - Color3f
- Tuple3d
  - Point3d
  - Vector3d
- Tuple3b
  - Color3b

- Tuple4f
  - Point4f
  - Quat4f
  - Vector4f
  - Color4f
- Tuple4d
  - Point4d
  - Vector4d
  - Quat4d
- Tuple4b
  - Color4b

Other Math Objects include AxisAngle4d, AxisAngle4f, GVector, Matrix3f, Matrix3d, Matrix4f, Matrix4d, and GMatrix



Mathematical Classes

- GVector and GMatrix classes are general and dynamically resizeable
- can access Tuple variables directly
  - – public variables named x, y, z, and w

```
Point3f point = new Point3f();
point.x = 1.0;
```

- methods supported forTuple and subclasses

06-98 29

Student Notes

java.vecmath methods (partial list for only a couple of classes):

Tuple*

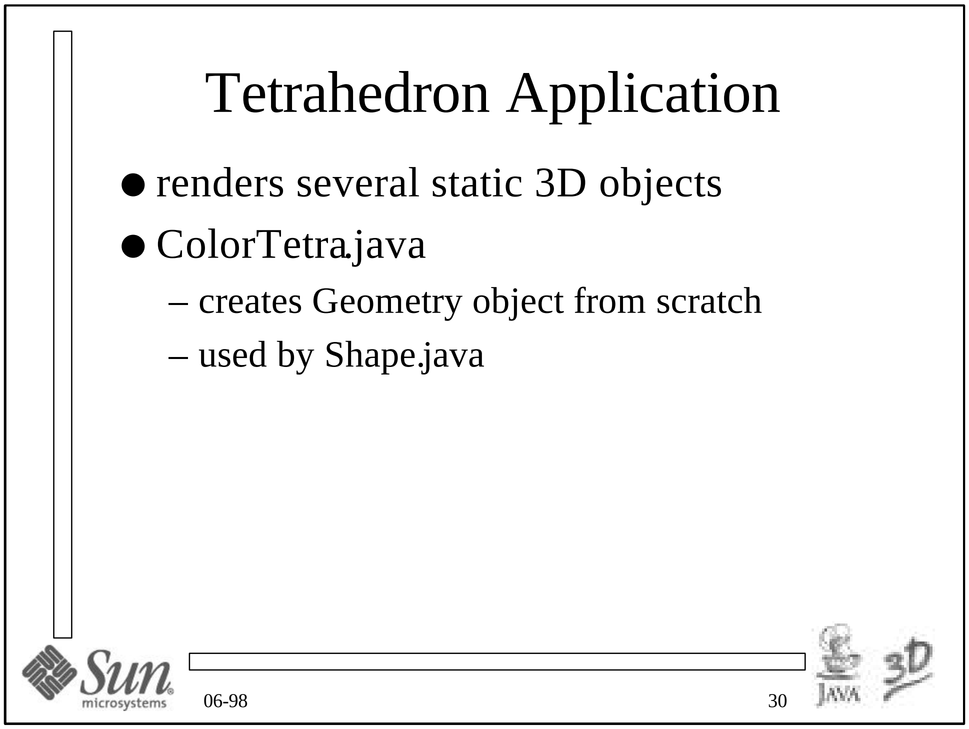
```
final void set(...)
final void add(...)
final void sub(...)
final void negate(...)
final void absolute(...)
final boolean equals(...)
```

Point* (inherits all Tuple* methods, too)

```
final float distance(Point*)
```

Vector* (inherits all Tuple* methods, too)

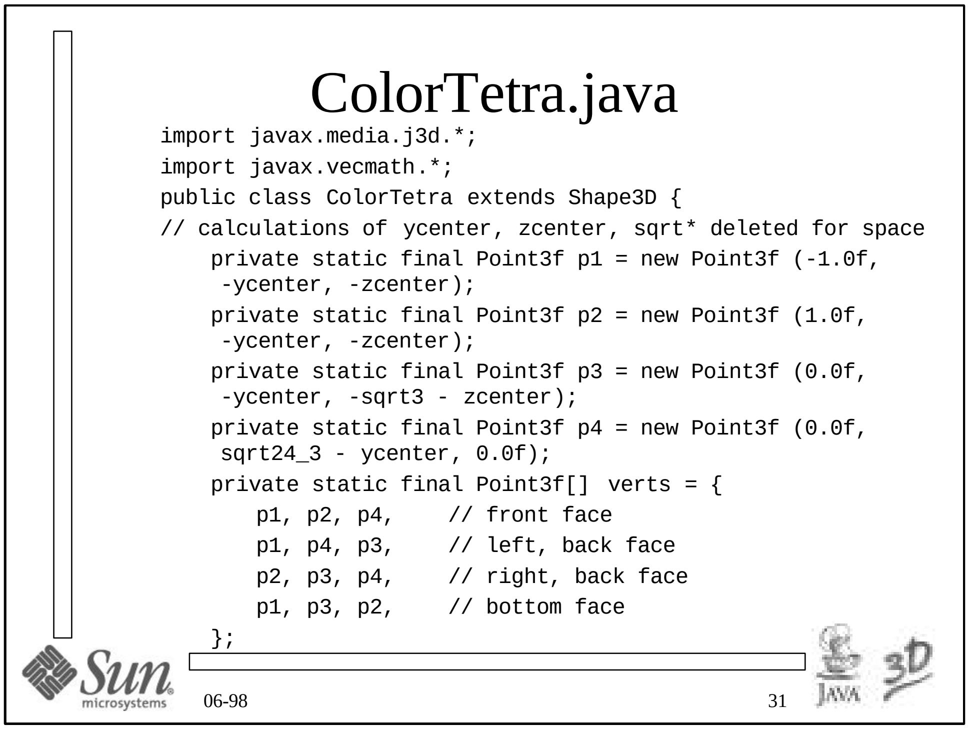
```
final float dot(Vector*)
final float length()
final void normalize(...)
final float angle(Vector*)
```



Tetrahedron Application

- renders several static 3D objects
- ColorTetra.java
  - – creates Geometry object from scratch
  - – used by Shape.java

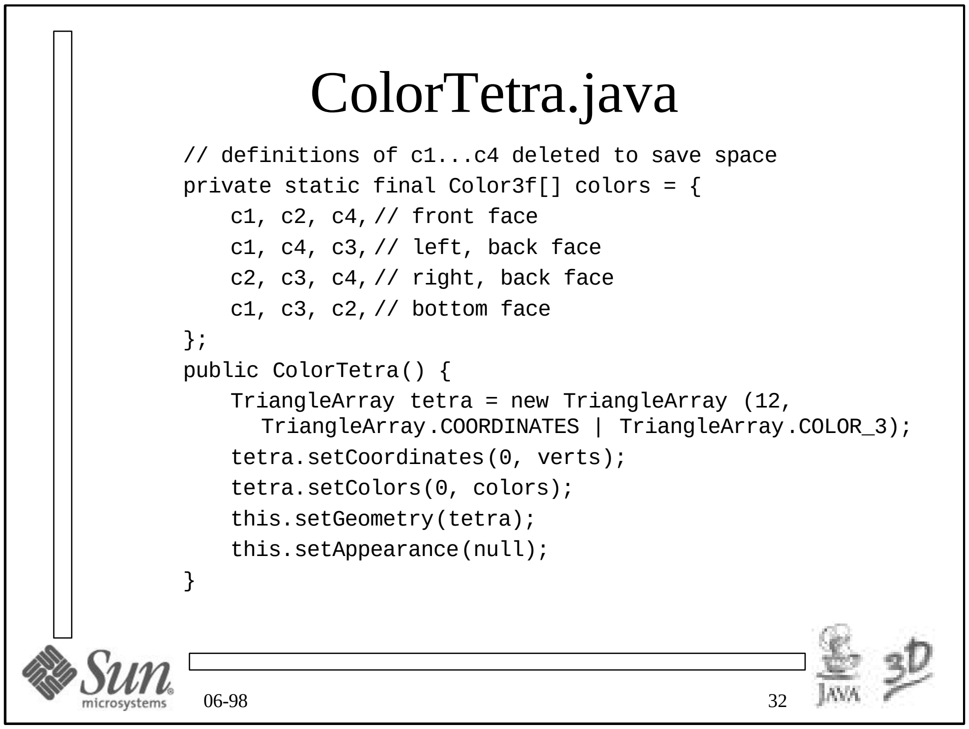
Student Notes



ColorTetra.java

```
import javax.media.j3d.*;
import javax.vecmath.*;
public class ColorTetra extends Shape3D {
// calculations of ycenter, zcenter, sqrt* deleted for space
    private static final Point3f p1 = new Point3f (-1.0f,
     -ycenter, -zcenter);
    private static final Point3f p2 = new Point3f (1.0f,
     -ycenter, -zcenter);
    private static final Point3f p3 = new Point3f (0.0f,
     -ycenter, -sqrt3 - zcenter);
    private static final Point3f p4 = new Point3f (0.0f,
     sqrt24_3 - ycenter, 0.0f);
    private static final Point3f[] verts = {
       p1, p2, p4,    // front face
       p1, p4, p3,    // left, back face
       p2, p3, p4,    // right, back face
       p1, p3, p2,    // bottom face
    };
```

Student Notes

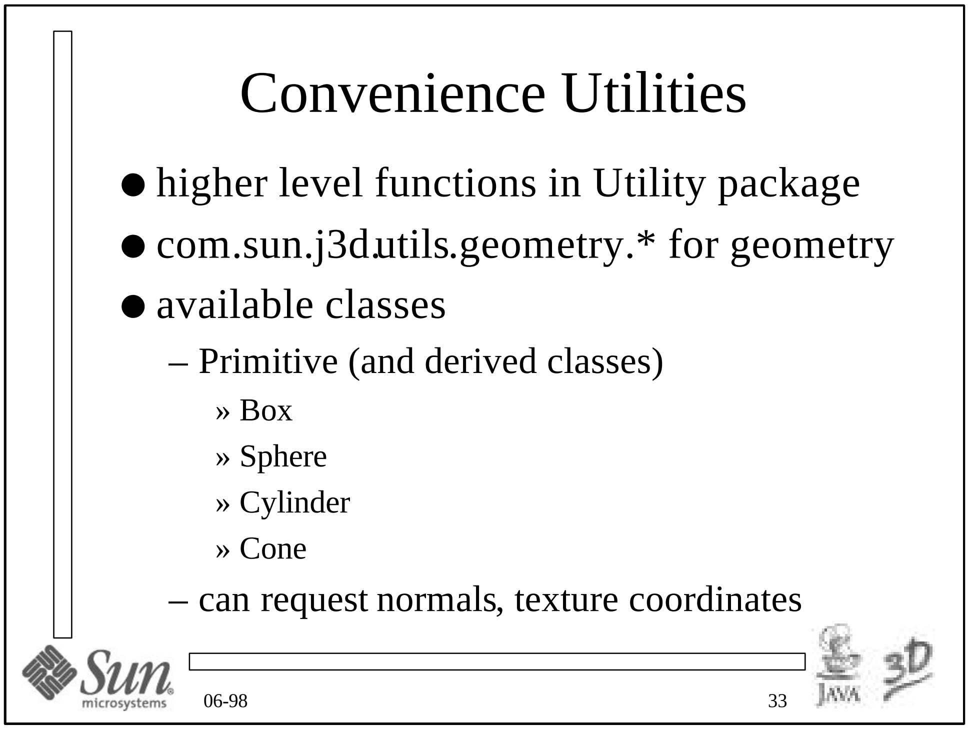


ColorTetra.java

```
// definitions of c1...c4 deleted to save space
private static final Color3f[] colors = {
    c1, c2, c4, // front face
    c1, c4, c3, // left, back face
    c2, c3, c4, // right, back face
    c1, c3, c2, // bottom face
};
public ColorTetra() {
    TriangleArray tetra = new TriangleArray (12,
      TriangleArray.COORDINATES | TriangleArray.COLOR_3);
    tetra.setCoordinates(0, verts);
    tetra.setColors(0, colors);
    this.setGeometry(tetra);
    this.setAppearance(null);
}
```

Student Notes

Since the colors are specified in the Geometry object (TriangleArray), these values override any colors which are set in the Appearance node component object.



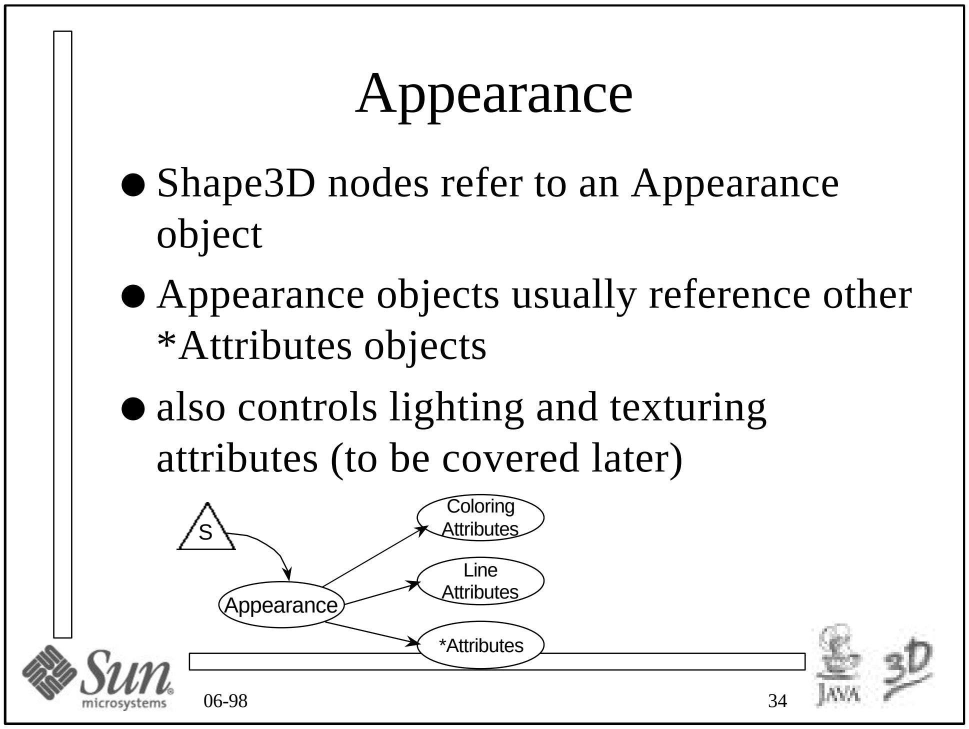
Convenience Utilities

- higher level functions in Utility package
- com.sun.j3d.utils.geometry.* for geometry
- available classes
  - – Primitive (and derived classes)
    - » Box
    - » Sphere
    - » Cylinder
    - » Cone
  - – can request normals, texture coordinates

Student Notes

Java 3D is intended to cover the basics of creating and maintaining a scene graph. The core Java 3D package is meant to remain small. The Utility packages are the place for additional functionality.

For example, the Shape3D class is the only class in the standard Java 3D package to represent a geometric object. The Box, Sphere, Cylinder, and Cone classes are additional, specific, geometric objects and are in the Utility package. Future functionality, such as NURBS, likely would be added as Utility classes, not in the core Java 3D package.



Appearance

- Shape3D nodes refer to an Appearance object
- Appearance objects usually reference other *Attributes objects
- also controls lighting and texturing attributes (to be covered later)

06-98 34

Student Notes

Attribute capabilities (excluding lighting and texturing):

ALLOW_COLORING_ATTRIBUTES_READ | WRITE

ALLOW_TRANSPARENCY_ATTRIBUTES_READ | WRITE

ALLOW_RENDERING_ATTRIBUTES_READ | WRITE

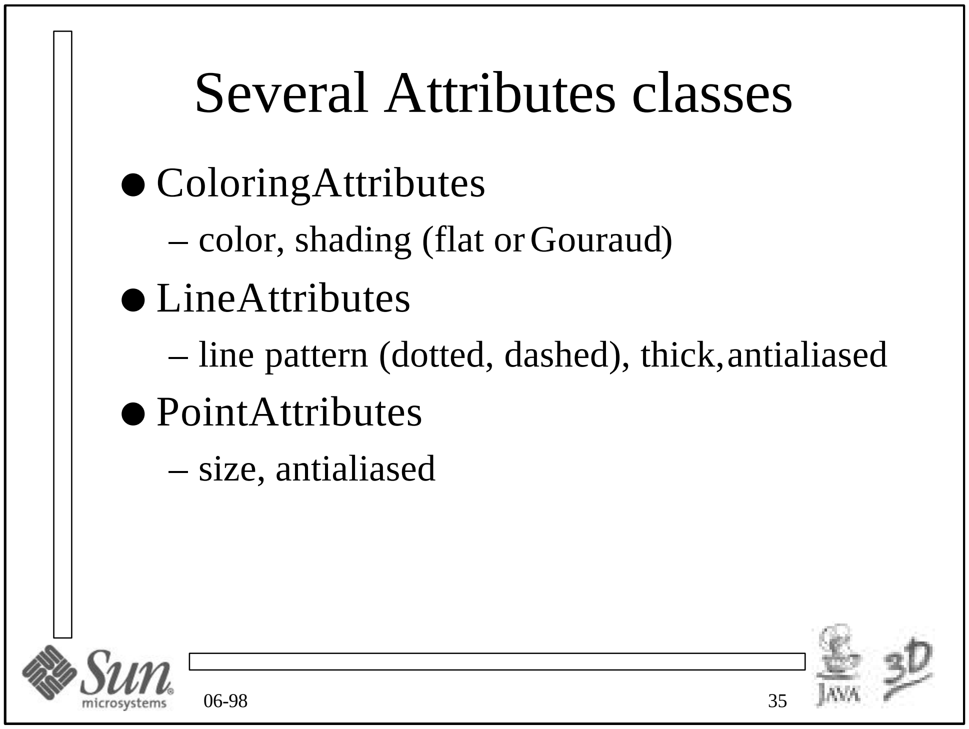
ALLOW_POLYGON_ATTRIBUTES_READ | WRITE

ALLOW_LINE_ATTRIBUTES_READ | WRITE

ALLOW_POINT_ATTRIBUTES_READ | WRITE

Attribute methods (partial list, excluding lighting and texturing):

```
final void setColoringAttributes(ColoringAttributes
coloringAttributes)
final void setTransparencyAttributes(TransparencyAttributes
transparencyAttributes)
final void setRenderingAttributes(RenderingAttributes
renderingAttributes)
final void setPolygonAttributes(PolygonAttributes polygonAttributes)
final void setLineAttributes(LineAttributes lineAttributes)
final void setPointAttributes(PointAttributes pointAttributes)
```



Several Attributes classes

- ColoringAttributes
  - – color, shading (flat or Gouraud)
- LineAttributes
  - – line pattern (dotted, dashed), thick, antialiased
- PointAttributes
  - – size, antialiased

06-98 35

Student Notes

ColoringAttributes methods:

```
final void setColor(...)
final void setShadeModel(int shadeModel)
```

where `shadeModel` is one of the following constants:

`FASTEST`, `NICEST`, `SHADE_FLAT`, or `SHADE_GOURAUD`

LineAttributes methods:

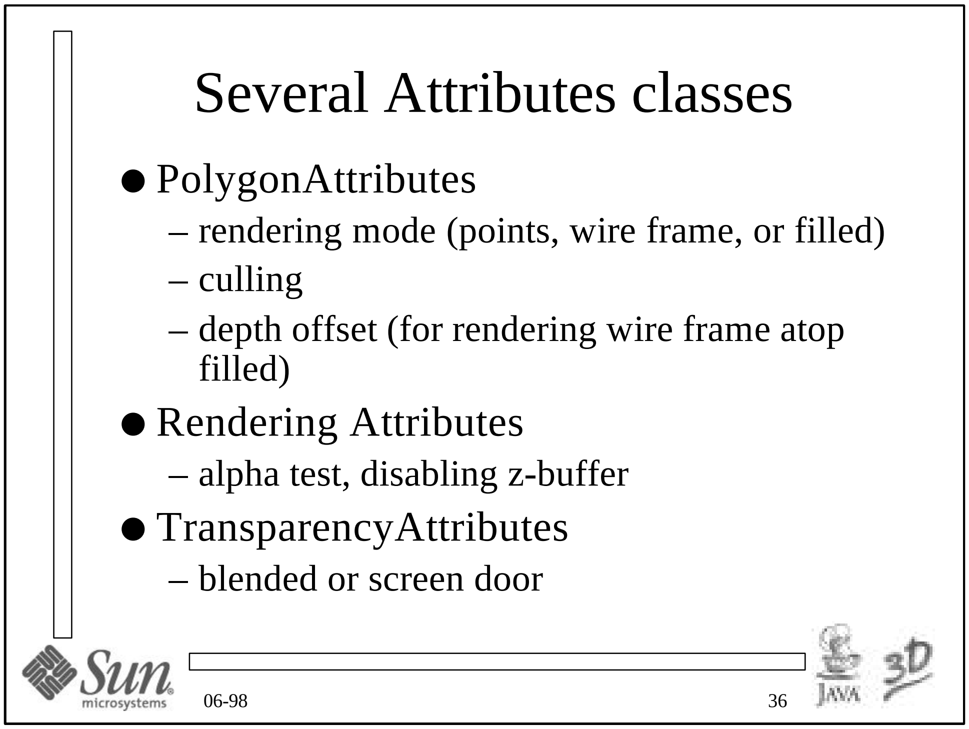
```
final void setLineWidth(float lineWidth)
final void setLinePattern(int linePattern)
```

where `linePattern` is one of the following constants: `PATTERN_SOLID`, `PATTERN_DASH`, `PATTERN_DOT`, or `PATTERN_DASH_DOT`.

```
final void setLineAntialiasingEnable(boolean state)
```

PointAttributes methods:

```
final void setPointSize(float pointSize)
final void setPointAntialiasingEnable(boolean state)
```



Several Attributes classes

- PolygonAttributes
  - – rendering mode (points, wire frame, or filled)
  - – culling
  - – depth offset (for rendering wire frame atop filled)
- Rendering Attributes
  - – alpha test, disabling z-buffer
- TransparencyAttributes
  - – blended or screen door

Student Notes

PolygonAttributes methods and constants:

```
final void setCullFace(int cullFace)
```

where `cullFace` is one of the following: `CULL_NONE`, `CULL_FRONT`, or `CULL_BACK`

```
final void setPolygonMode(int polygonMode)
```

where `polygonMode` is one of the following: `POLYGON_POINT`, or `POLYGON_LINE`, `POLYGON_FILL`

```
final void setPolygonOffset(float polygonOffset)
```

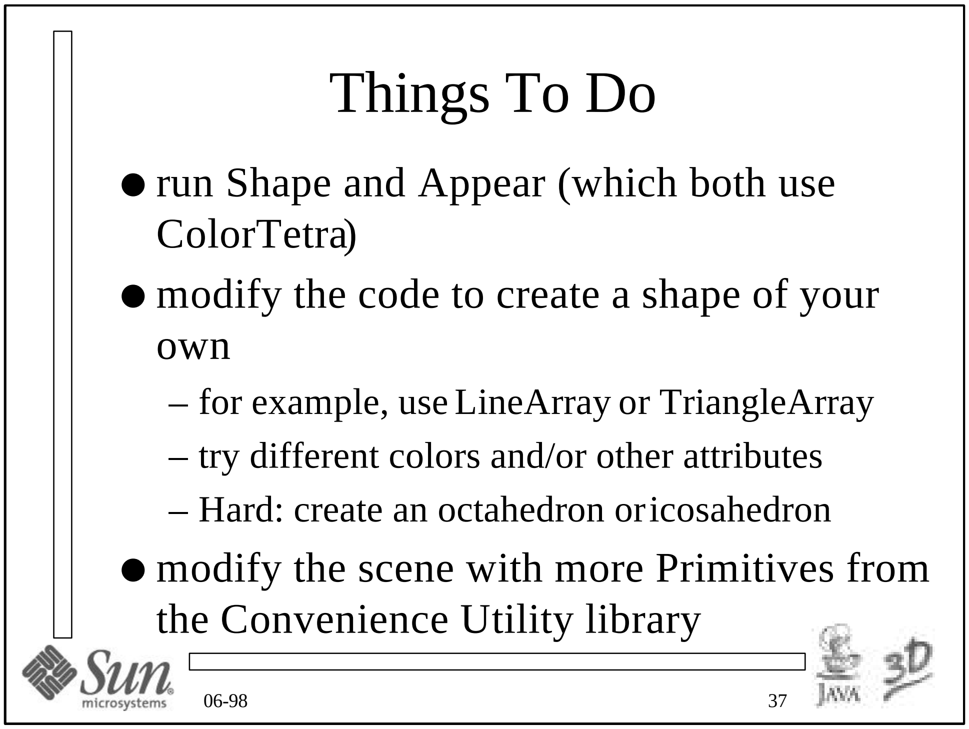
RenderingAttributes methods and constants:

```
final void setDepthBufferEnable(boolean state)
final void setDepthBufferWriteEnable(boolean state)
final void setAlphaTestValue(float value)
final void setAlphaTestFunction(int function)
```

TransparencyAttributes methods and constants:

```
final void setTransparency(float transparency)
final void setTransparencyMode(int transparencyMode)
```

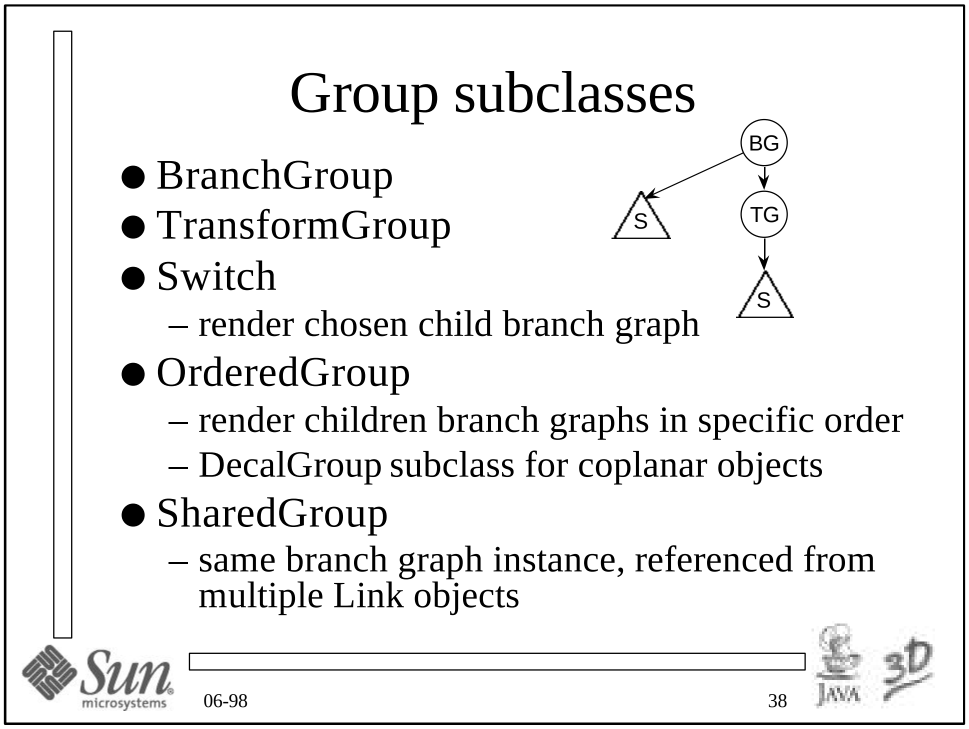
where `transparencyMode` is one of the following: `FASTEST`, `NICEST`, `SCREEN_DOOR`, `BLENDED`, or `NONE`.



Things To Do

- run Shape and Appear (which both use ColorTetra)
- modify the code to create a shape of your own
  - for example, use LineArray or TriangleArray
  - try different colors and/or other attributes
  - Hard: create an octahedron or icosahedron
- modify the scene with more Primitives from the Convenience Utility library

Student Notes

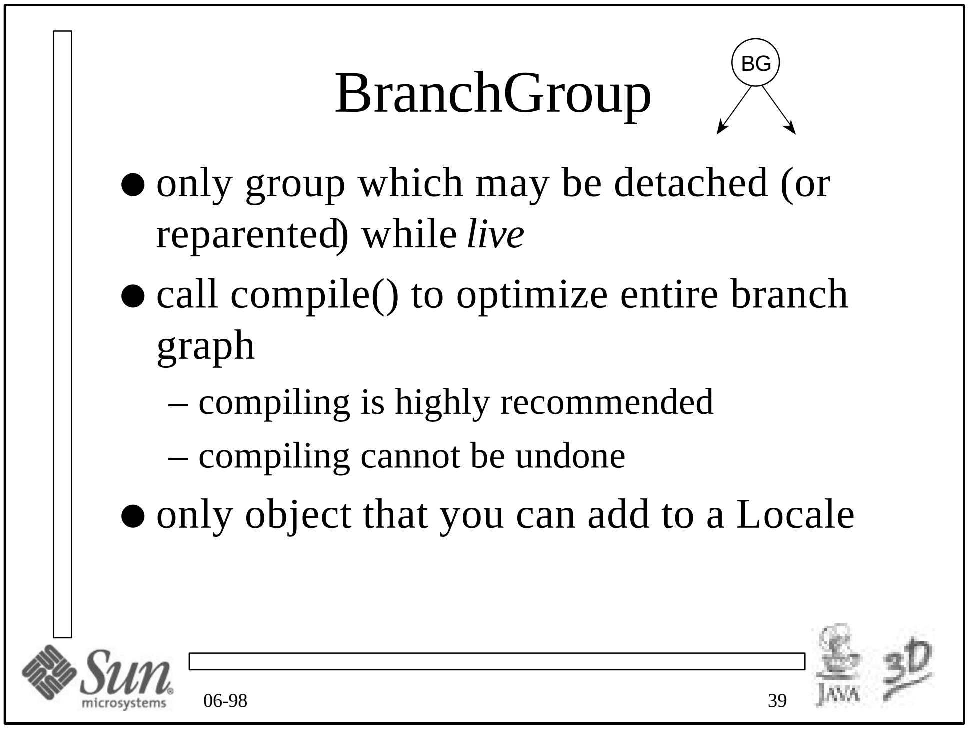


Group subclasses

- BranchGroup
- TransformGroup
- Switch
  - – render chosen child branch graph
- OrderedGroup
  - – render children branch graphs in specific order
  - – DecalGroup subclass for coplanar objects
- SharedGroup
  - – same branch graph instance, referenced from multiple Link objects

Student Notes

A Switch node has a bitmask, which can mark several children for rendering.



BranchGroup

- only group which may be detached (or reparented) while *live*
- call compile() to optimize entire branch graph
  - – compiling is highly recommended
  - – compiling cannot be undone
- only object that you can add to a Locale

06-98 39

Student Notes

BranchGroup capability:

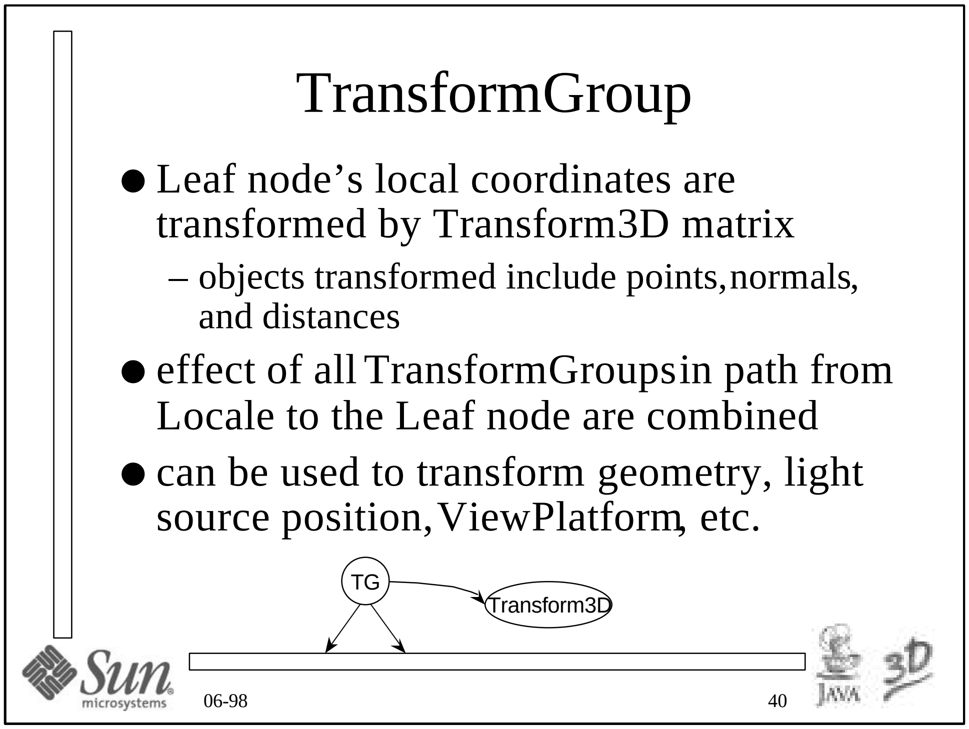
ALLOW_DETACH

BranchGroup methods (partial list):

```
final void compile()
final void detach()
```

There are also several methods related to picking.

You will often insert BranchGroups into your scene graph, just to support detachability.



TransformGroup

- Leaf node’s local coordinates are transformed by Transform3D matrix
  – objects transformed include points,normals, and distances
- effect of all TransformGroupsin path from Locale to the Leaf node are combined
- can be used to transform geometry, light source position,ViewPlatform, etc.

TG
Transform3D

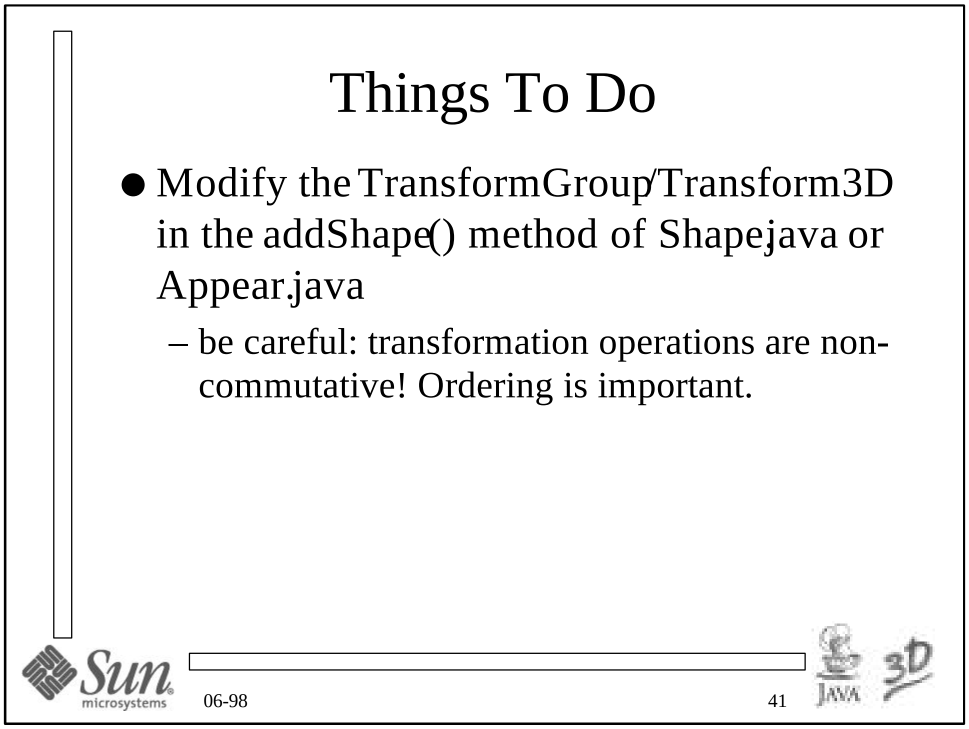
Student Notes

TransformGroup capabilities:

ALLOW_TRANSFORM_READ | WRITE

TransformGroup method (partial list):

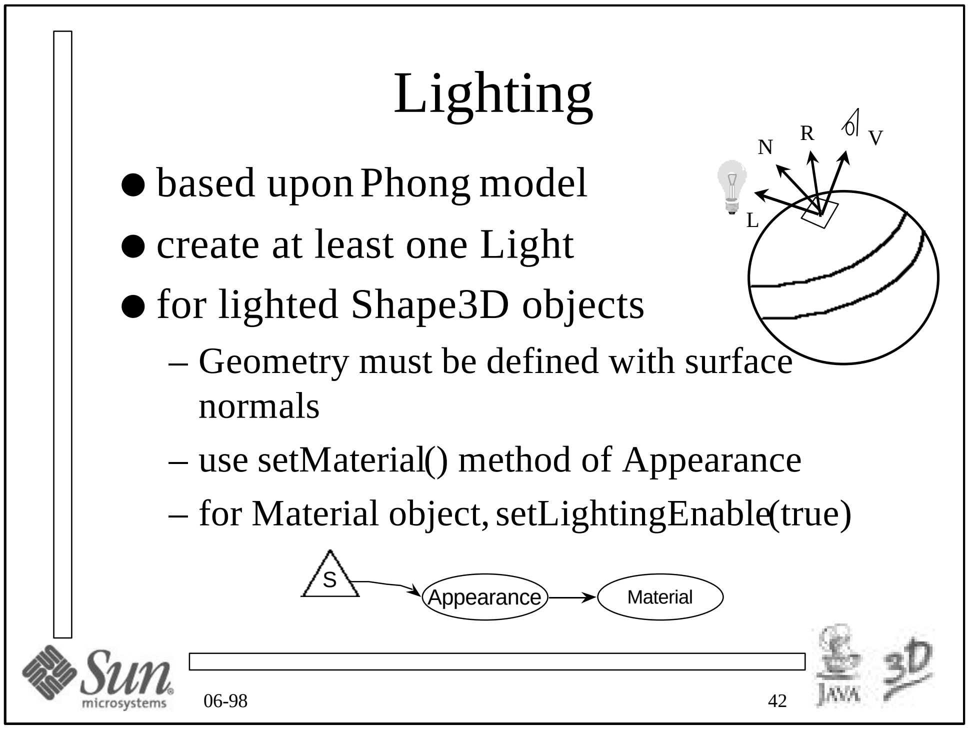
```
final void setTransform(Transform3D t1)
Node cloneNode(boolean forceDuplicate)
void duplicateNode(Node originalNode, boolean forceDuplicate)
```



Things To Do

- Modify the TransformGroup/Transform3D in the addShape() method of Shape.java or Appear.java
  - be careful: transformation operations are non-commutative! Ordering is important.

Student Notes



Lighting

- based upon Phong model
- create at least one Light
- for lighted Shape3D objects
  - – Geometry must be defined with surface normals
  - – use setMaterial() method of Appearance
  - – for Material object, setLightingEnable(true)

06-98 42

Student Notes

Java 3D Object Hierarchy

Leaf

Light

AmbientLight

DirectionalLight

PointLight

SpotLight

Light capabilities:

ALLOW_INFLUENCING_BOUNDS_READ | WRITE

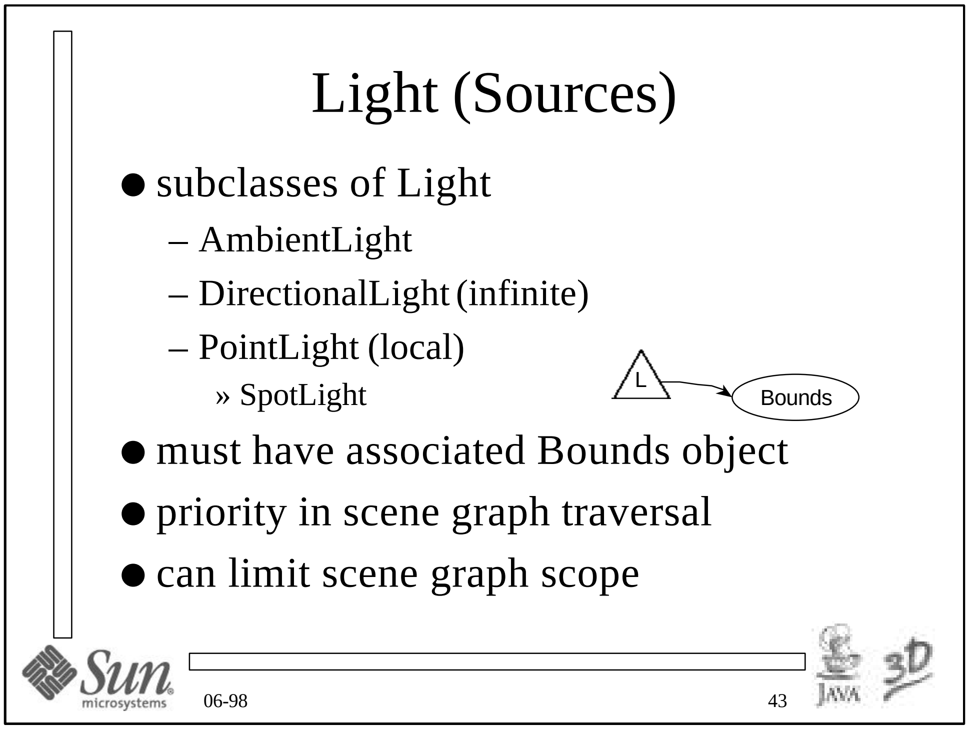
ALLOW_STATE_READ | WRITE

ALLOW_COLOR_READ | WRITE

Light methods:

```
final void setEnable(boolean state)
final void setColor(Color3f color)
final setInfluencingBounds(Bounds region)
final setInfluencingBoundingLeaf(BoundingLeaf region)
```

and several more involving scope of the light source.



Light (Sources)

- subclasses of Light
  - – AmbientLight
  - – DirectionalLight (infinite)
  - – PointLight (local)
    - » SpotLight
- must have associated Bounds object
- priority in scene graph traversal
- can limit scene graph scope

Student Notes

Make certain that the light source is associated to a Bounds object. If the light appears to have no effect, check this first.

DirectionalLight capabilities and methods:

ALLOW_DIRECTION_READ | WRITE

```
final void setDirection(...)
```

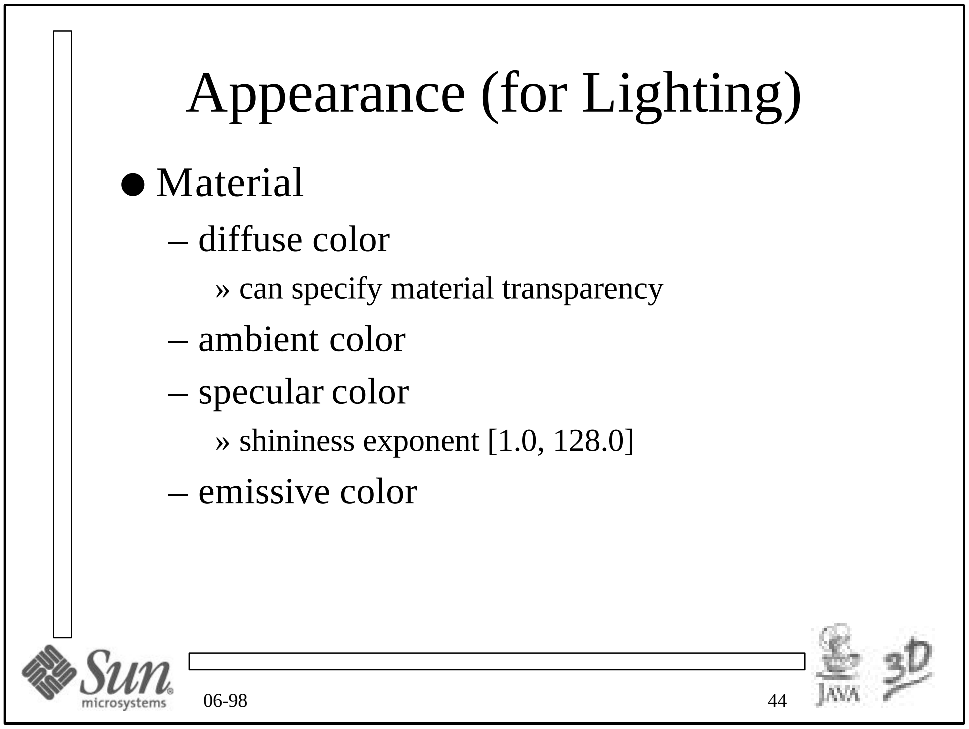
PointLight capabilities and methods:

ALLOW_POSITION_READ | WRITE

ALLOW_ATTENUATION_READ | WRITE

```
final void setPosition(...)
final void setAttenuation(...)
```

And several more capabilities and methods for SpotLight (including control of concentration and spread angle).



Appearance (for Lighting)

- Material
  - – diffuse color
    - » can specify material transparency
  - – ambient color
  - – specular color
    - » shininess exponent [1.0, 128.0]
  - – emissive color

Student Notes

Appearance capability and method (pertaining to lighting)

ALLOW_MATERIAL_READ | WRITE

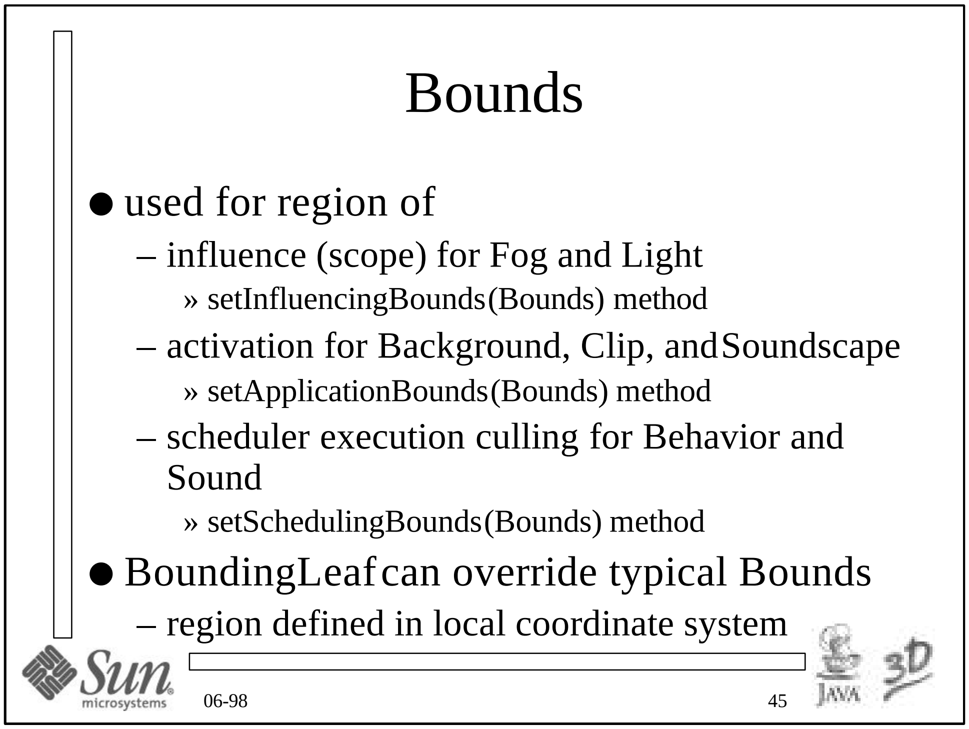
```
final void setMaterial(Material material)
```

Material constructor

```
Material(Color3f ambientColor, Color3f emmissiveColor, Color3f
diffuseColor, Color3f specularColor, float  shininess)
```

Material methods (partial list)

```
final void setLightingEnable(boolean state)
final void setAmbientColor(...)
final void setEmissiveColor(...)
final void setDiffuseColor(...)
final void setSpecularColor(...)
final void setShininess(float shininess)
```



Bounds

- used for region of
  - influence (scope) for Fog and Light
    - » setInfluencingBounds(Bounds) method
  - activation for Background, Clip, and Soundscape
    - » setApplicationBounds(Bounds) method
  - scheduler execution culling for Behavior and Sound
    - » setSchedulingBounds(Bounds) method
- BoundingLeaf can override typical Bounds
  - region defined in local coordinate system

Student Notes

Java 3D Bounds Object Hierarchy

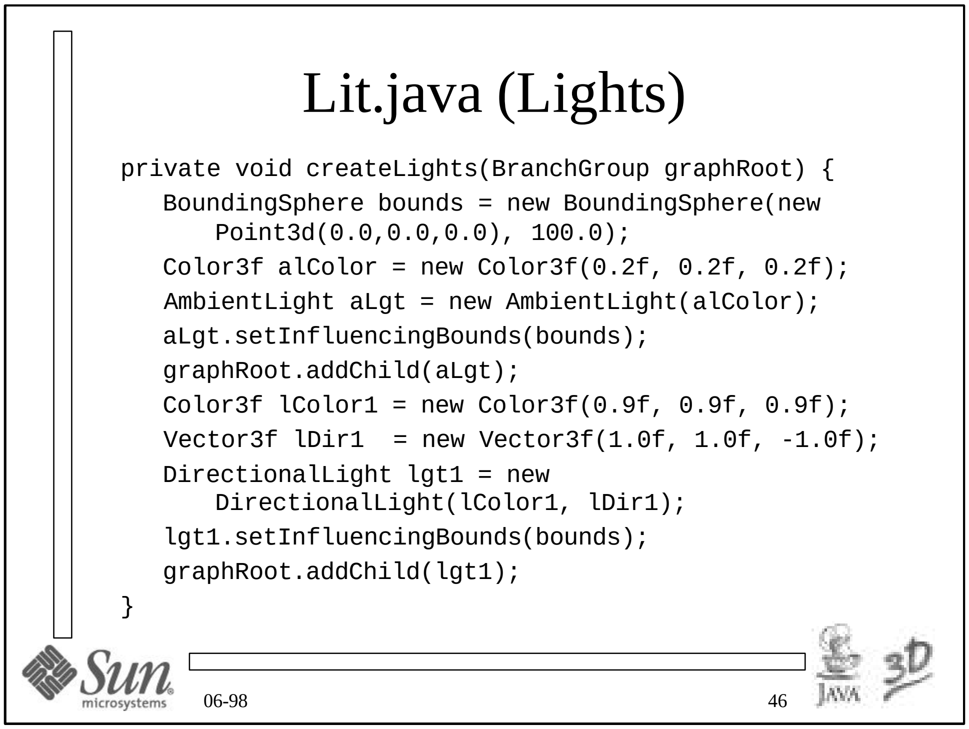
Bounds

- BoundingBox
- BoundingPolytope
- BoundingSphere

Methods common to Bounds (or subclasses) objects:

```
void set(Bounds boundsObject)
boolean intersect(...)
Bounds closestIntersection(Bounds boundsObjects[])
void combine(...)
void transform(...)
boolean isEmpty()
Object clone()
```

BoundingBox, BoundingPolytope, and BoundingSphere also have class-specific methods, based upon the shape of the bounding region (such as setRadius() or setPlanes()).



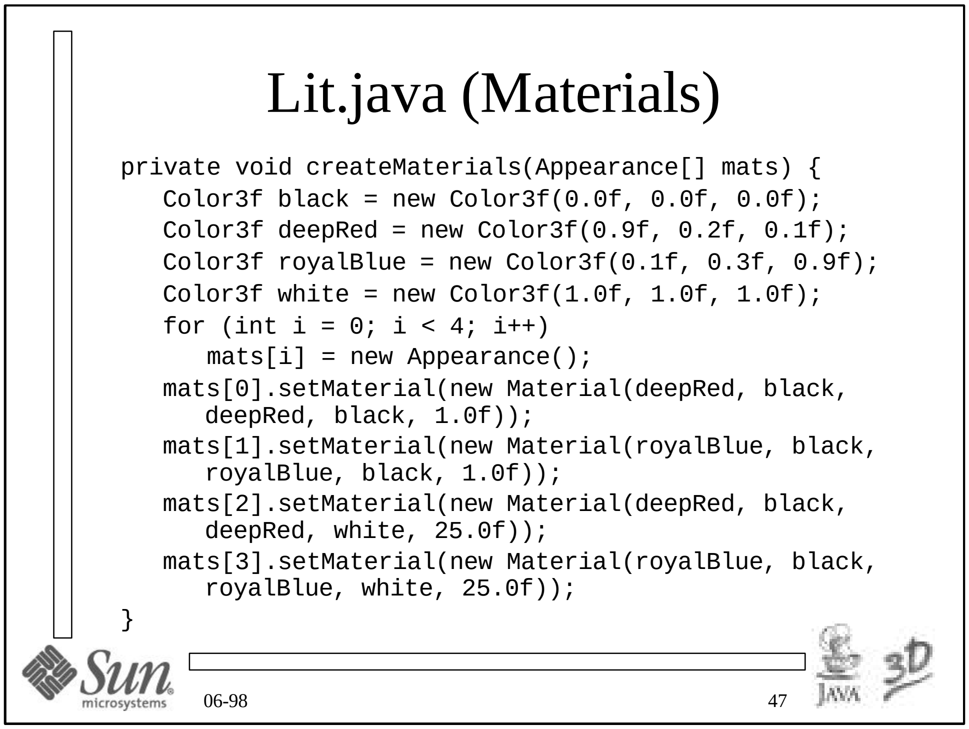
Lit.java (Lights)

```
private void createLights(BranchGroup graphRoot) {
   BoundingSphere bounds = new BoundingSphere(new
      Point3d(0.0,0.0,0.0), 100.0);
   Color3f alColor = new Color3f(0.2f, 0.2f, 0.2f);
   AmbientLight aLgt = new AmbientLight(alColor);
   aLgt.setInfluencingBounds(bounds);
   graphRoot.addChild(aLgt);
   Color3f lColor1 = new Color3f(0.9f, 0.9f, 0.9f);
   Vector3f lDir1  = new Vector3f(1.0f, 1.0f, -1.0f);
   DirectionalLight lgt1 = new
      DirectionalLight(lColor1, lDir1);
   lgt1.setInfluencingBounds(bounds);
   graphRoot.addChild(lgt1);
}
```

Student Notes

In the Lit.java example, two lights are created: one ambient and one directional (infinite).

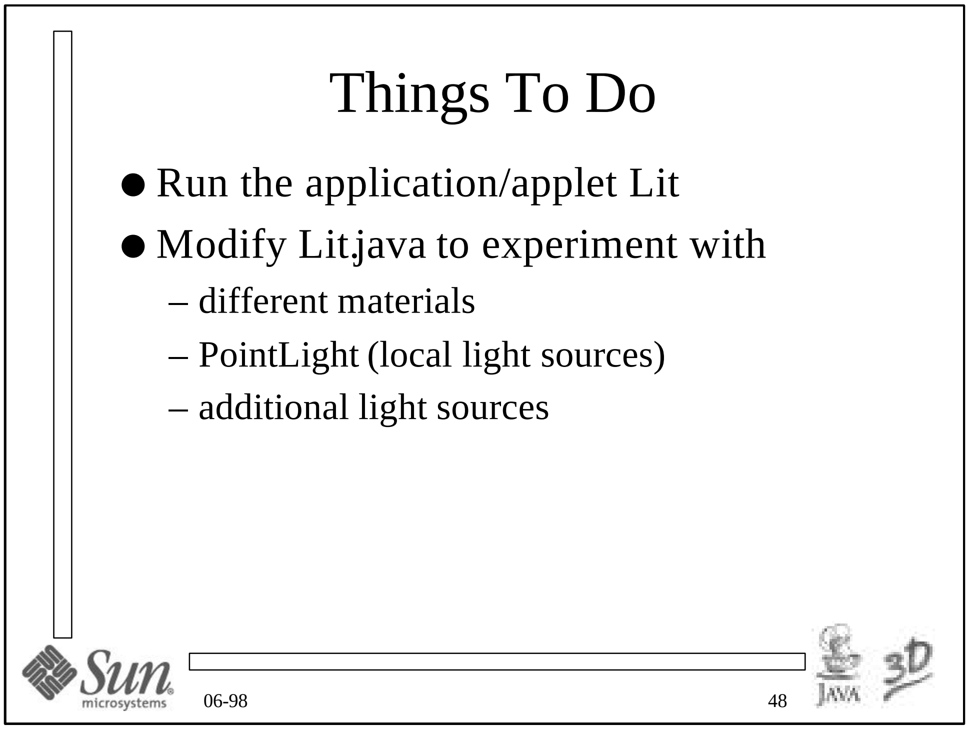
Very important : note the BoundingSphere is created first, so that both lights can use it.



Lit.java (Materials)

```
private void createMaterials(Appearance[] mats) {
   Color3f black = new Color3f(0.0f, 0.0f, 0.0f);
   Color3f deepRed = new Color3f(0.9f, 0.2f, 0.1f);
   Color3f royalBlue = new Color3f(0.1f, 0.3f, 0.9f);
   Color3f white = new Color3f(1.0f, 1.0f, 1.0f);
   for (int i = 0; i < 4; i++)
      mats[i] = new Appearance();
   mats[0].setMaterial(new Material(deepRed, black,
      deepRed, black, 1.0f));
   mats[1].setMaterial(new Material(royalBlue, black,
      royalBlue, black, 1.0f));
   mats[2].setMaterial(new Material(deepRed, black,
      deepRed, white, 25.0f));
   mats[3].setMaterial(new Material(royalBlue, black,
      royalBlue, white, 25.0f));
}
```

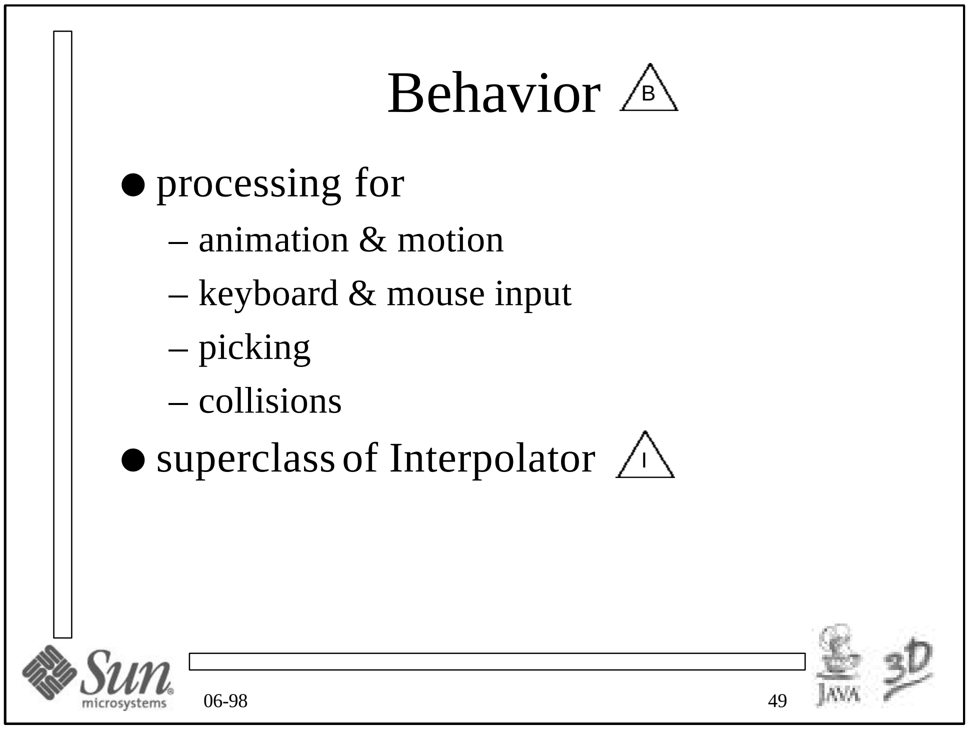
Student Notes



Things To Do

- Run the application/applet Lit
- Modify Lit.java to experiment with
  - – different materials
  - – PointLight (local light sources)
  - – additional light sources

Student Notes



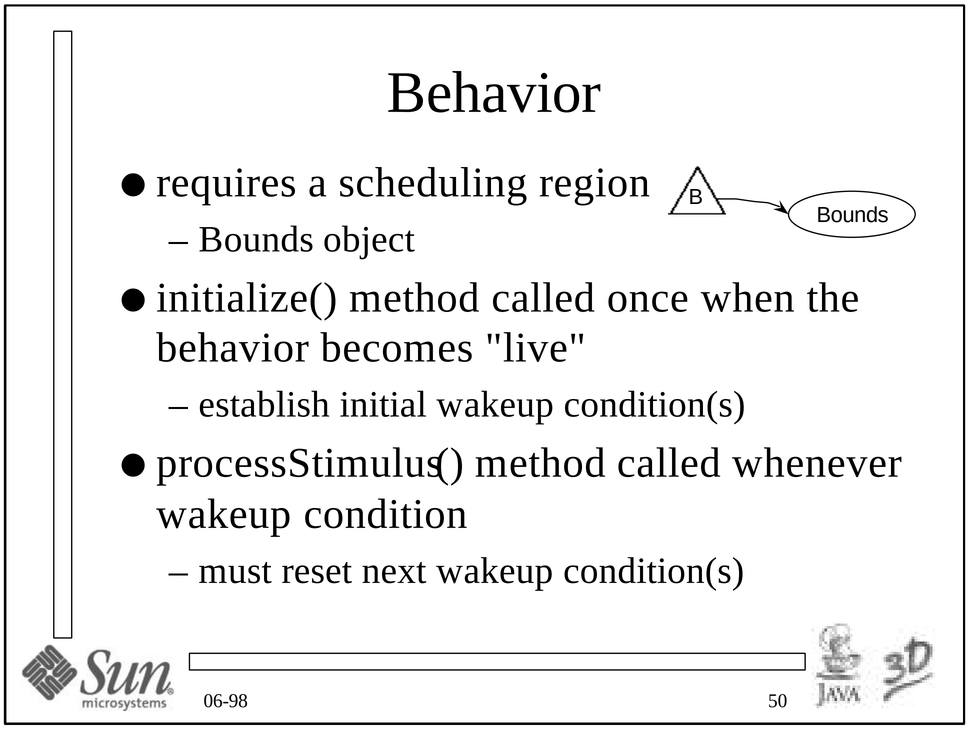
Behavior B

- processing for
  - animation & motion
  - keyboard & mouse input
  - picking
  - collisions
- superclass of Interpolator I

Student Notes

Java 3D Object Hierarchy

- Leaf
  - Behavior
    - Billboard
    - LOD
      - DistanceLOD
    - Interpolator



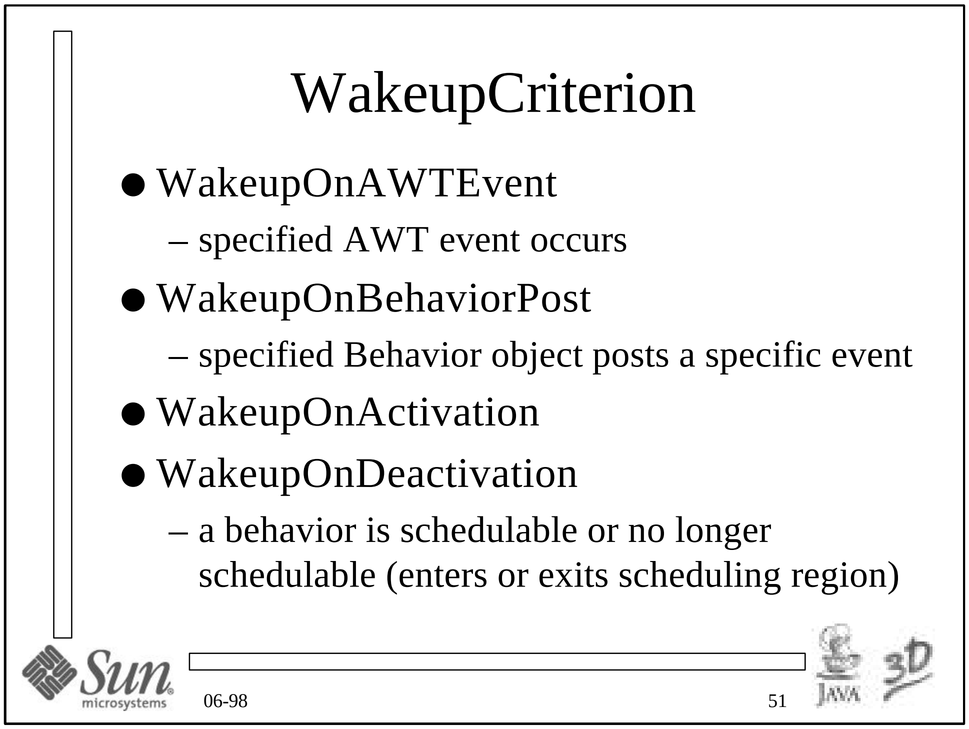
Behavior

- requires a scheduling region
  - Bounds object
- initialize() method called once when the behavior becomes "live"
  - establish initial wakeup condition(s)
- processStimulus() method called whenever wakeup condition
  - must reset next wakeup condition(s)

Student Notes

The Bounds node defines a spatial volume that serves to enable the scheduling of Behavior nodes. A Behavior node is active (can receive stimuli) whenever a ViewPlatform's activation volume intersects a Behavior object's scheduling region. Only active behaviors can receive stimuli.

WakeupCriterion is a subclass of WakeupCondition.



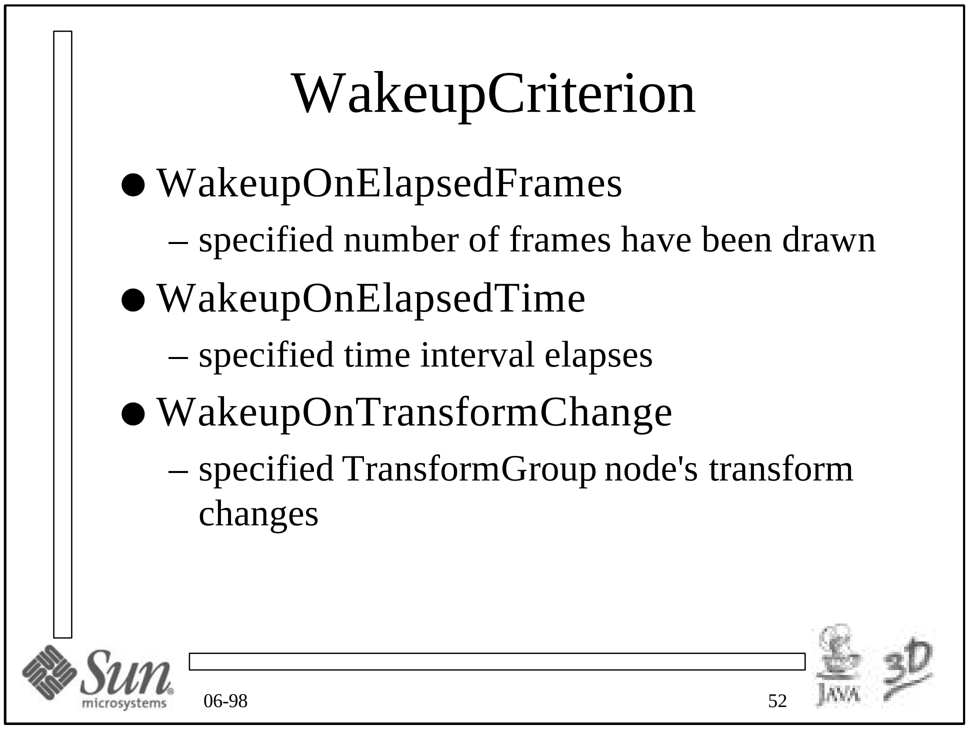
WakeupCriterion

- WakeupOnAWTEvent
  - specified AWT event occurs
- WakeupOnBehaviorPost
  - specified Behavior object posts a specific event
- WakeupOnActivation
- WakeupOnDeactivation
  - a behavior is schedulable or no longer schedulable (enters or exits scheduling region)

Student Notes

Java 3D WakeupCondition Hierarchy

- WakeupCondition
  - WakeupCriterion
    - WakeupOnAWTEvent
    - WakeupOnBehaviorPost
    - WakeupOnActivation
    - WakeupOnDeactivation
    - WakeupOnElapsedFrames
    - WakeupOnElapsedTime
    - WakeupOnSensorEntry
    - WakeupOnSensorExit
    - WakeupOnViewPlatformEntry
    - WakeupOnViewPlatformExit
    - WakeupOnTransformChange
    - WakeupOnCollisionEntry
    - WakeupOnCollisionExit
    - WakeupOnCollisionMovement
  - WakeupOr — Boolean combinations of WakeupCriterion arrays:
  - WakeupAnd
  - WakeupAndOfOrs
  - WakeupOrOfAnds



WakeupCriterion

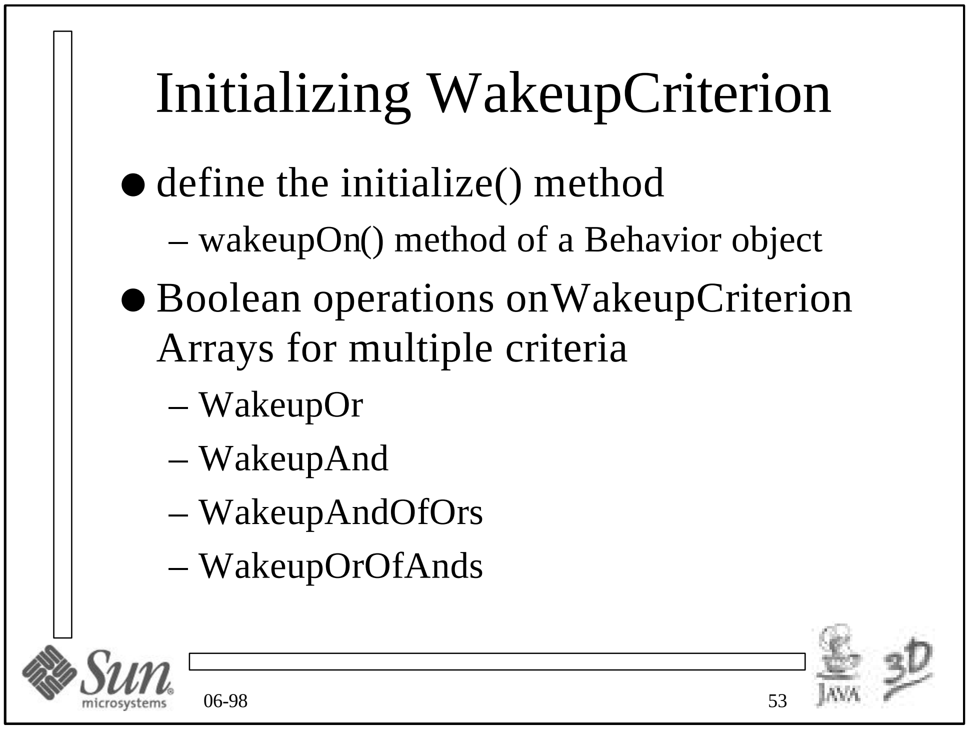
- WakeupOnElapsedFrames
  - specified number of frames have been drawn
- WakeupOnElapsedTime
  - specified time interval elapses
- WakeupOnTransformChange
  - specified TransformGroup node's transform changes

Student Notes

And a few more:

- WakeupOnSensorEntry
- WakeupOnSensorExit
  - center of a specified Sensor enters/exits a specified region
- WakeupOnViewPlatformEntry
- WakeupOnViewPlatformExit
  - center of a ViewPlatform enters/exits a specified region

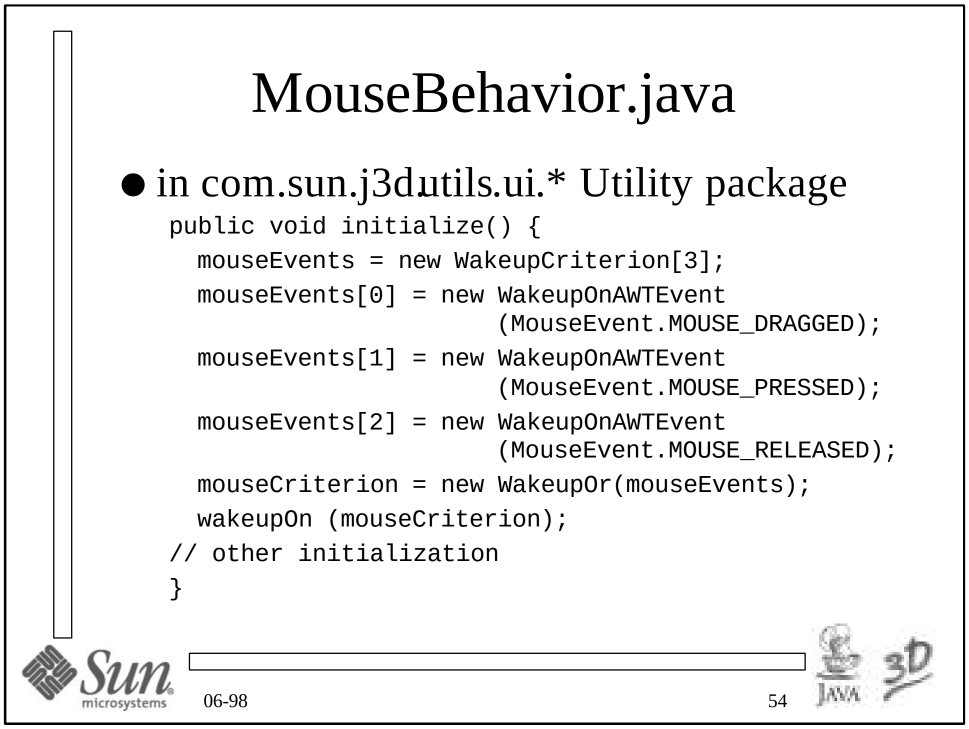
Wakeup conditions related to collision detection are covered later.



Initializing WakeupCriterion

- define the initialize() method
  - – wakeupOn() method of a Behavior object
- Boolean operations onWakeupCriterion Arrays for multiple criteria
  - – WakeupOr
  - – WakeupAnd
  - – WakeupAndOfOrs
  - – WakeupOrOfAnds

Student Notes



MouseBehavior.java

- in com.sun.j3d.utils.ui.* Utility package

```
public void initialize() {
  mouseEvents = new WakeupCriterion[3];
  mouseEvents[0] = new WakeupOnAWTEvent
                     (MouseEvent.MOUSE_DRAGGED);
  mouseEvents[1] = new WakeupOnAWTEvent
                     (MouseEvent.MOUSE_PRESSED);
  mouseEvents[2] = new WakeupOnAWTEvent
                     (MouseEvent.MOUSE_RELEASED);
  mouseCriterion = new WakeupOr(mouseEvents);
  wakeupOn (mouseCriterion);
// other initialization
}
```

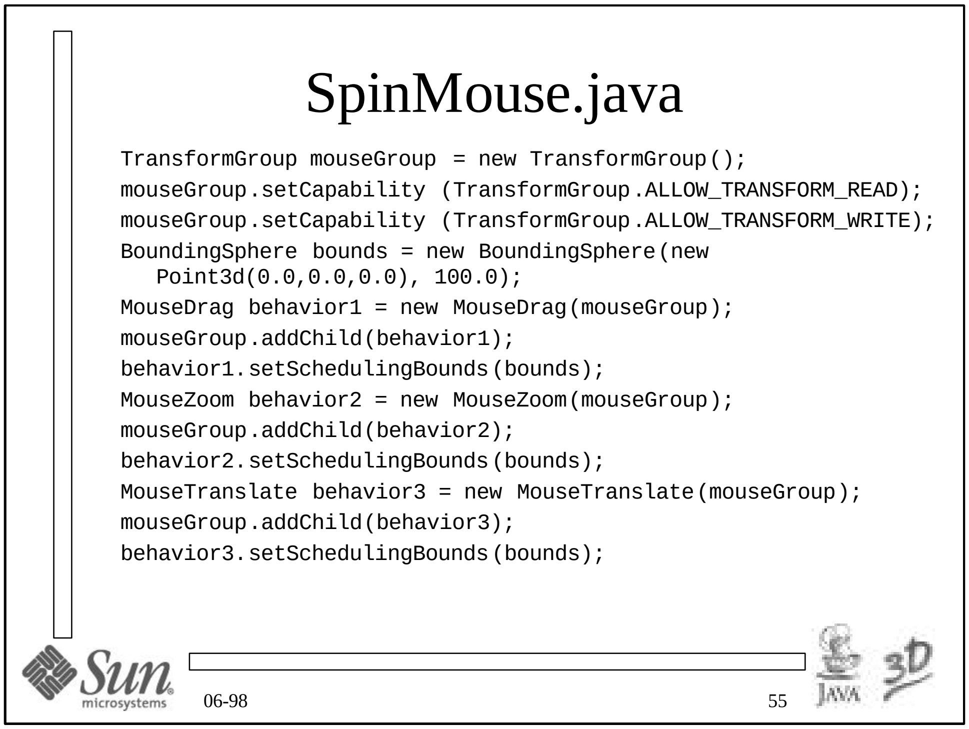
Student Notes

MouseBehavior is an abstract class in the com.sun.j3d.utils.ui.* Convenience utilities package. It is the base class used to derive the MouseDrag, MouseZoom, MouseTranslate classes. These 3 classes define the processStimulus() method to handle mouse pressing, dragging, and releasing, and to change those AWT events into matrix operations to a TransformGroup node.

MouseDrag causes the left mouse to spin (rotate) an object

MouseZoom causes the middle mouse to translate an object in z

MouseTranslate causes the right mouse to translate an object in x or y



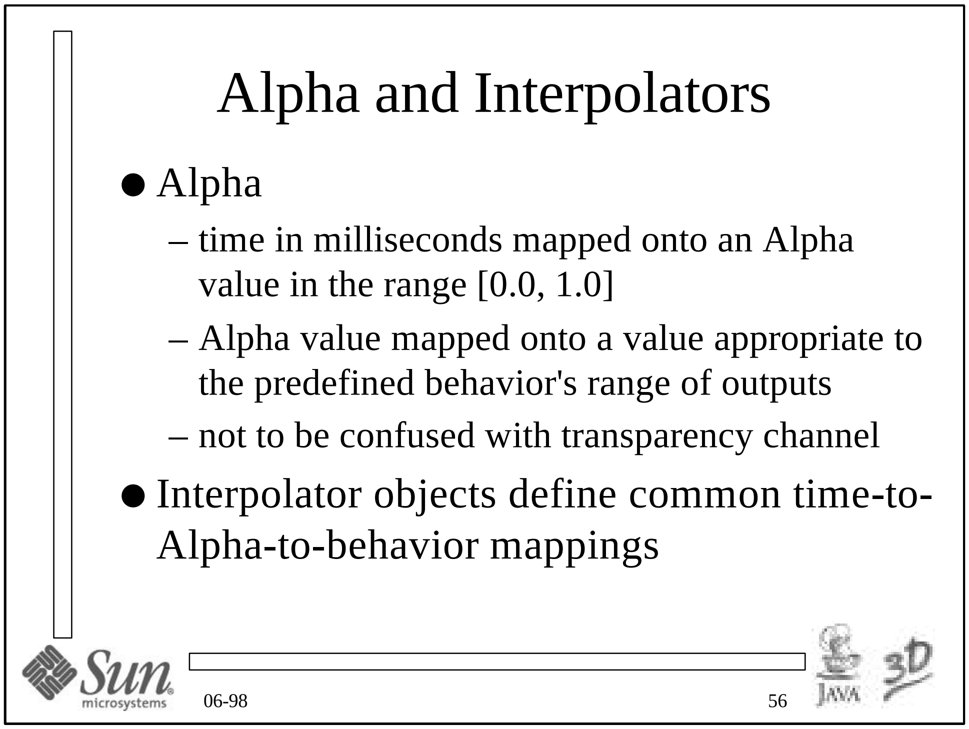
SpinMouse.java

```
TransformGroup mouseGroup = new TransformGroup();
mouseGroup.setCapability (TransformGroup.ALLOW_TRANSFORM_READ);
mouseGroup.setCapability (TransformGroup.ALLOW_TRANSFORM_WRITE);
BoundingSphere bounds = new BoundingSphere(new
   Point3d(0.0,0.0,0.0), 100.0);
MouseDrag behavior1 = new MouseDrag(mouseGroup);
mouseGroup.addChild(behavior1);
behavior1.setSchedulingBounds(bounds);
MouseZoom behavior2 = new MouseZoom(mouseGroup);
mouseGroup.addChild(behavior2);
behavior2.setSchedulingBounds(bounds);
MouseTranslate behavior3 = new MouseTranslate(mouseGroup);
mouseGroup.addChild(behavior3);
behavior3.setSchedulingBounds(bounds);
```

Student Notes

This portion of the SpinMouse.java code shows how to create objects, using the MouseDrag, MouseZoom, and MouseTranslate classes from the Convenience Utility library. The mouseGroup is the TransformGroup, which is the parent of the scene subgraph for the Geometry objects in the scene.

Note these actions are hardwired in the Convenience library to the first, second, and third mouse buttons. Also hardwired are the rates of rotation and translation for these three actions. If you want to alter these, you must create your own classes, derived from either the MouseBehavior Convenience Utility or the standard Java 3D Behavior class, for different event handling. See MyMouse.java in one of the Tennis programs for an example of how to do this.



Alpha and Interpolators

- Alpha
  - – time in milliseconds mapped onto an Alpha value in the range [0.0, 1.0]
  - – Alpha value mapped onto a value appropriate to the predefined behavior's range of outputs
  - – not to be confused with transparency channel
- Interpolator objects define common time-to-Alpha-to-behavior mappings

Student Notes

Alpha constructors:

```
Alpha(int loopCount, long triggerTime, long
phaseDelayDuration, long increasingAlphaDuration, long
increasingAlphaRampDuration, long  alphaAtOneDuration)
```

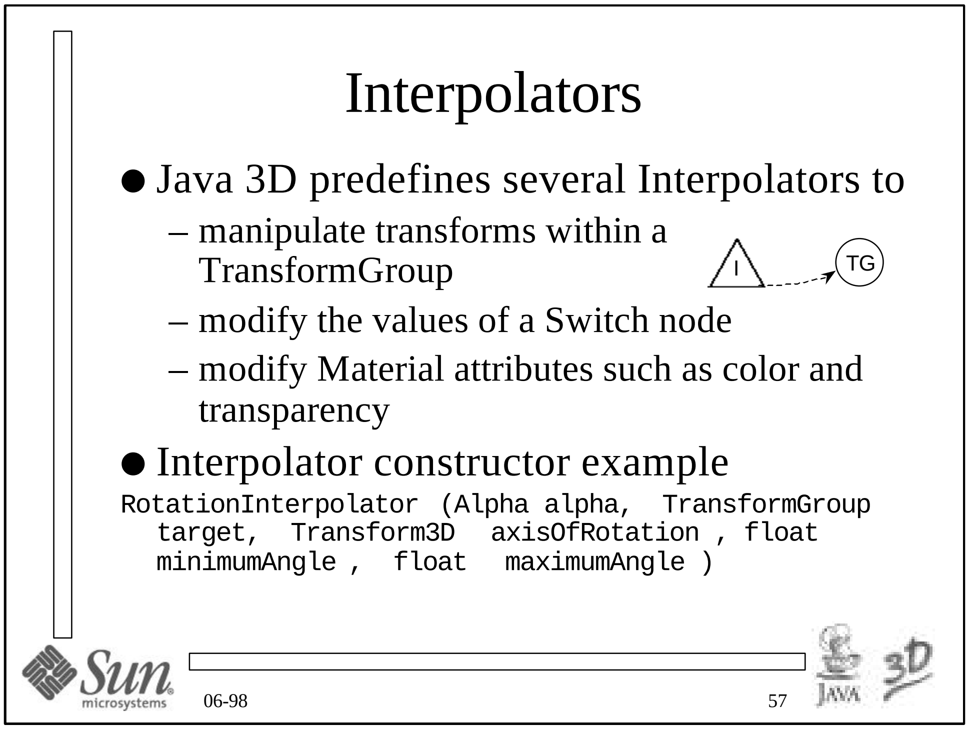
```
Alpha(int loopCount, int mode, long triggerTime, long
phaseDelayDuration, long increasingAlphaDuration, long
increasingAlphaRampDuration, long  alphaAtOneDuration, long
decreasingAlphaDuration, long  decreasingAlphaRampDuration,
long  alphaAtZeroDuration)
```

Note: loopCount of -1 means infinite loop

Alpha methods (partial list):

```
boolean finished()    // “past activity”--all looping completed
float value()         // between 0.0 and 1.0--is it, now?
float value(long atTime)
void setStartTime(long startTime)
```

Note: setting startTime to the current time restarts an Alpha object



Interpolators

- Java 3D predefines several Interpolators to
  - manipulate transforms within a TransformGroup
  - modify the values of a Switch node
  - modify Material attributes such as color and transparency
- Interpolator constructor example

```
RotationInterpolator (Alpha alpha,  TransformGroup
   target,  Transform3D  axisOfRotation , float
   minimumAngle ,  float   maximumAngle )
```

Student Notes

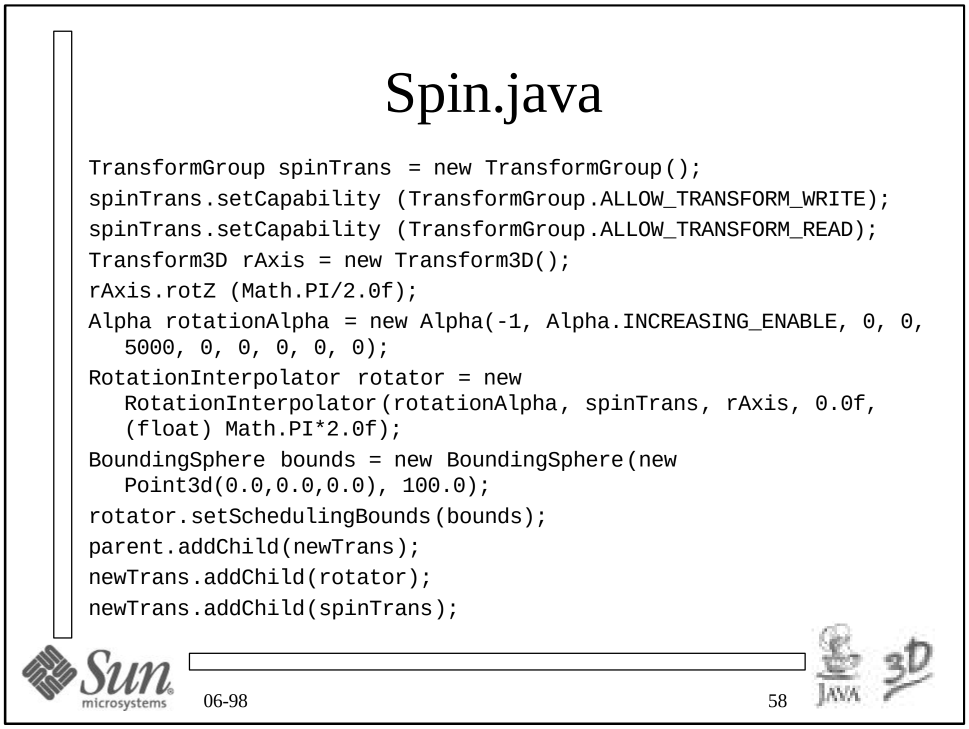
Very common to derive a special subclass of an existing Interpolator class, overriding processStimulus() method to do something special.

Java 3D Object Hierarchy

- Leaf
  - Behavior
    - Interpolator
      - ColorInterpolator
      - PositionInterpolator
      - RotationInterpolator
      - ScaleInterpolator
      - SwitchValueInterpolator
      - TransparencyInterpolator
      - PathInterpolator
        - PositionPathInterpolator
        - RotationPathInterpolator
        - RotPosPathInterpolator
        - RotPosScalePathInterpolator

Another example of an Interpolator constructor:

```
PositionPathInterpolator(Alpha alpha, TransformGroup  target,
Transform3D axisOfTranslation, float knots[], Point3f
positions[])
```



Spin.java

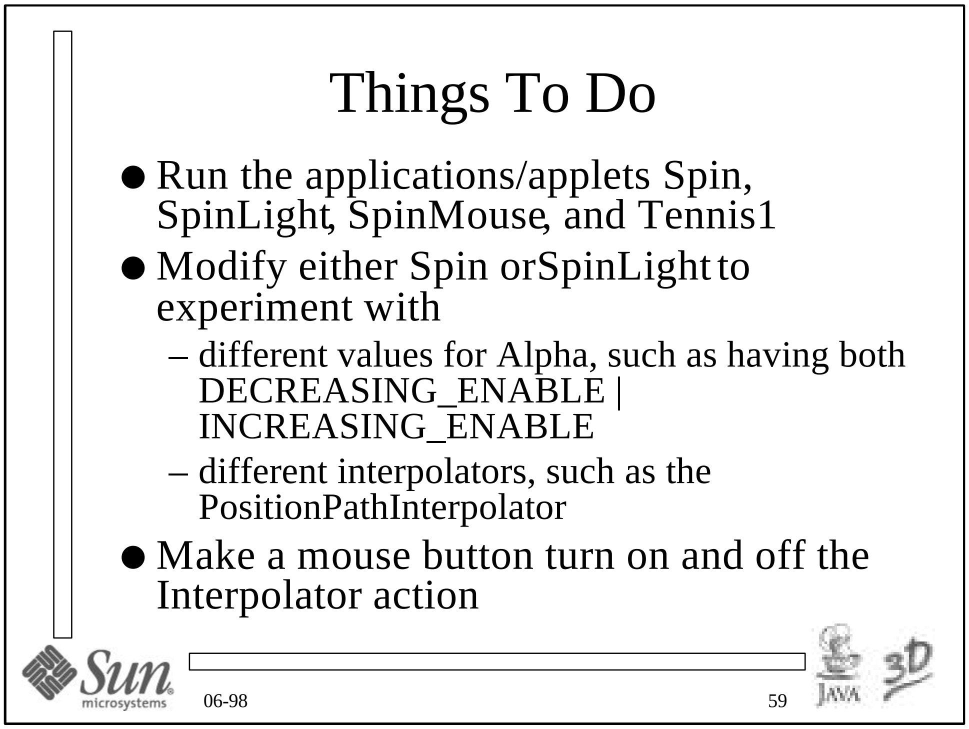
```
TransformGroup spinTrans = new TransformGroup();
spinTrans.setCapability (TransformGroup.ALLOW_TRANSFORM_WRITE);
spinTrans.setCapability (TransformGroup.ALLOW_TRANSFORM_READ);
Transform3D rAxis = new Transform3D();
rAxis.rotZ (Math.PI/2.0f);
Alpha rotationAlpha = new Alpha(-1, Alpha.INCREASING_ENABLE, 0, 0,
   5000, 0, 0, 0, 0, 0);
RotationInterpolator rotator = new
   RotationInterpolator(rotationAlpha, spinTrans, rAxis, 0.0f,
   (float) Math.PI*2.0f);
BoundingSphere bounds = new BoundingSphere(new
   Point3d(0.0,0.0,0.0), 100.0);
rotator.setSchedulingBounds(bounds);
parent.addChild(newTrans);
newTrans.addChild(rotator);
newTrans.addChild(spinTrans);
```

Student Notes

Spin.java:

- creates a TransformGroup, spinTrans, which can be read and write, while live.
- creates an Alpha object that loops infinitely and takes 5 seconds (5000 milliseconds) to linearly cycle from 0.0 to 1.0.
- creates a new Behavior object (RotationInterpolator) that performs the desired operation on the specified transform object.
- creates a Bounds object for when to schedule the Behavior.
- adds the Behavior object into the scene graph.

Note neither the BoundingSphere, nor the Alpha are added directly to the scene graph. Rather the RotationInterpolator references both those objects.



Things To Do

- Run the applications/applets Spin, SpinLight, SpinMouse, and Tennis1
- Modify either Spin orSpinLight to experiment with
  - different values for Alpha, such as having both DECREASING_ENABLE | INCREASING_ENABLE
  - different interpolators, such as the PositionPathInterpolator
- Make a mouse button turn on and off the Interpolator action

Student Notes

Tennis1 is an applet/application which shows a couple of table tennis paddles. When the left mouse is pressed, moving the mouse in the x direction moves the nearest paddle from left to right. Pressing the middle mouse causes a ball to be launched. Otherwise, it’s pretty boring, because there are no collision detection and therefore no volleying. Tennis2 will add collision detection.

Tennis1 consists of several classes. Here are a list of those classes, with brief descriptions:

Tennis1: where all the initial construction takes place. The scene graph is started from here, light sources are initialized, and main() is here.

Ball: a Sphere which represents the ping-pong ball.

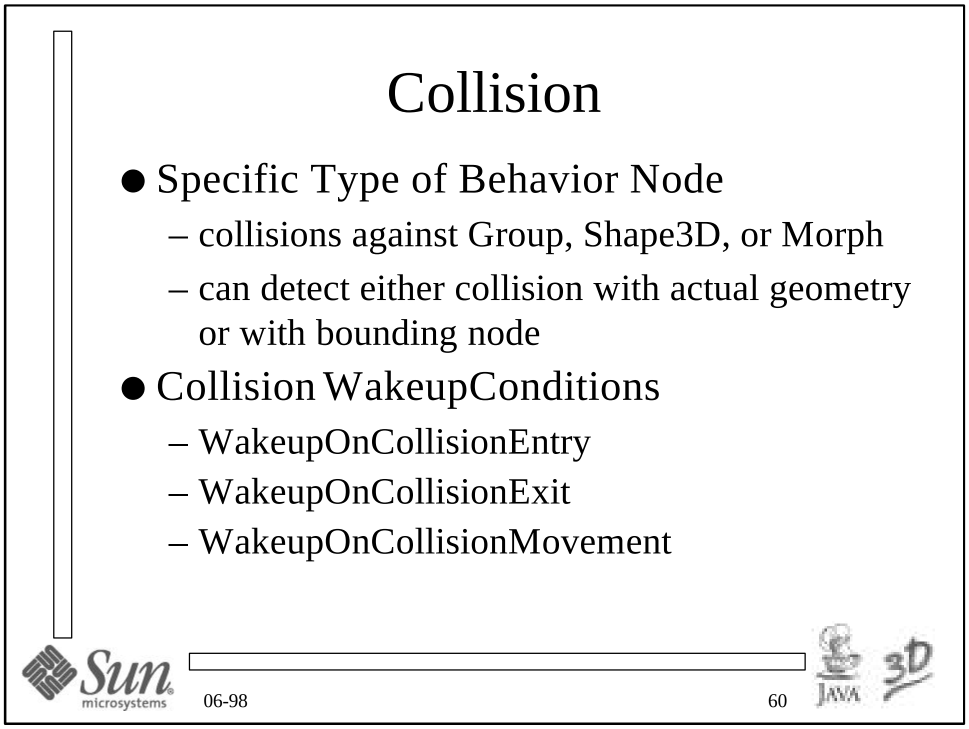
Paddle: three cylinders (with different Material objects) form a table tennis paddle shape.

BallInterp: an Interpolator which moves the Ball along a path.

RandPos: choose a random position for objects to be moving towards.

MyMouse: wakes up on left mouse input and controls position of near paddle.

MouseFire: wakes up on middle mouse; creates and moves Ball. This is the only class that changes between Tennis1 and Tennis2.



Collision

- Specific Type of Behavior Node
  - collisions against Group, Shape3D, or Morph
  - can detect either collision with actual geometry or with bounding node
- Collision WakeupConditions
  - WakeupOnCollisionEntry
  - WakeupOnCollisionExit
  - WakeupOnCollisionMovement

Student Notes

Java 3D Collision Criterion Hierarchy

- WakeupCondition
  - WakeupCriterion
    - WakeupOnCollisionEntry
    - WakeupOnCollisionExit
    - WakeupOnCollisionMovement

Explanations of Collision WakeupConditions

- WakeupOnCollisionEntry

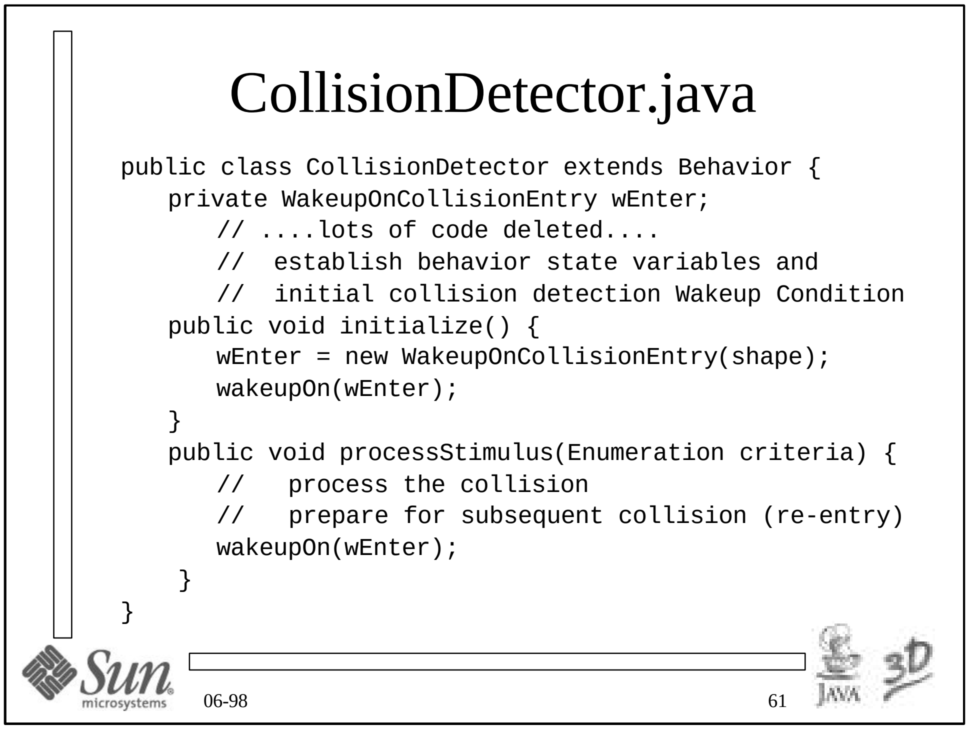
  Collision detected between a specified geometry or bounding object and any other object

- WakeupOnCollisionExit

  Specified geometry or bounding object no longer collides with any other object

- WakeupOnCollisionMovement

  Movement occurs between a specified geometry or bounding object and any other object with which it has already collided

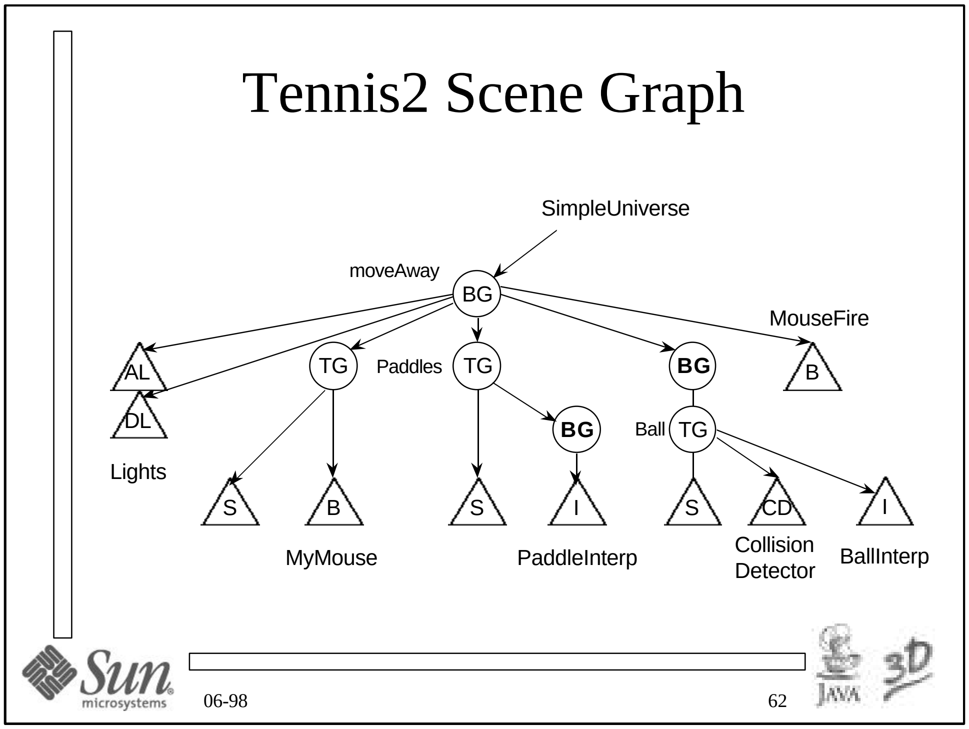


CollisionDetector.java

```
public class CollisionDetector extends Behavior {
   private WakeupOnCollisionEntry wEnter;
      // ....lots of code deleted....
      //  establish behavior state variables and
      //  initial collision detection Wakeup Condition
   public void initialize() {
      wEnter = new WakeupOnCollisionEntry(shape);
      wakeupOn(wEnter);
   }
   public void processStimulus(Enumeration criteria) {
      //   process the collision
      //   prepare for subsequent collision (re-entry)
      wakeupOn(wEnter);
    }
}
```

Student Notes

This is a template for a CollisionDetector class. A fleshed out example is in CollisionDetector.java, as part of the Tennis2 program.

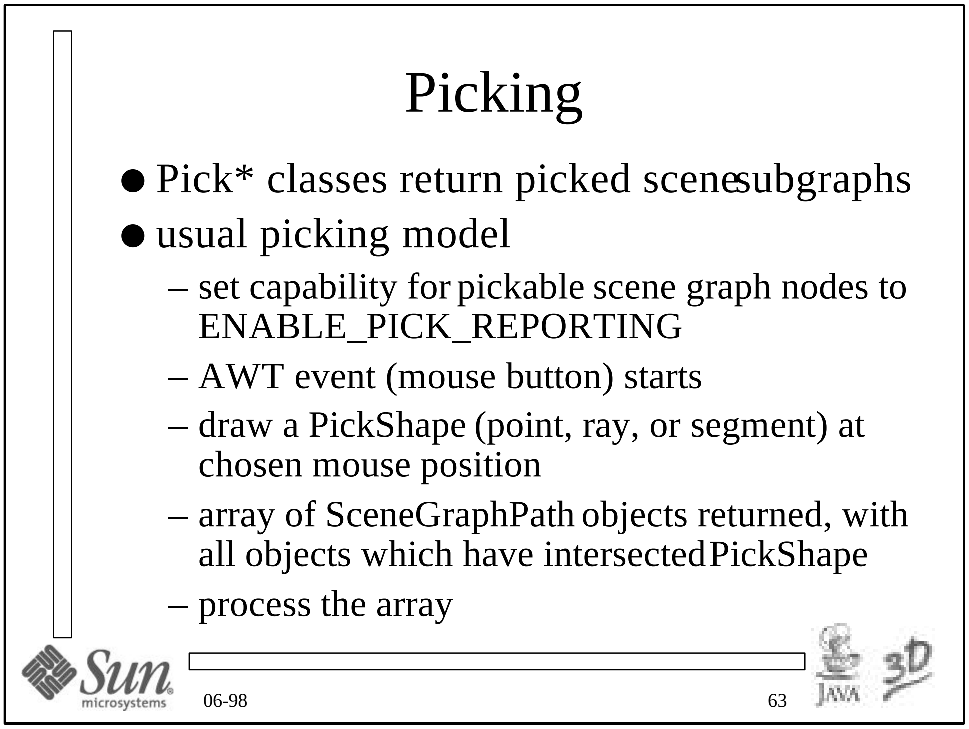


Tennis2 Scene Graph

Student Notes

This is the scene graph for the Tennis2 program, only while the Ball is in motion. The two highlighted BranchGroup objects are added to the scene graph when the Ball is active. When the Ball goes out of play, these two subgraphs are detached.

Both the Paddle and Ball classes are derived from the TransformGroup. The resulting objects consist of a TransformGroup atop one or more Shape3D nodes and associated Appearance objects.



Picking

- ● Pick* classes return picked scenesubgraphs
- ● usual picking model
  - – set capability for pickable scene graph nodes to ENABLE_PICK_REPORTING
  - – AWT event (mouse button) starts
  - – draw a PickShape (point, ray, or segment) at chosen mouse position
  - – array of SceneGraphPath objects returned, with all objects which have intersected PickShape
  - – process the array

Student Notes

Java 3D Picking Shape Hierarchy

PickShape

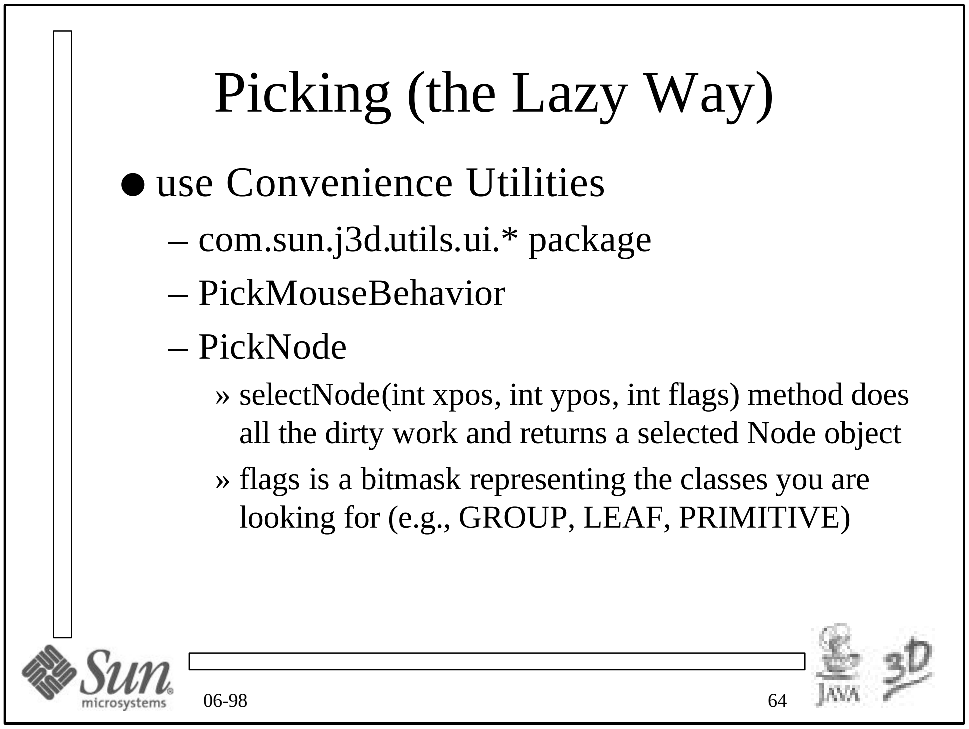
- PickPoint
- PickRay
- PickSegment

A SceneGraphPath object represents the path from an object to a BranchGroup or Locale parent.

The following are BranchGroup and Locate class methods that are related to picking:

```
final SceneGraphPath pickAny(PickShape pickShape)
final SceneGraphPath pickClosest(PickShape pickShape)
final SceneGraphPath[] pickAll(PickShape pickShape)
final SceneGraphPath[] pickAllSorted(PickShape pickShape)
```

Sorted objects are returned in order, starting with objects closest to the ViewPlatform.



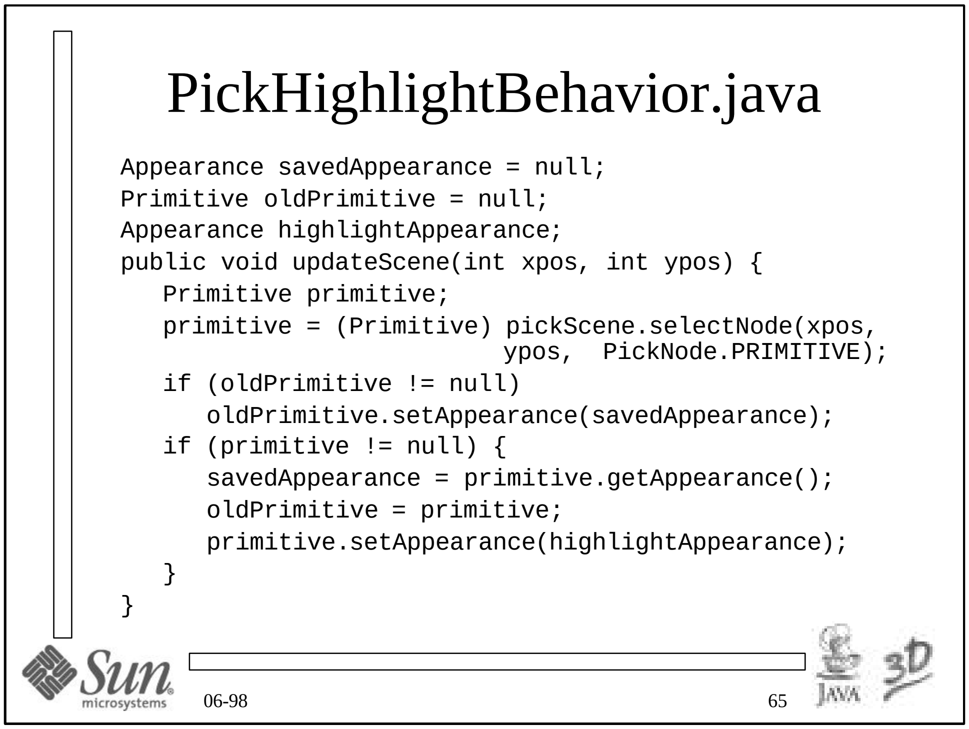
Picking (the Lazy Way)

- use Convenience Utilities
  - com.sun.j3d.utils.ui.* package
  - PickMouseBehavior
  - PickNode
    - selectNode(int xpos, int ypos, int flags) method does all the dirty work and returns a selected Node object
    - flags is a bitmask representing the classes you are looking for (e.g., GROUP, LEAF, PRIMITIVE)

Student Notes

Even if you insist on doing all the dirty work yourself, you should still look at the source code for the PickNode and PickMouseBehavior classes to analyze how they convert the mouse input into a picked scene graph Node. Of particular interest, is the use of three Canvas3D methods:

```
getCenterEyeInImagePlate (eyePosn)
getPixelLocationInImagePlate (xpos, ypos, mousePosn)
getImagePlateToVworld (motion)
```



PickHighlightBehavior.java

```
Appearance savedAppearance = null;
Primitive oldPrimitive = null;
Appearance highlightAppearance;
public void updateScene(int xpos, int ypos) {
   Primitive primitive;
   primitive = (Primitive) pickScene.selectNode(xpos,
                             ypos,  PickNode.PRIMITIVE);
   if (oldPrimitive != null)
      oldPrimitive.setAppearance(savedAppearance);
   if (primitive != null) {
      savedAppearance = primitive.getAppearance();
      oldPrimitive = primitive;
      primitive.setAppearance(highlightAppearance);
   }
}
```

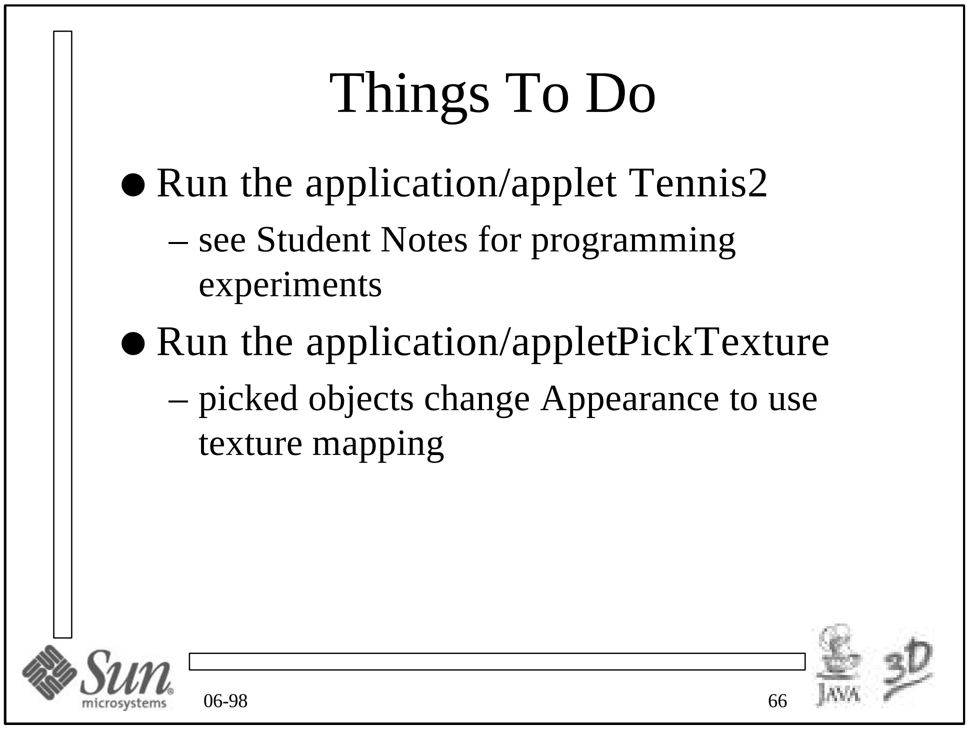
06-98

65

Student Notes

PickHighlightBehavior.java is part of the PickTexture application/applet. It is an update of the Lit (lighted shapes) application/applet. When the left mouse is pressed while the cursor is over an object, the Appearance object is changed, so the object appears textured. When another object is chosen, the “oldPrimitive” is restored to its original appearance. If nothing is picked (the mouse is pressed while over the background), then the oldPrimitive is restored, but no shape is currently highlighted.

PickNode is part of the Convenience Utility library.



Things To Do

- Run the application/applet Tennis2
  - see Student Notes for programming experiments
- Run the application/appletPickTexture
  - picked objects change Appearance to use texture mapping

Student Notes

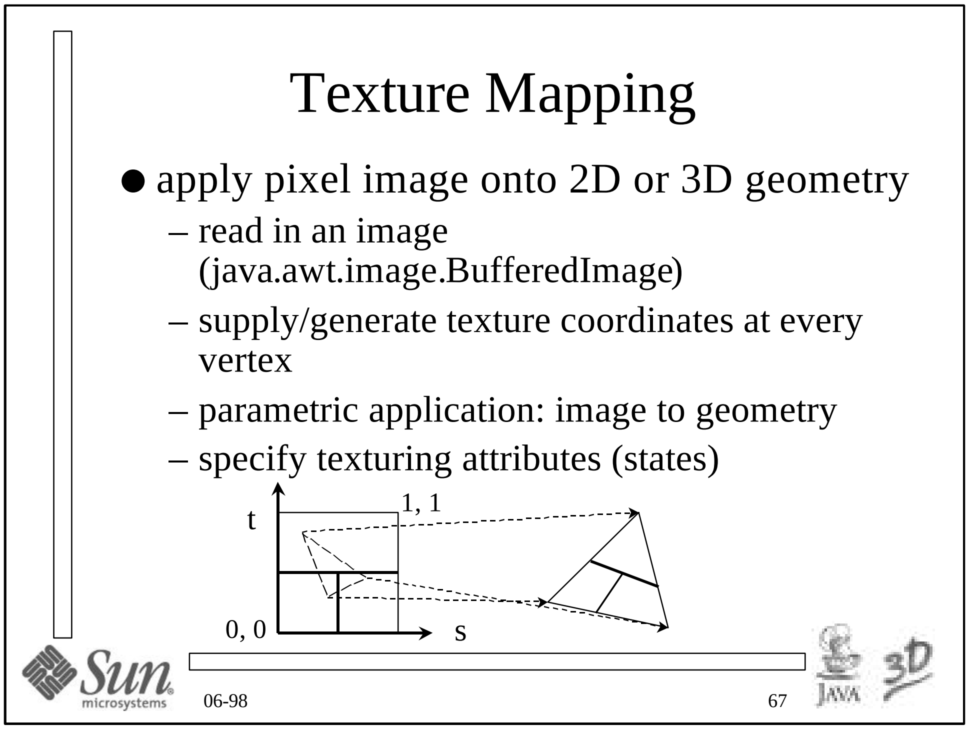
Things to do with Tennis2 (only the first is easy to do)

• Right now, only one ball can be active at a time. Change MouseFire.java to allow several balls in motion. (Note: you’ll have to make sure only one PaddleInterpolator is affecting the robot paddle.)

• Instead of using the middle mouse to fire a ball into the scene, add a Shape3D object to represent a button. When the button is picked, fire the ball and remove the button from the screen.

• Right now, volleying stops when the Alpha for the BallInterpolator reaches the end of its cycle (when finished() returns true). Create bounding objects which represent a wall “behind” the two paddles. Use collision with these walls to determine when to delete the ball.

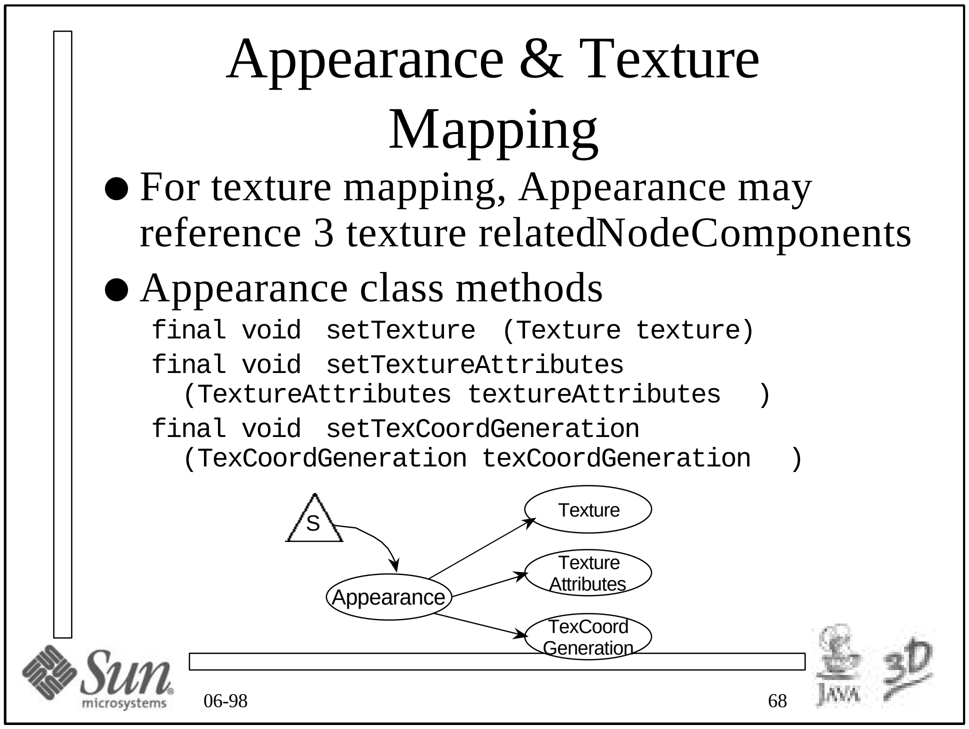
• Also there are no side walls, so the ball does not bounce off the sides. Create some side walls. Then either use collision with these walls to determine when to delete the ball.



Texture Mapping

- apply pixel image onto 2D or 3D geometry
  - – read in an image (java.awt.image.BufferedImage)
  - – supply/generate texture coordinates at every vertex
  - – parametric application: image to geometry
  - – specify texturing attributes (states)

Student Notes



Appearance & Texture Mapping

- For texture mapping, Appearance may reference 3 texture relatedNodeComponents
- Appearance class methods

```
final void  setTexture  (Texture texture)
final void  setTextureAttributes
  (TextureAttributes textureAttributes  )
final void  setTexCoordGeneration
  (TexCoordGeneration texCoordGeneration  )
```

S
Appearance
Texture
Texture Attributes
TexCoord Generation

06-98

68

Student Notes

Appearance capabilities (for texturing):

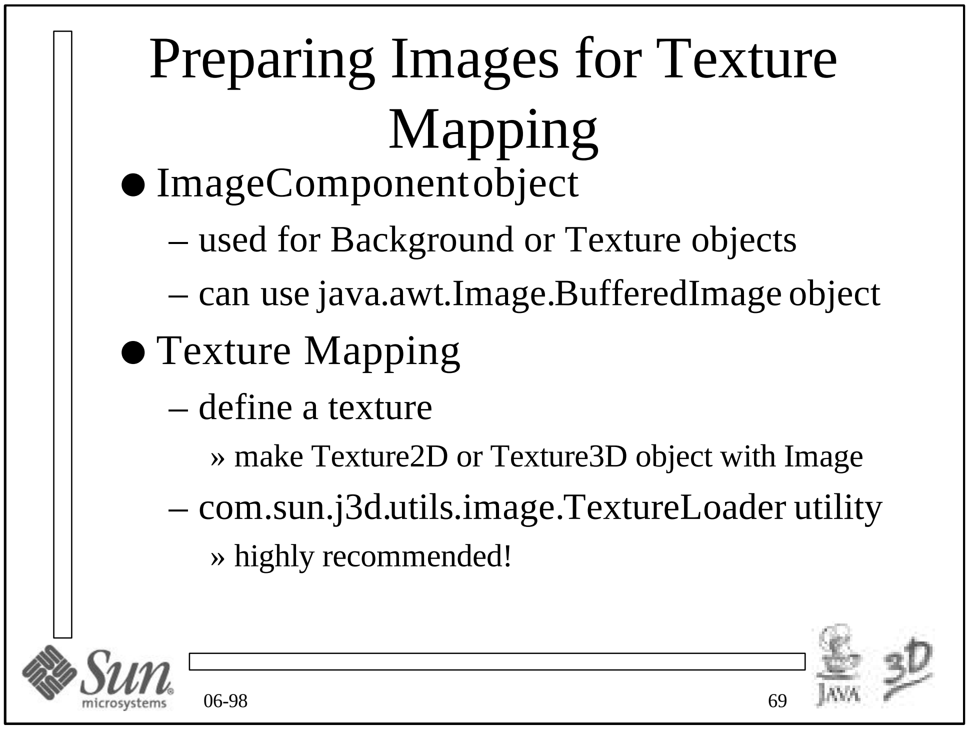
ALLOW_TEXTURE_READ | WRITE

ALLOW_TEXGEN_READ | WRITE

ALLOW_TEXTURE_ATTRIBUTES_READ | WRITE

Appearance methods (for texturing):

```
final void setTexture (Texture texture)
final void setTextureAttributes (TextureAttributes
textureAttributes)
final void setTexCoordGeneration (TexCoordGeneration
texCoordGeneration)
```



Preparing Images for Texture Mapping

- ImageComponent object
  - – used for Background or Texture objects
  - – can use java.awt.Image.BufferedImage object
- Texture Mapping
  - – define a texture
    - » make Texture2D or Texture3D object with Image
  - – com.sun.j3d.utils.image.TextureLoader utility
    - » highly recommended!

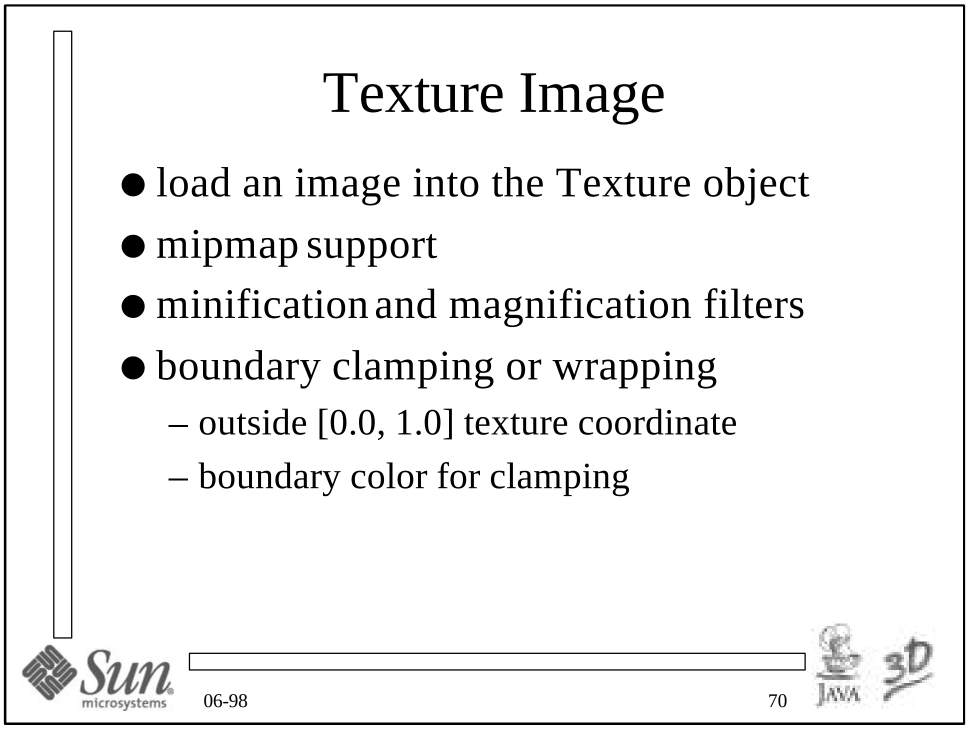
Student Notes

Java 3D Image Component Hierarchy

- NodeComponent
  - ImageComponent
    - ImageComponent2D
    - ImageComponent3D

ImageComponent2D and 3D methods:

```
final int getWidth()
final int getHeight()
final int getDepth()  // 3D only
final int getFormat() // lots of internal pixel formats
final void set(Image) // copies buffered image into object
```



Texture Image

- load an image into the Texture object
- mipmap support
- minification and magnification filters
- boundary clamping or wrapping
  - – outside [0.0, 1.0] texture coordinate
  - – boundary color for clamping

Student Notes

Java 3D Texture Image Hierarchy

```
NodeComponent
        Texture
                Texture2D
                Texture3D
```

Texture2D and 3D methods

```
final void setEnable(boolean state)
final void setImage(int level, ImageComponent image)
```

where `level` is the mipmap level

```
final void setMipMapMode(int mipmapMode)
```

where `mipmapMode` is either `BASE_LEVEL` (no mipmap) or `MULTI_LEVEL_MIP_MAP`

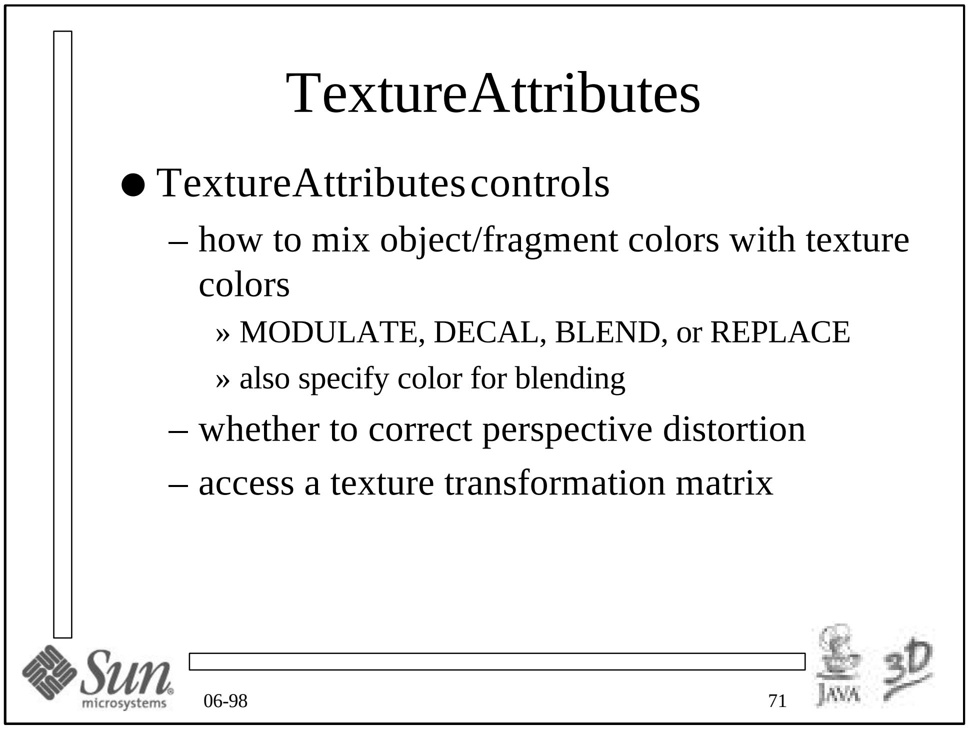
```
final void setMinFilter(int minFilter)
final void setMagFilter(int magFilter)
```

where the filter is one of `FASTEST, NICEST, BASE_LEVEL_POINT, BASE_LEVEL_LINEAR, MULTI_LEVEL_POINT, MULTI_LEVEL_LINEAR` (multi level mipmap only for minification filter)

`final void setBoundaryModeS (int boundaryModeS)` or `T` or `R`

where the `boundaryMode` for the S, T, or R coordinates is either `CLAMP` or `WRAP`

```
final void setBoundaryColor(...)
```



TextureAttributes

- TextureAttributes controls
  - how to mix object/fragment colors with texture colors
    - MODULATE, DECAL, BLEND, or REPLACE
    - also specify color for blending
  - whether to correct perspective distortion
  - access a texture transformation matrix

Student Notes

TextureAttributes capabilities:

ALLOW_MODE_READ | WRITE

ALLOW_BLEND_COLOR_READ | WRITE

ALLOW_TRANSFORM_READ | WRITE

TextureAttributes methods (partial list):

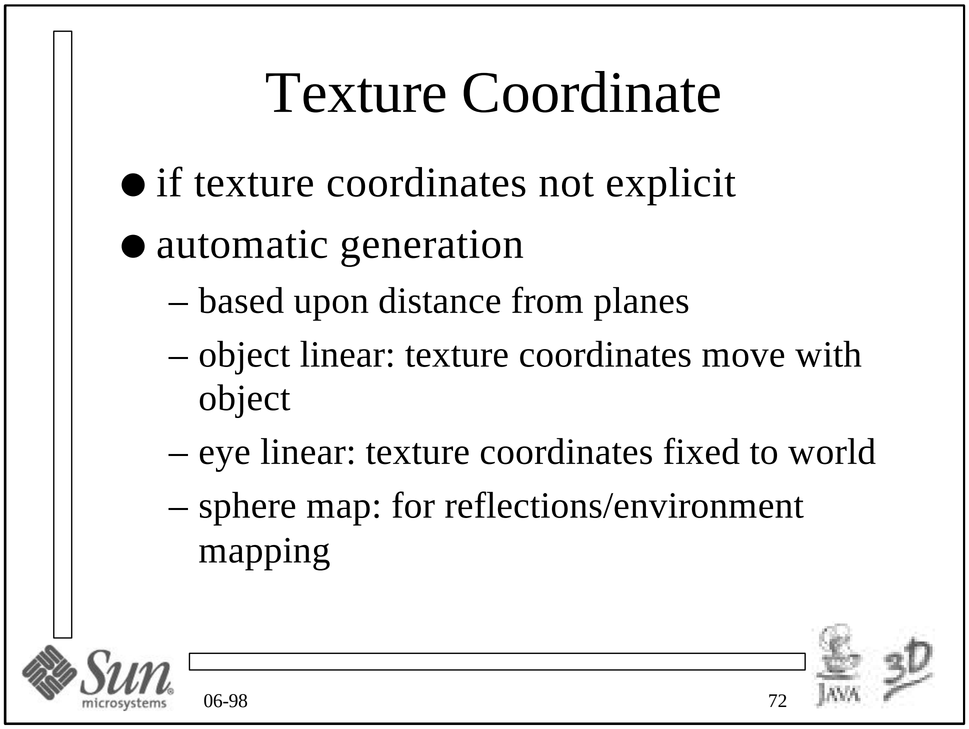
```
final void setTextureMode(int textureMode)
```

where textureMode is one of: MODULATE, DECAL, BLEND, or REPLACE.

```
final void setPerspectiveCorrectionMode(int mode)
```

where mode is one of NICEST or FASTEST.

```
final void setTextureBlendColor(...)
final void setTextureTransform(Transform3D transform)
```



Texture Coordinate

- if texture coordinates not explicit
- automatic generation
  - based upon distance from planes
  - object linear: texture coordinates move with object
  - eye linear: texture coordinates fixed to world
  - sphere map: for reflections/environment mapping

Student Notes

TexCoordGeneration capabilities:

ALLOW_ENABLE_READ | WRITE

ALLOW_FORMAT_READ

ALLOW_MODE_READ

ALLOW_PLANE_READ

TexCoordGeneration methods (partial list):

```
final void setEnable (boolean state)
final void setFormat (int format)
```

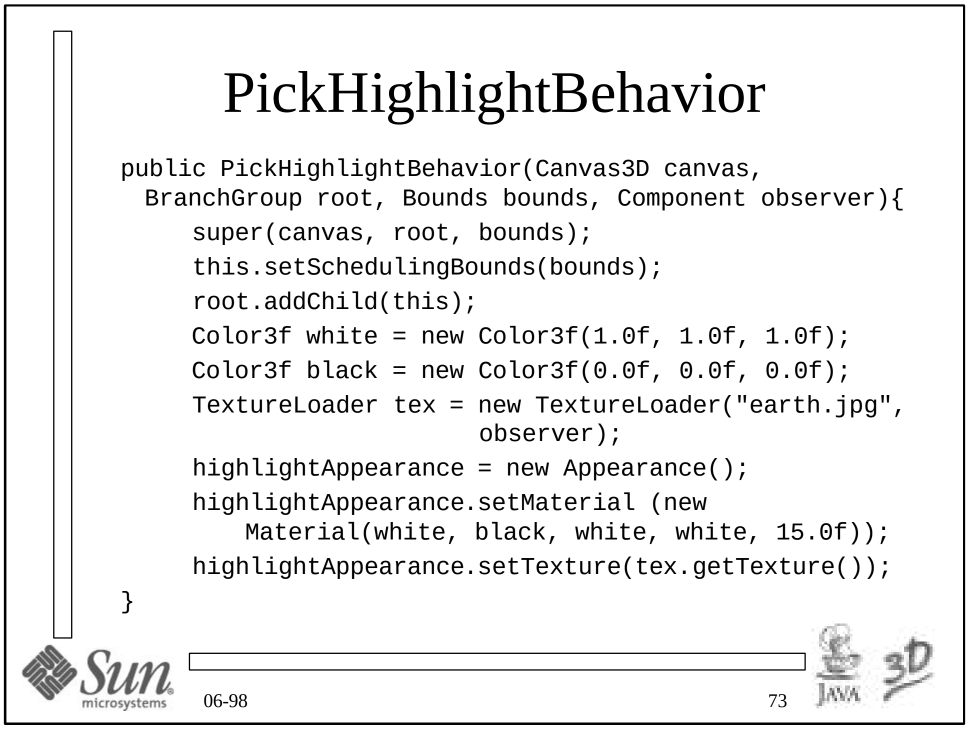
where `format` is either `TEXTURE_COORDINATE_2` or `TEXTURE_COORDINATE_3`

```
final void setGenMode(int genMode)
```

where `genMode` is one of `OBJECT_LINEAR`, `EYE_LINEAR` , or `SPHERE_MAP`

`final void setPlaneS (Vector4f plane)` or `T` or `R`

where `plane` is the plane equation used to generate the S, T, or R coordinate in `OBJECT_LINEAR` and `EYE_LINEAR` texture generation modes

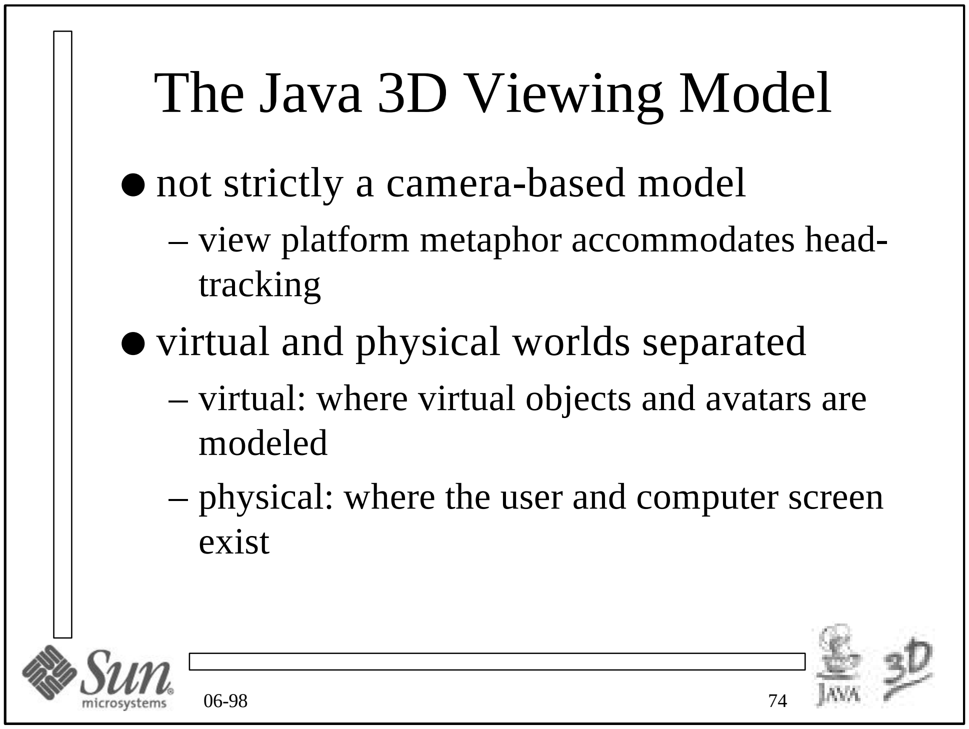


PickHighlightBehavior

```
public PickHighlightBehavior(Canvas3D canvas,
  BranchGroup root, Bounds bounds, Component observer){
     super(canvas, root, bounds);
     this.setSchedulingBounds(bounds);
     root.addChild(this);
     Color3f white = new Color3f(1.0f, 1.0f, 1.0f);
     Color3f black = new Color3f(0.0f, 0.0f, 0.0f);
     TextureLoader tex = new TextureLoader("earth.jpg",
                         observer);
     highlightAppearance = new Appearance();
     highlightAppearance.setMaterial (new
         Material(white, black, white, white, 15.0f));
     highlightAppearance.setTexture(tex.getTexture());
}
```

Student Notes

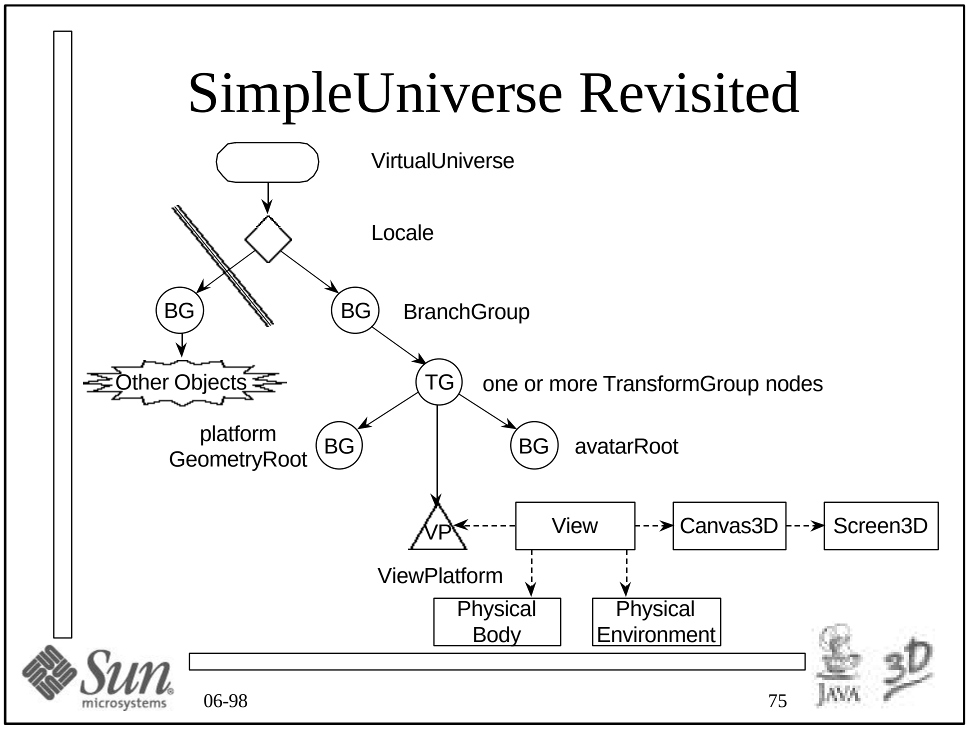
There is no error checking here, in case TextureLoader fails. However, should it fail, null is returned. Appearance.setTexture(null) doesn't cause an exception; it just disables texture mapping.



The Java 3D Viewing Model

- not strictly a camera-based model
  - view platform metaphor accommodates head-tracking
- virtual and physical worlds separated
  - virtual: where virtual objects and avatars are modeled
  - physical: where the user and computer screen exist

Student Notes



SimpleUniverse Revisited

VirtualUniverse

Locale

BG

BG

BranchGroup

Other Objects

TG

one or more TransformGroup nodes

platform GeometryRoot

BG

BG

avatarRoot

VP

View

Canvas3D

Screen3D

ViewPlatform

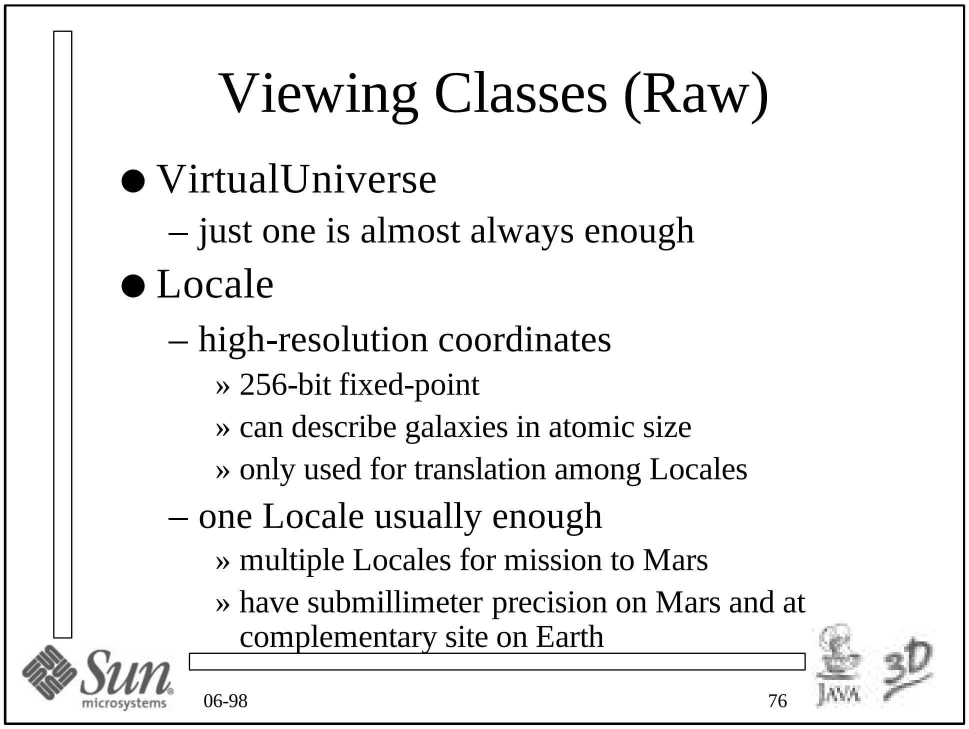
Physical Body

Physical Environment

Student Notes

The SimpleUniverse utility creates all the view-related objects. The TransformGroup is actually a specially derived class, MultiTransformGroup, which can support a chain of one or more TransformGroup nodes.

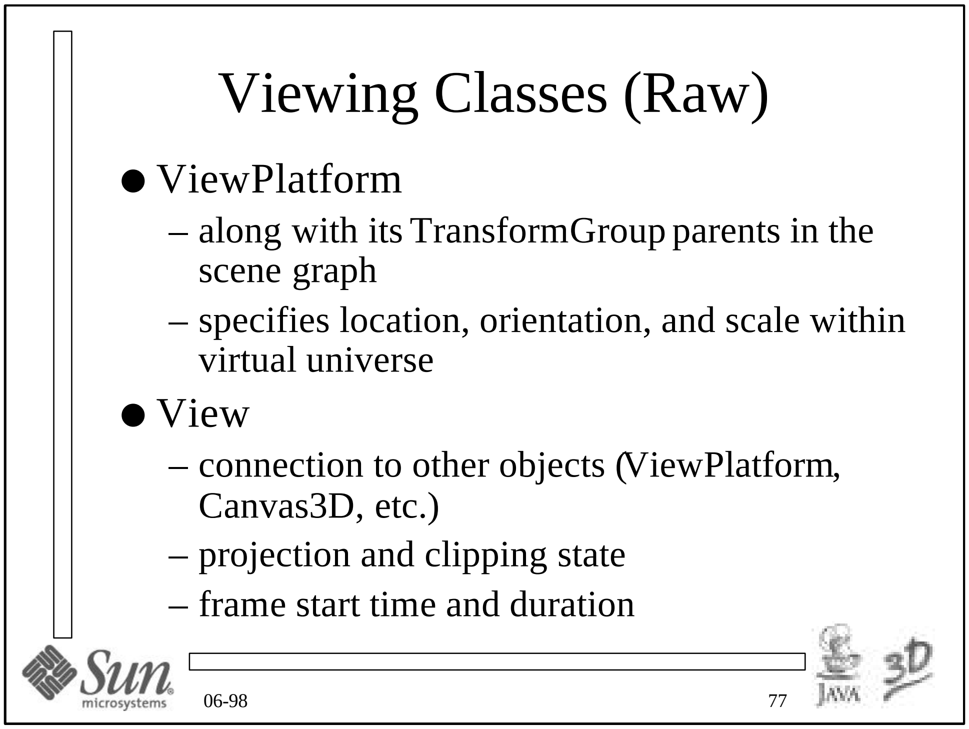
SimpleUniverse creates the entire scene graph shown above, except for the two objects on the left side. SimpleUniverse does have a method to allow the programmer to attach the BranchGroup (and any attached subgraph) to the Locale.



Viewing Classes (Raw)

- VirtualUniverse
  - just one is almost always enough
- Locale
  - high-resolution coordinates
    - 256-bit fixed-point
    - can describe galaxies in atomic size
    - only used for translation among Locales
  - one Locale usually enough
    - multiple Locales for mission to Mars
    - have submillimeter precision on Mars and at complementary site on Earth

Student Notes



Viewing Classes (Raw)

- ViewPlatform
  - along with its TransformGroup parents in the scene graph
  - specifies location, orientation, and scale within virtual universe
- View
  - connection to other objects (ViewPlatform, Canvas3D, etc.)
  - projection and clipping state
  - frame start time and duration

Student Notes

Multiple View objects are supported. Each View object controls its own set of canvases.

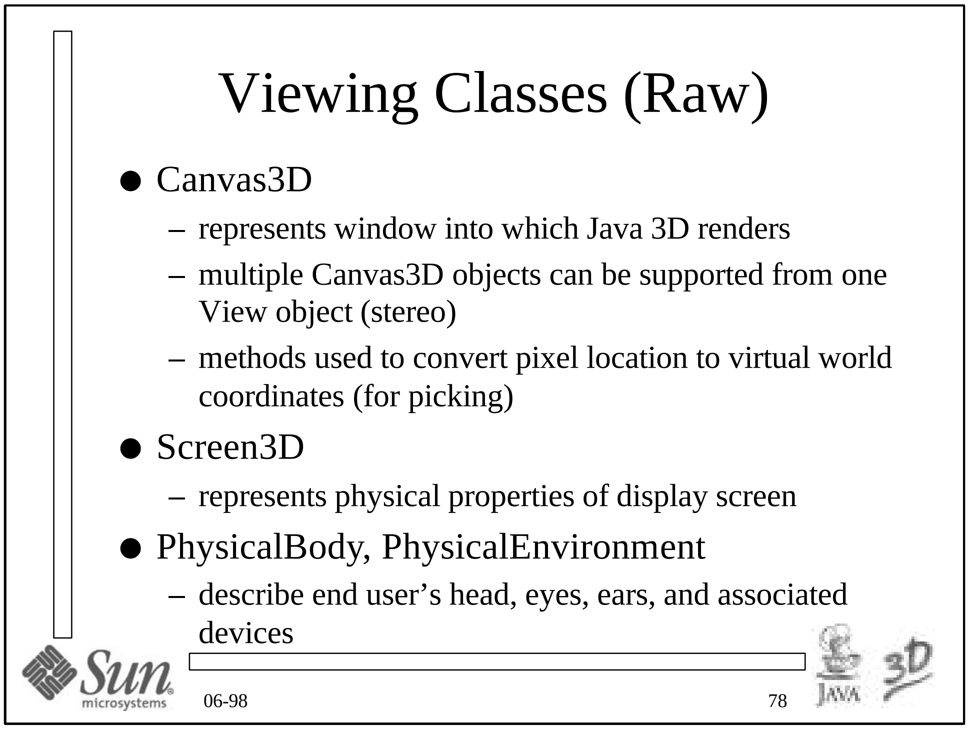
ViewPlatform methods (partial list):

```
final void setViewAttachPolicy (int policy)
```

where `policy` is one of: `NOMINAL_HEAD` (default; origin at head), `NOMINAL_FEET`, `NOMINAL_SCREEN` (origin at screen; head offset from origin)

View methods (partial list):

```
final void setPhysicalBody (PhysicalBody physicalBody)
final void setPhysicalEnvironment (PhysicalEnvironment
physicalEnvironment)
final void attachViewPlatform (ViewPlatform vp)
final void setCanvas3D (Canvas3D canvas3D, int index)
final void setProjectionPolicy (int policy)
long getCurrentFrameStartTime()
long getLastFrameDuration()
long getFrameNumber()
```



Viewing Classes (Raw)

- Canvas3D
  - represents window into which Java 3D renders
  - multiple Canvas3D objects can be supported from one View object (stereo)
  - methods used to convert pixel location to virtual world coordinates (for picking)
- Screen3D
  - represents physical properties of display screen
- PhysicalBody, PhysicalEnvironment
  - describe end user’s head, eyes, ears, and associated devices

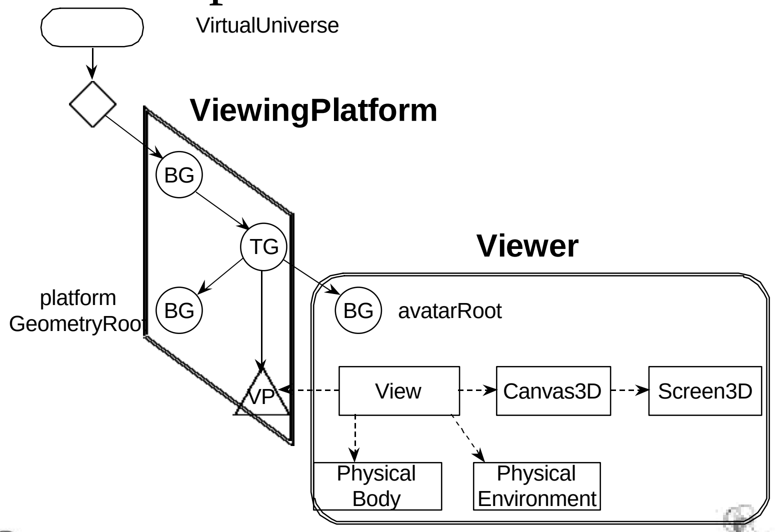
Student Notes

SimpleUniverse

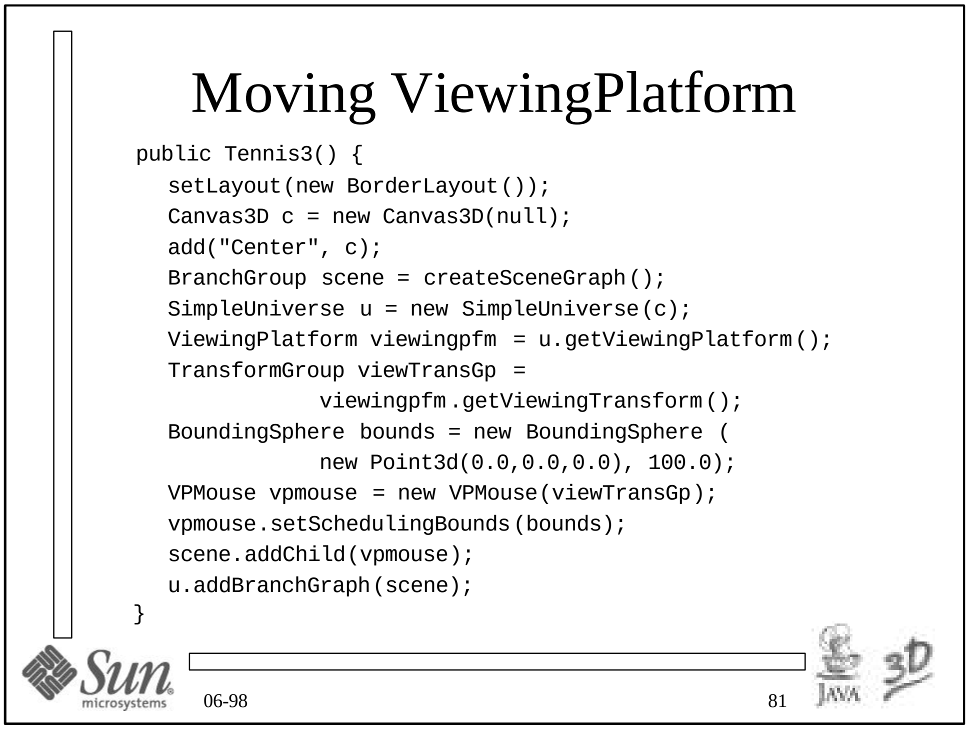
- Convenience Utility
- describes new Convenience classes
  - – Viewer
    - » virtual & physical “presence”
  - – ViewingPlatform
    - » PlatformGeometry
      - could be dashboard of car or airplane cockpit
    - » ViewerAvatar
      - could represent user’s hands

Student Notes

SimpleUniverse Classes



Student Notes

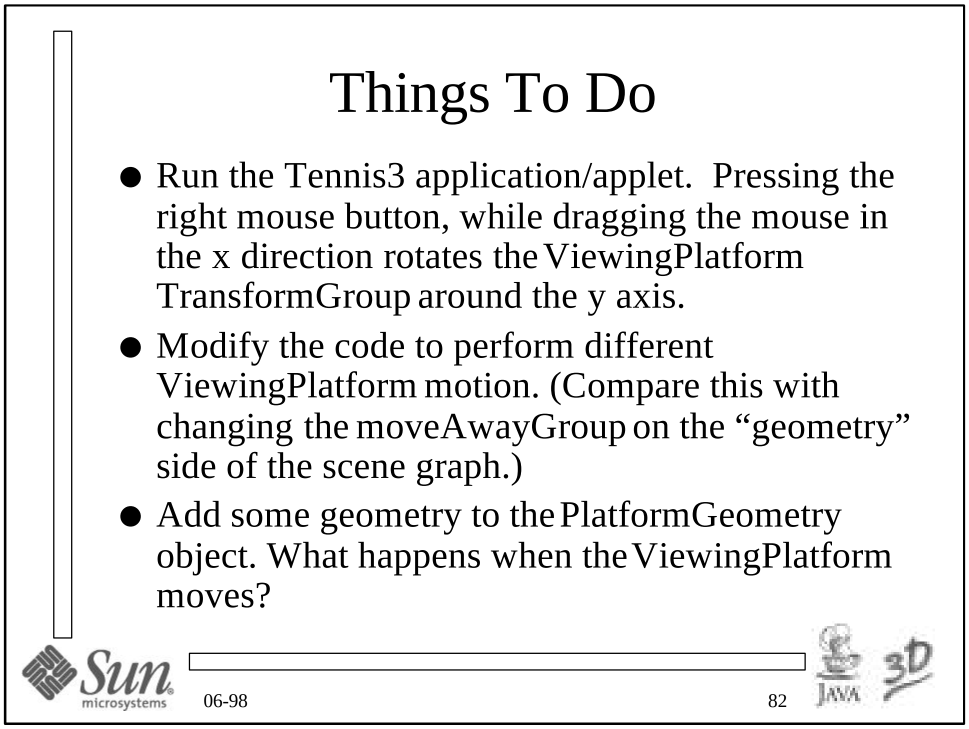


Moving ViewingPlatform

```
public Tennis3() {
   setLayout(new BorderLayout());
   Canvas3D c = new Canvas3D(null);
   add("Center", c);
   BranchGroup scene = createSceneGraph();
   SimpleUniverse u = new SimpleUniverse(c);
   ViewingPlatform viewingpfm = u.getViewingPlatform();
   TransformGroup viewTransGp =
               viewingpfm.getViewingTransform();
   BoundingSphere bounds = new BoundingSphere (
               new Point3d(0.0,0.0,0.0), 100.0);
   VPMouse vpmouse = new VPMouse(viewTransGp);
   vpmouse.setSchedulingBounds(bounds);
   scene.addChild(vpmouse);
   u.addBranchGraph(scene);
}
```

Student Notes

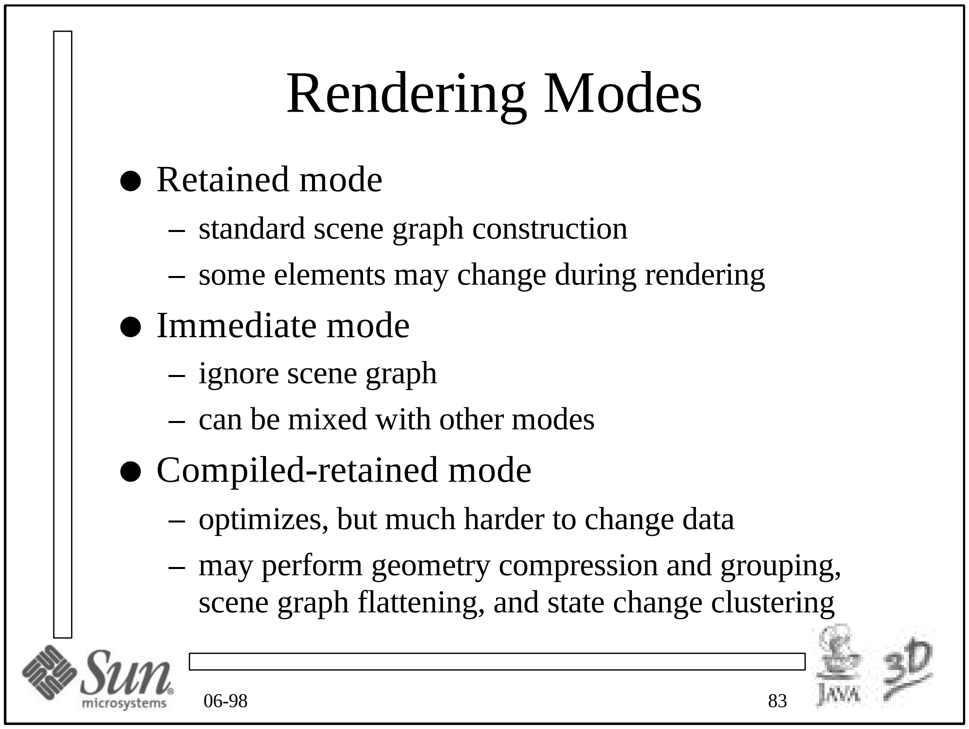
This introduces a VPMouse object, which is a behavior that monitors an AWT event (right mouse drag in x direction). Since SimpleUniverse branch graph is already compiled (and cannot be added to), the VPmouse behavior is added to the content branch graph.



Things To Do

- Run the Tennis3 application/applet. Pressing the right mouse button, while dragging the mouse in the x direction rotates the ViewingPlatform TransformGroup around the y axis.
- Modify the code to perform different ViewingPlatform motion. (Compare this with changing the moveAwayGroup on the “geometry” side of the scene graph.)
- Add some geometry to the PlatformGeometry object. What happens when the ViewingPlatform moves?

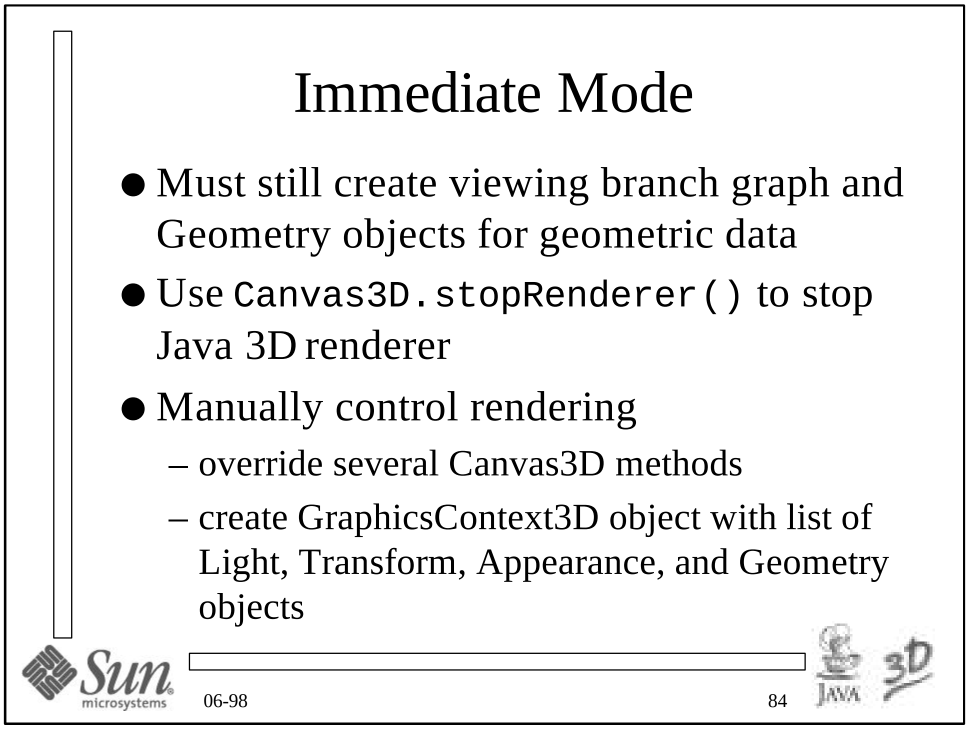
Student Notes



Rendering Modes

- Retained mode
  - standard scene graph construction
  - some elements may change during rendering
- Immediate mode
  - ignore scene graph
  - can be mixed with other modes
- Compiled-retained mode
  - optimizes, but much harder to change data
  - may perform geometry compression and grouping, scene graph flattening, and state change clustering

Student Notes



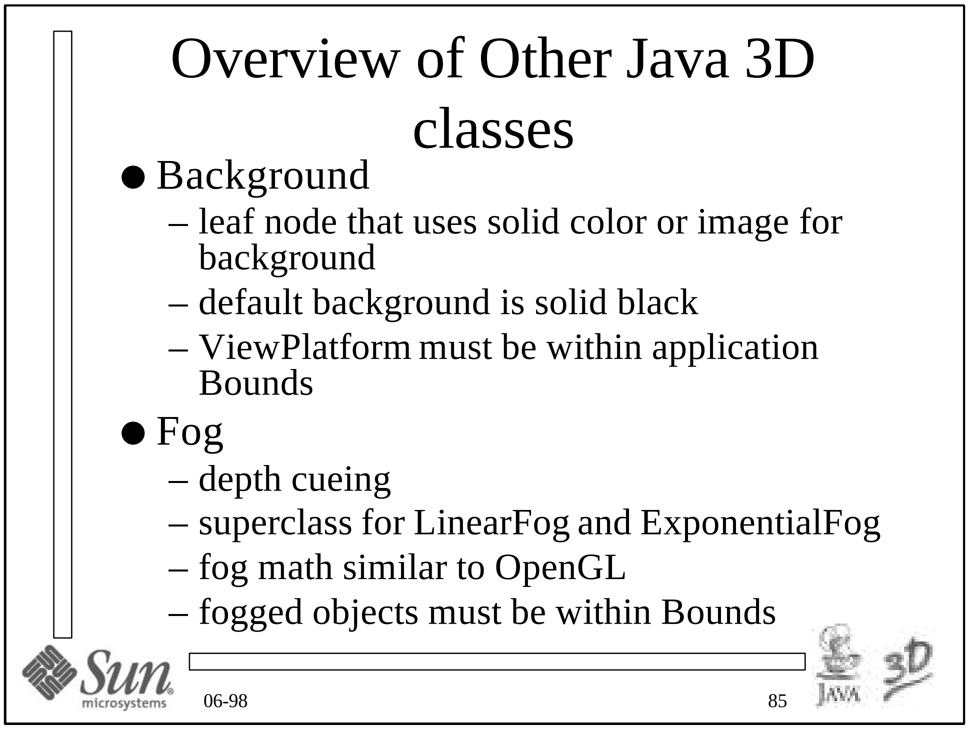
Immediate Mode

- Must still create viewing branch graph and Geometry objects for geometric data
- Use `Canvas3D.stopRenderer()` to stop Java 3D renderer
- Manually control rendering
  - – override several Canvas3D methods
  - – create GraphicsContext3D object with list of Light, Transform, Appearance, and Geometry objects

Student Notes

The basic Java 3D rendering loop is:

clear canvas
call overridden `Canvas3D.preRender()`
set view (in viewing branch graph)
render opaque scene graph objects
call overridden `Canvas3D.renderField(FIELD_ALL)`
render transparent scene graph objects
call overridden `Canvas3D.postRender()`
synchronize and swap buffers
call overridden `Canvas3D.postSwap()`

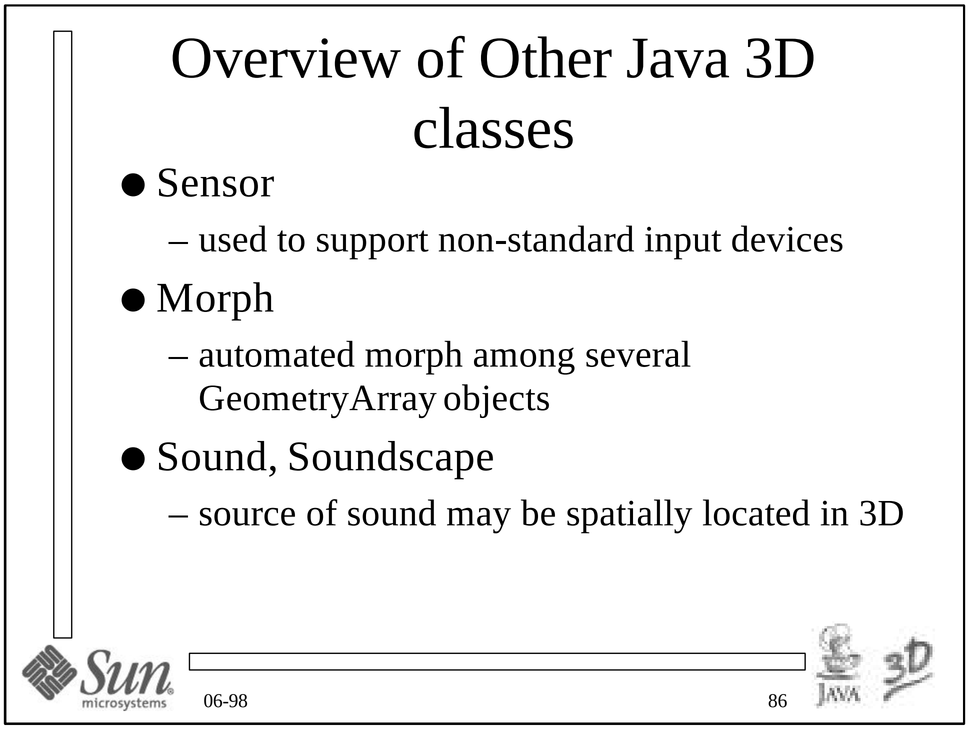


Overview of Other Java 3D classes

- Background
  - – leaf node that uses solid color or image for background
  - – default background is solid black
  - – ViewPlatform must be within application Bounds
- Fog
  - – depth cueing
  - – superclass for LinearFog and ExponentialFog
  - – fog math similar to OpenGL
  - – fogged objects must be within Bounds

Student Notes

Several Background or Fog nodes may be active, but the “closest” (to ViewPlatform or object) is used.



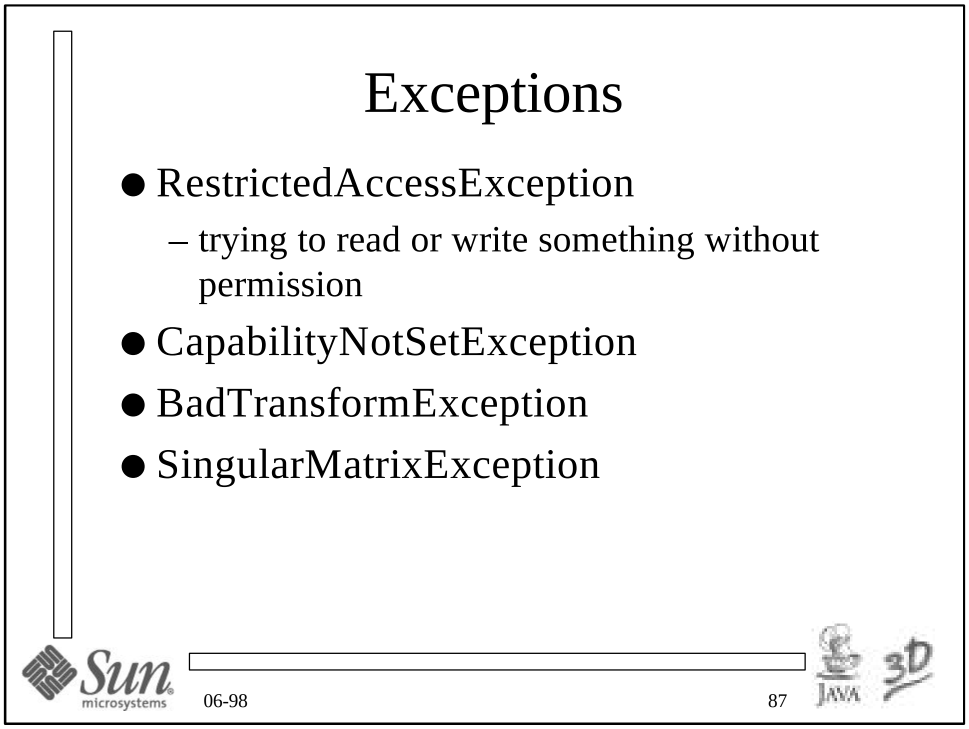
Overview of Other Java 3D classes

- Sensor
  - – used to support non-standard input devices
- Morph
  - – automated morph among several GeometryArray objects
- Sound, Soundscape
  - – source of sound may be spatially located in 3D

Student Notes

A Morph object consists of:

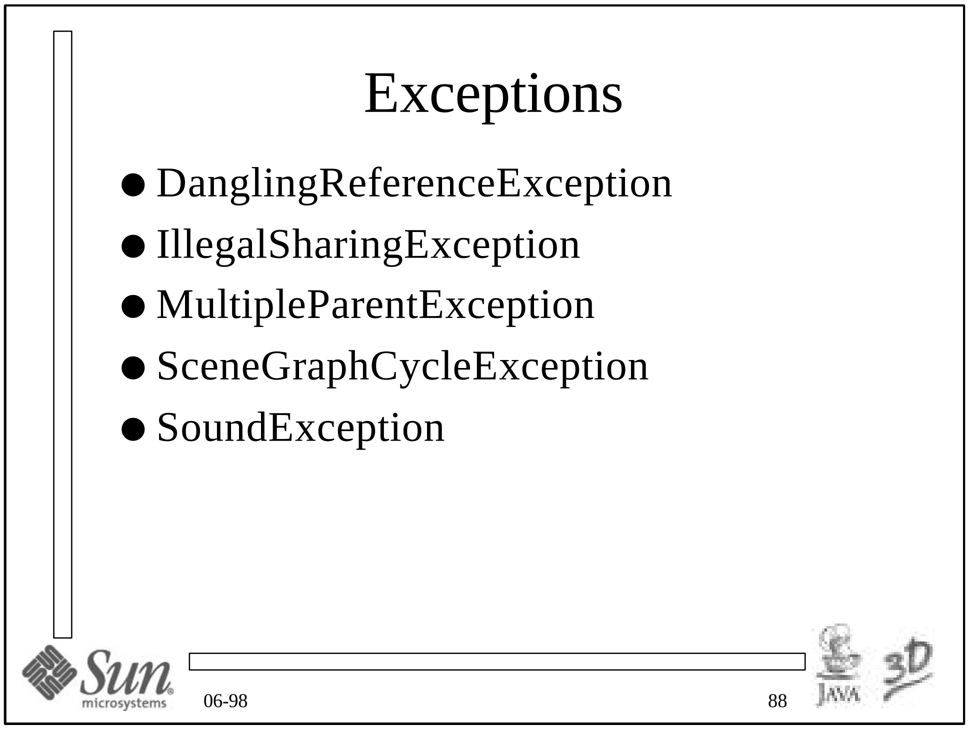
- a single Appearance object
- an array of GeometryArray objects
- an array of corresponding weights



Exceptions

- RestrictedAccessException
  - trying to read or write something without permission
- CapabilityNotSetException
- BadTransformException
- SingularMatrixException

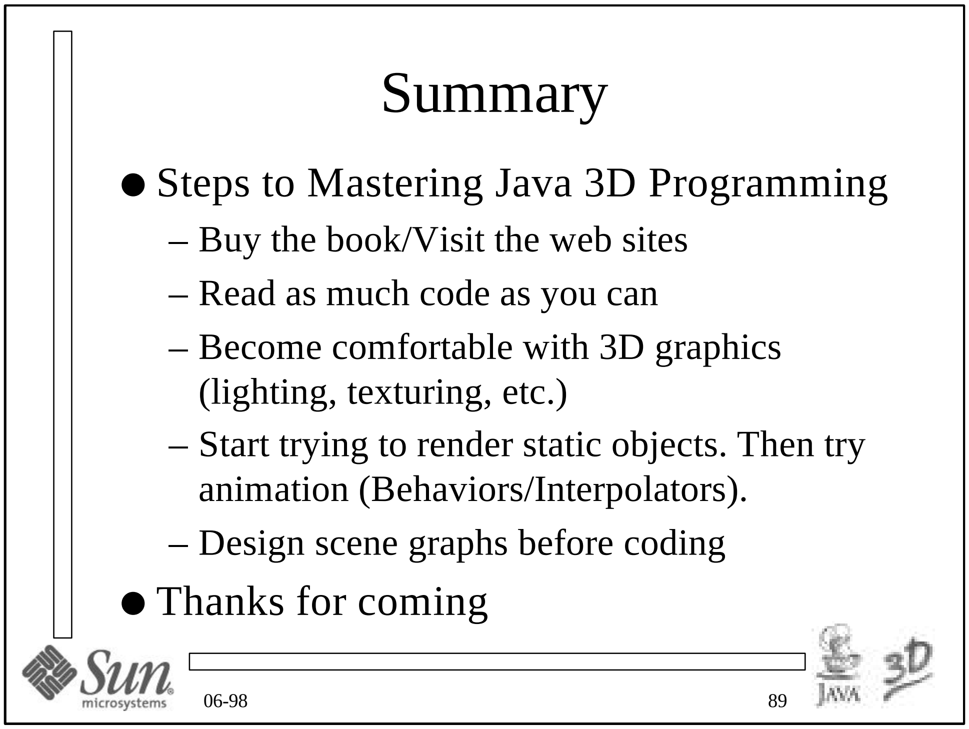
Student Notes



Exceptions

- DanglingReferenceException
- IllegalSharingException
- MultipleParentException
- SceneGraphCycleException
- SoundException

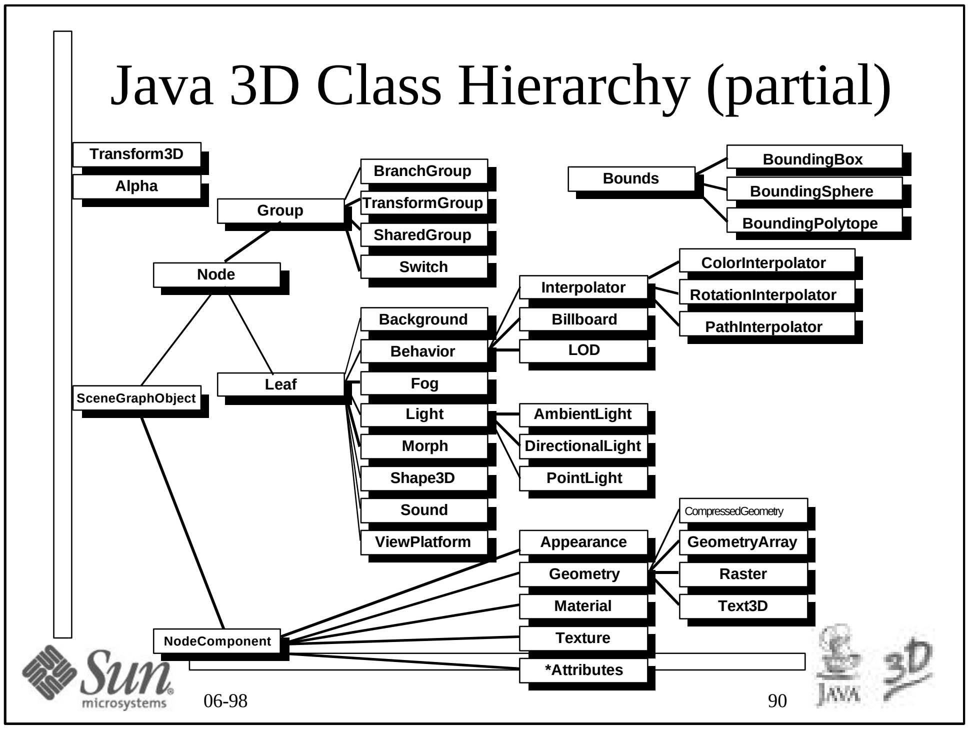
Student Notes



Summary

- Steps to Mastering Java 3D Programming
  - Buy the book/Visit the web sites
  - Read as much code as you can
  - Become comfortable with 3D graphics (lighting, texturing, etc.)
  - Start trying to render static objects. Then try animation (Behaviors/Interpolators).
  - Design scene graphs before coding
- Thanks for coming

Student Notes



Java 3D Class Hierarchy (partial)

Transform3D

Alpha

SceneGraphObject

Node

Group

BranchGroup

TransformGroup

SharedGroup

Switch

Leaf

Background

Behavior

Interpolator

ColorInterpolator

RotationInterpolator

PathInterpolator

Billboard

LOD

Fog

Light

AmbientLight

DirectionalLight

PointLight

Morph

Shape3D

Sound

ViewPlatform

Bounds

BoundingBox

BoundingSphere

BoundingPolytope

NodeComponent

Appearance

Geometry

CompressedGeometry

GeometryArray

Raster

Text3D

Material

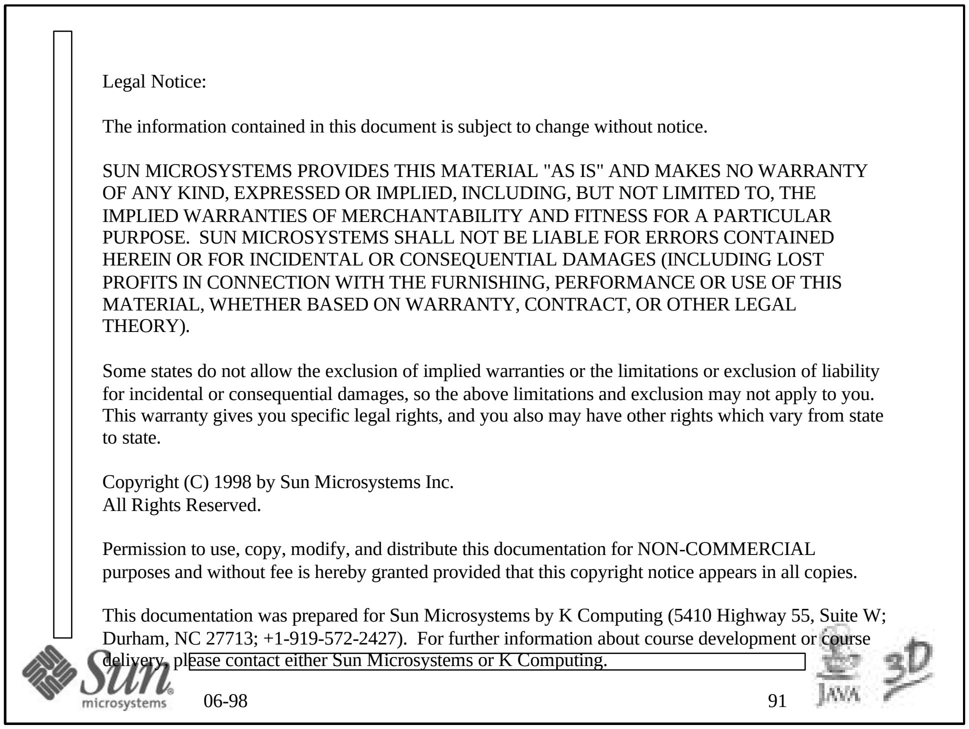
Texture

*Attributes

06-98

90

Student Notes



Legal Notice:

The information contained in this document is subject to change without notice.

SUN MICROSYSTEMS PROVIDES THIS MATERIAL "AS IS" AND MAKES NO WARRANTY OF ANY KIND, EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. SUN MICROSYSTEMS SHALL NOT BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES (INCLUDING LOST PROFITS IN CONNECTION WITH THE FURNISHING, PERFORMANCE OR USE OF THIS MATERIAL, WHETHER BASED ON WARRANTY, CONTRACT, OR OTHER LEGAL THEORY).

Some states do not allow the exclusion of implied warranties or the limitations or exclusion of liability for incidental or consequential damages, so the above limitations and exclusion may not apply to you. This warranty gives you specific legal rights, and you also may have other rights which vary from state to state.

This documentation was prepared for Sun Microsystems by K Computing (5410 Highway 55, Suite W; Durham, NC 27713; +1-919-572-2427). For further information about course development or course delivery, please contact either Sun Microsystems or K Computing.

Student Notes