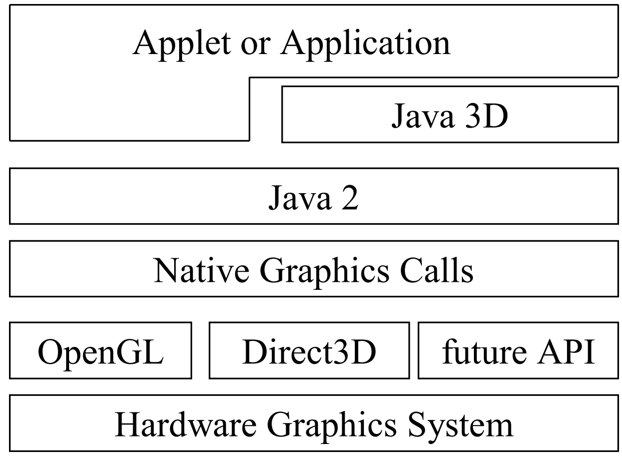
Java 3D API

High level graphics programming interface

Scene graph based graphics universe

Java threads for parallel rendering

100+ classes in Java 3D core library javax.media.j3d package

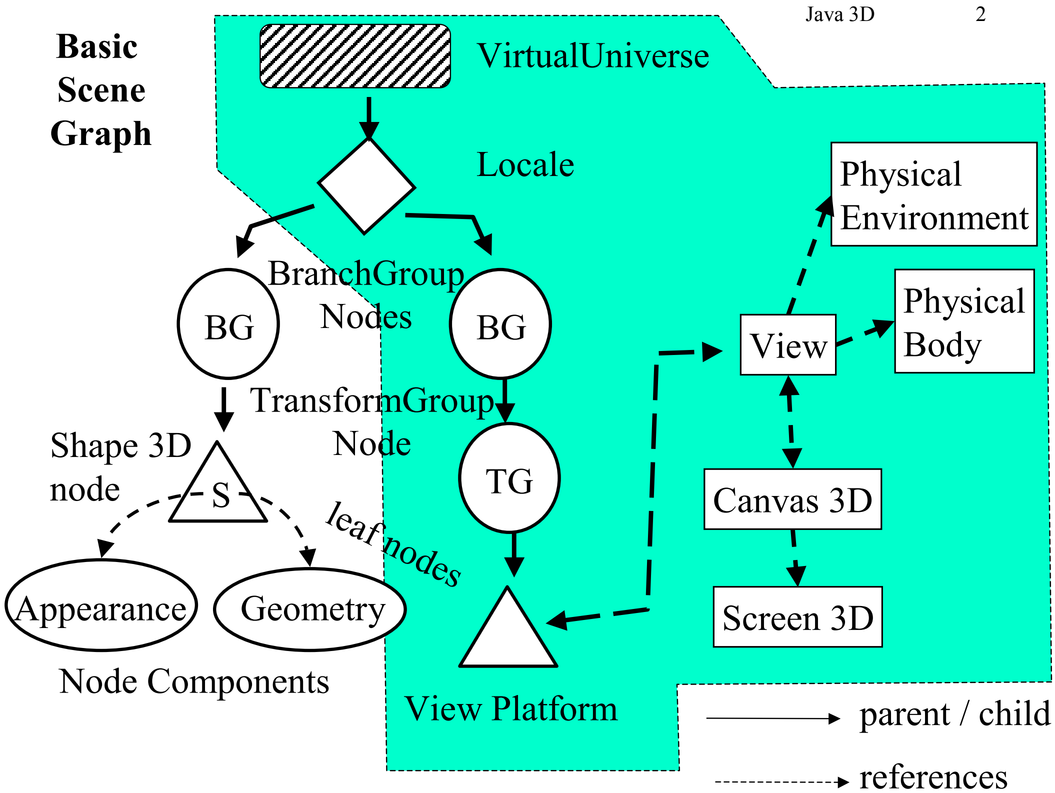


Java 3D utility package com.sun.j3d.utils

Use other Java libraries (Swing, AWT) and capabilities (url class for networking, multimedia classes etc.)

Notes adapted from Sun's j3d_tutorial.pdf (in vrlab or sunsoft.com)

Basic Scene Graph



The Virtual Universe and View Platform

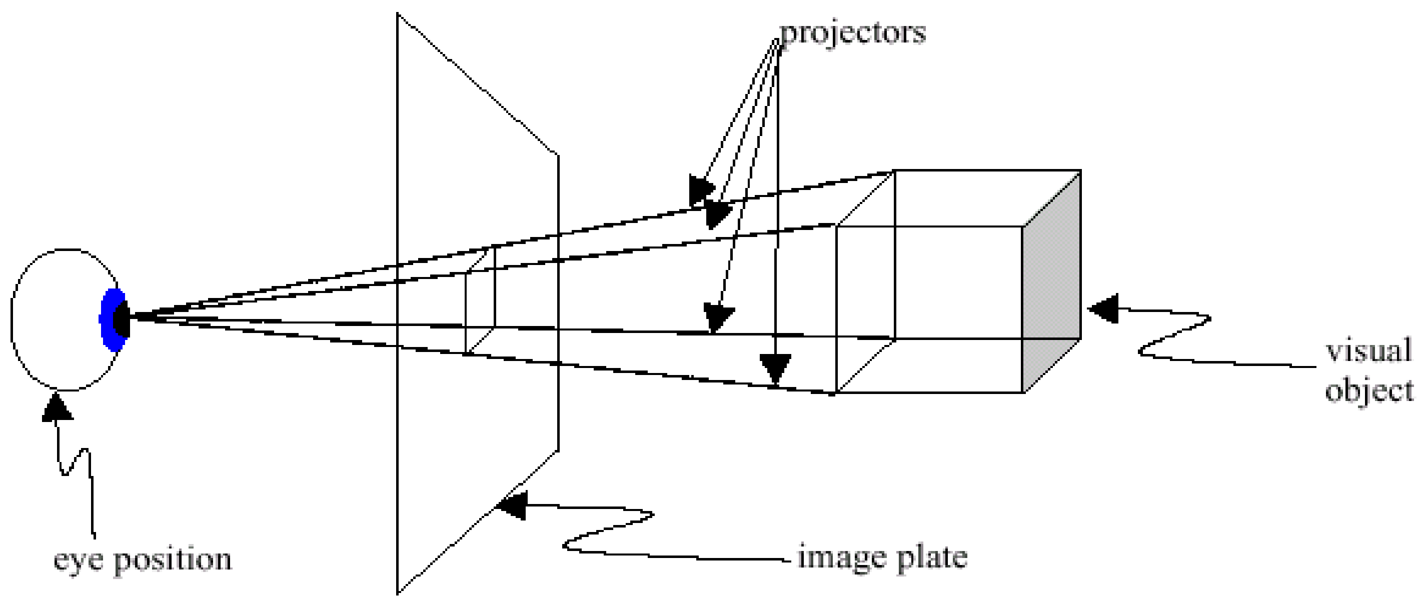


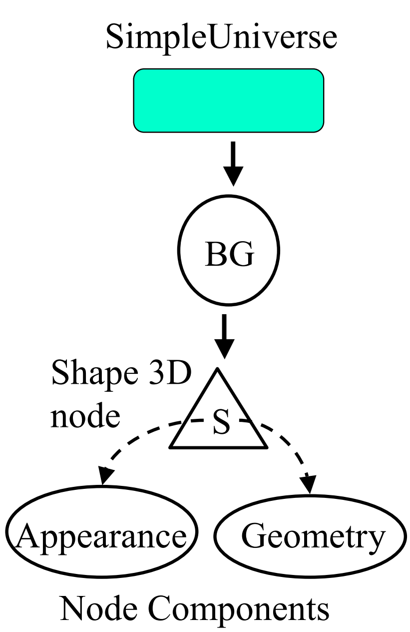
Figure 1-9 Conceptual Drawing of Image Plate and Eye Position in a Virtual Universe.

Scene graph is a DAG -- there is one path from the locale to a leaf nodepath describes how the leaf is rendered.

Writing a Java 3D Program

1 create a Canvas3D
2 create a VirtualUniverse
3 create a Locale object, attach to VirtualUniverse
4 construct a view branch graph
   a create View object
   b create ViewPlatform
   c create a PhysicalBody
   d create a PhysicalEnvironment
   e attach ViewPlatform, PhysicalBody, PhysicalEnvironment, Canvas3D to View
5 construct content branch graph
6 compile branch graph
7 insert subgraphs into Locale

SimpleUniverse -- convenience, beginning
ignore view branch graph.
no multiple views of universe



Writing a SimpleUniverse program

1 create a Canvas3D
2 create a SimpleUniverse that references Canvas3D
   a customize SimpleUniverse
3 construct content branch
4 compile content graph
5 insert content branch into Locale of SimpleUniverse

SimpleUniverse methods

```
SimpleUniverse()
SimpleUniverse(Canvas3D canvas3D)   // references canvas3D
void addBranchGraph(BranchGroup gb) // add content to Locale
```

BranchGroup methods

```
void compile() // compiles branch group facilitates rendering
```

ViewingPlatform methods

```
ViewingPlatform getViewPlatform()  // retrieve viewplatform
void setNorminalViewingTransform() // move back to see world
```

Inserting a branch graph into a Locate makes it **live** and it will be rendered.

All modifications to branch graph should be done before it becomes live.

Compiling allows Java3D to optimize branch graph once rather than every render loop cycle

Rend loop begins when a branch group with an instance of View becomes live.

```
while (true) {
    process input
    if (request to exit) break render loop
        perform behaviors
        traverse scene graph and render visual objects
    }
cleanup and exit
```

Example world: adapted from Sun's j3d_tutorial.pdf (on vrlab systems)

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

```
public class HelloJava extends Applet {
   public HelloJava() {
      setLayout (new BorderLayout());
      Canvas3D canvas3D = new Canvas3D(null);
      add("Center", canvas3D);
      BranchGroup scene = createSceneGraph();
      scene.compile();
      SimpleUniverse sU = new SimpleUniverse(Canvas3D);
      // move viewplatform back
      sU.getViewingPlatform().setNominalViewingTransform();
      sU.addBranchGraph(scene);
      }
   public BranchGroup createSceneGraph() {
      BranchGroup root = new BranchGroup();
      // ColorCube convenience shape, different colored sides
      root.addChild(new ColorCube(0.4));
      return root;
      };
   // run as applet or application
   public static void main (String[] args) {
      Frame frame = new MainFrame(new HelloJava(), 256, 256);
      }
   }
```

Adding Transformation

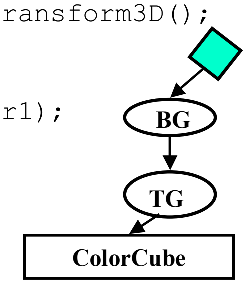
Transform3D object is used to specify the transformation of a TransformGroup object.

```
   Transform3D( ) // identity matrix
   TransformGroup(Transform3D t3d)      // construct with  t3d
   setTransform(Transform3D t3d)        // set to t3d
```

numerous matrix, vector, point3D classes in javax.vecmath.*

i.e.: rotX(double radian), set(Vector3f translate), Math.PI

```
public BranchGroup createSceneGraph() {
   BranchGroup objRoot = new BranchGroup();
   Transform3D r1 = new Transform3D(), r12 = new Transform3D();
   r1.rotX(Math.PI/4.0d);  r2.rotY(Math.PI/5.0d);
   r1.mul(r2);
   TransformGroup objRotated = new TransformGroup(r1);
   objRotated.addChild(new ColorCube(0.4));
   objRoot.addChild(objRotated)
   return objRoot:
   }
```



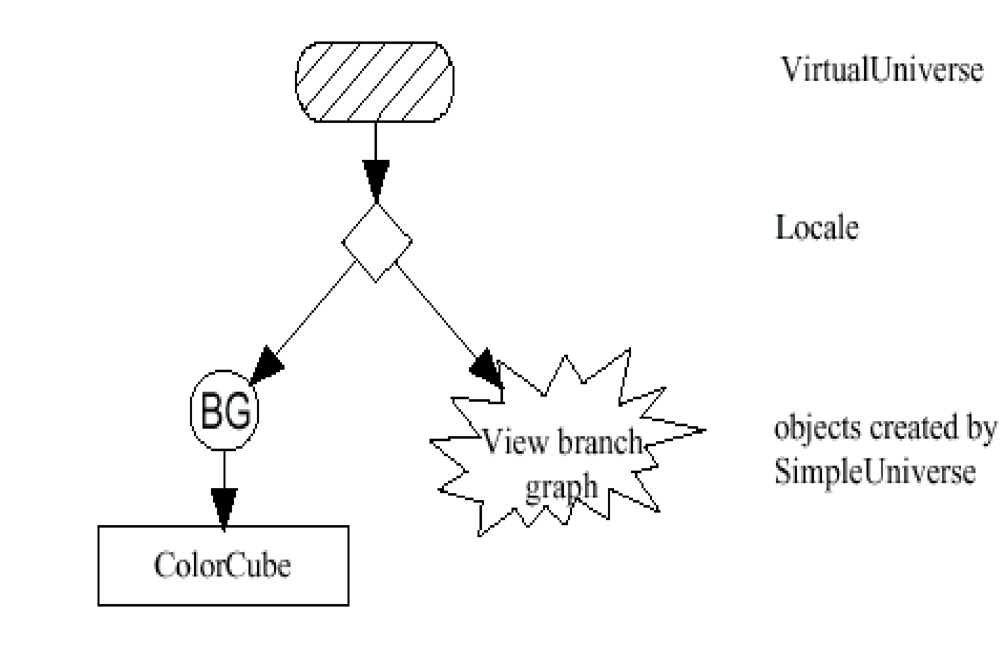


Figure 1-11 Scene Graph for HelloJava3Da Example

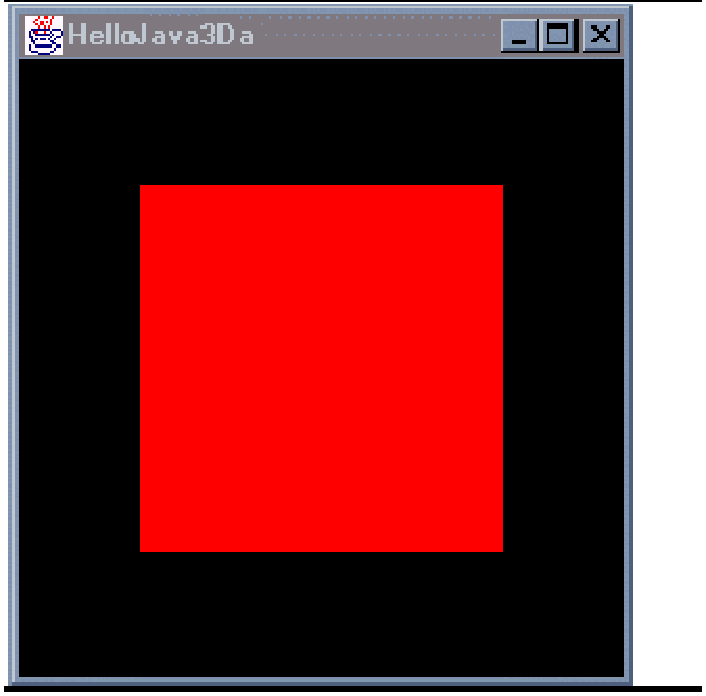


Figure 1-12 Image Produced by HelloJava3Da

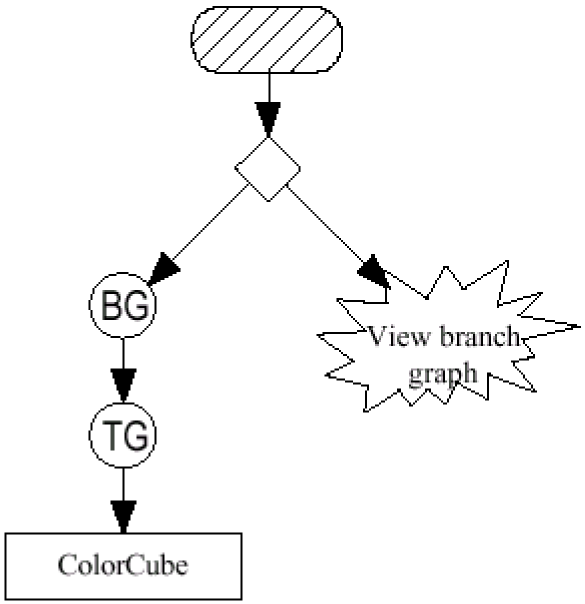


Figure 1-14 Scene Graph for HelloJava3Db Example

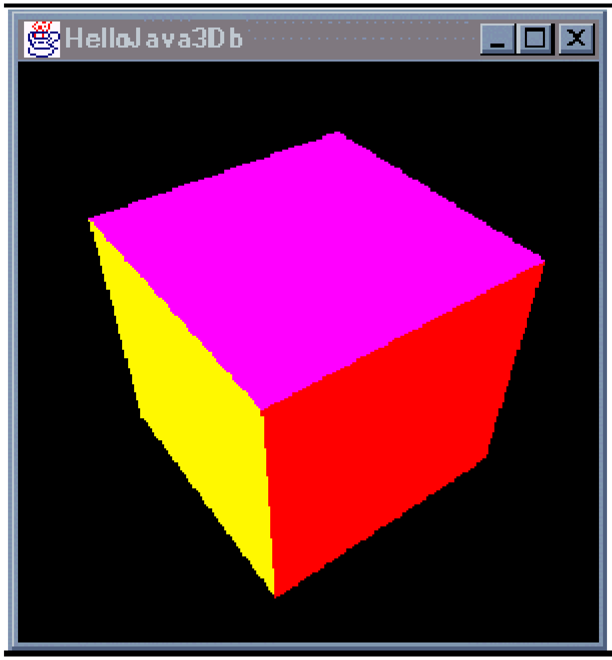


Figure 1-15 Image of the Rotated ColorCube Rendered by HelloJava3Db

The basic concept of Behavior

- Behavior is a class for specifying animations of or interaction with visual objects.
- The distinction between animation and interaction is whether the behavior is activated in response to the passing of time or in response to user activities, respectively.
- To specify a behavior for a visual object, the programmer creates the objects that specify the behavior, adds the visual object to the scene graph, and making the appropriate references among scene graph objects and the behavior objects

Scheduling region and activation volume

- In a virtual universe with many behaviors, a significant amount of computing power could be required just for computing the behaviors. Since both the renderer and behaviors use the same processor(s), it is possible the computational power requirement for behaviors could degrade rendering performance.
- Java 3D allows the programmer to manage this problem by specifying a spatial boundary for a behavior to take place. This boundary is called a ***scheduling region***. A behavior is not active unless the ViewPlatform's ***activation volume*** intersects a Behavior object's scheduling region. In other words, if there is no one in the forest to see the tree falling, it does not fall. The scheduling region feature makes Java 3D more efficient in handling a virtual universe with many behaviors.

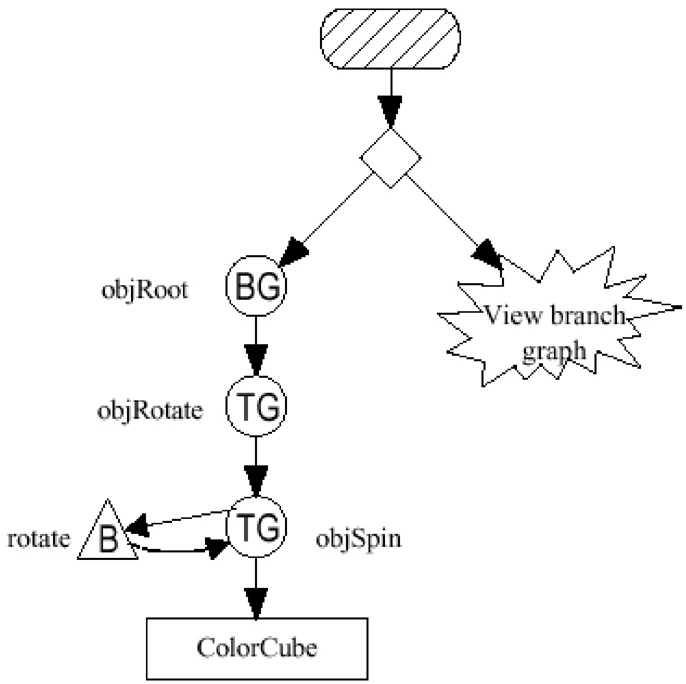


Figure 1-20 Scene Graph for HelloJava3Dd Example

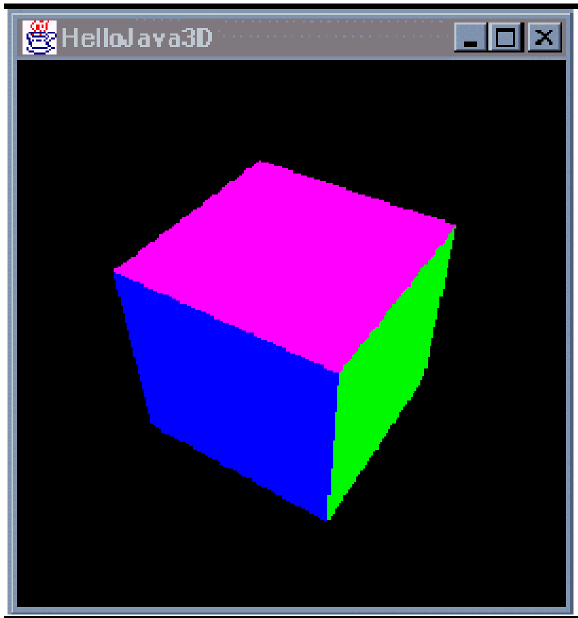


Figure 1-21 An Image of the ColorCube in Rotation as Rendered by HelloJava3Dd

Branch graphs can't be changed once live (or compiled) unless their capabilities are set for modification (prior to becoming live).

```
void setCapability(int bit)
   ALLOW_TRANSFORM_READ    can read values
   ALLOW_TRANSFORM_WRITE   can write values
```

Behavior class specifies animations or interactions with visual objects.
- animations are activated by passing of time
- interactions are activated by user activities

Many behaviors can affect performance.

Behaviors can be limited by a proximity test.
- Behaviors have scheduling regions (bounding boxes or spheres)

Behaviors w/ scheduling regions are active only when they intersect with ViewPlatform's activation volume

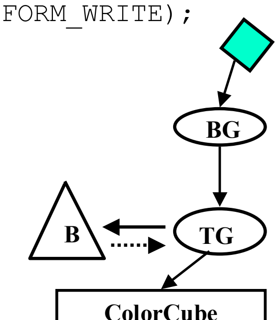
Interpolator objects can manipulate behaviors in scene graph based on a time function.

Interpolator objects can manipulate behaviors in scene graph based on a time function.

Alpha class generates values 0 to 1 depending on parameters

```
Alpha(); // continuous loop 1 second period
Alpha(int loopCount, long periodDuration); // in milliseconds
   loopCount == -1 repeats
```

```
public BranchGroup createSceneGraph() {
   BranchGroup objRoot = new BranchGroup();
   TransformGroup spin = new TransformGroup();
   spin.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
   objRoot.addChild(spin);
   spin.addChild( new ColorCube, 0.4));
   Alpha rotation = new Alpha(-1, 4000);
   RotationInterpolator rotator =
      new RotationInterpolator(rotation, spin);
   BoundingSphere bounds = new BoundingSphere();
   rotator.setSchedulingBounds(bounds);
   spin.addChild(rotator);
   return objRoot
   }
```

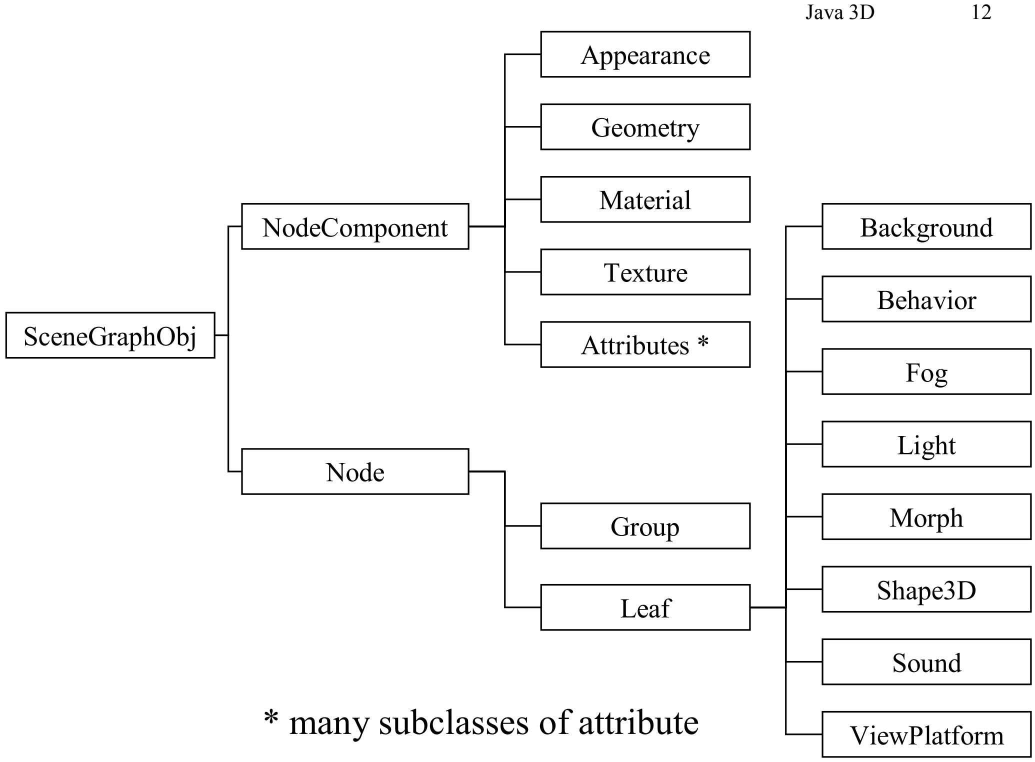


Geometry

Shape3D( ) // no geometry or appearance node components
Shape3D(Geometry geometry)
Shape3D(Geometry geometry, Appearance appearance)

before a Shape3D is live or compiled
  void setGeometry(Geometry geom)
  void setAppearance(Appearance appear)

after live or compiled need to set capability bits to enable changes
  ALLOW_GEOMETRY_READ | WRITE
  ALLOW_APPEARANCE_READ | WRITE
  ALLOW_COLLISION_BOUNDS_READ | WRITE



* many subclasses of attribute

Application class definition “psuedocode”

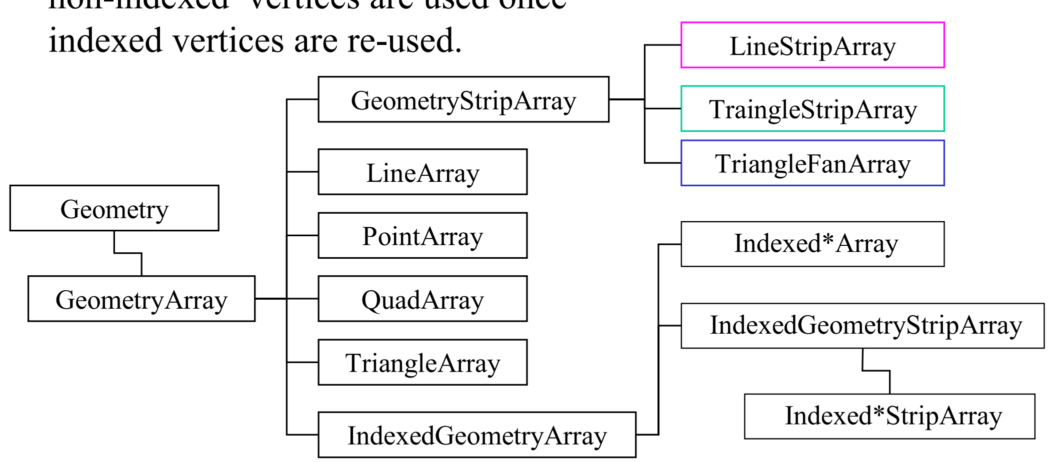
```
public class VisualObject {
   private Transform3D voTransform;
   private Shape3D voShape3d;
   private Geometry voGeometry;
   private Appearance voAppearance;

   public visualObject(Transform3D t,Geometry g, Appearance a) {
      voTransform = new Transform3D(t);
      voGeometry = new Geometry(g);
      voAppearance = new Appearance(a);
      voShape3d = new Shape3D(voGeometry, voAppearance);
      voShape3D.setCapability( ALLOW_GEOMETRY_READ |
         ALLOW_GEOMETRY_WRITE | ALLOW_APPEARANCE_READ |
         ALLOW_APPEARANCE_WRITE);
      setTransform(voTransform);
      voTransform.setCapability(ALLOW_TRANSFORM_READ |
         ALLOW_TRANSFORM_WRITE)
      voTransform.addChild(voShape3D);
      }

   // ... numerous set and get methods
}
```

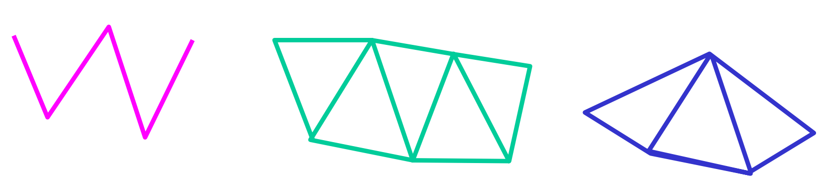

Geometry utility classes: box, cone, cylinder, sphere

Categories of Geometry are: non - indexed, indexed, and other

non-indexed vertices are used once

indexed vertices are re-used.





Appearance

Defines all rendering state attributes.
Appearance( ) constructs a default Appearance object
- color: white (1,1,1)
- texture environment mode: TEXENV_REPLACE
- texture environment color: white(1,1,1)
- depth test enable: true
- shade model: SHADE_SMOOTH
- polygon mode: POLYGON_FILL
- transparency enable: false
- transparency mode: FASTEST
- cull face: CULL_BACK
- point size: 1.0
- line width: 1.0
- line pattern: PATTERN_SOLID
- point antialiasing enabled: false
- line antialiasing enabled: false

There are set* and get* methods for all attributes
Changeable attributes must be set w/ a setCapability(flag)

For example:

```
setCapability(ALLOW_COLOR_READ | ALLOW_COLOR_WRITE)
ColoringAttribute(Color3f color, int shadeModel)
or
ColoringAttribute(float r, float g, float b, int shade)
or
setColor(Color3f color)
setShadeModel(SHADE_GOURAUD) // _FLAT _NICEST _FASTEST
```

Light Nodes

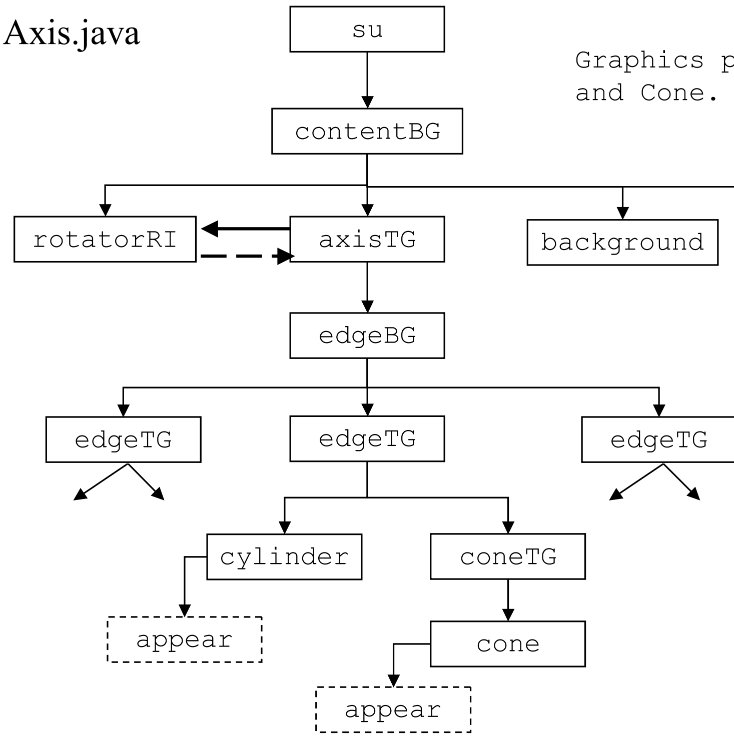
AmbientLight, default light, reflective surface, SimpleUniverse
DirectionalLight, PointLight, SpotLight

Sound Nodes

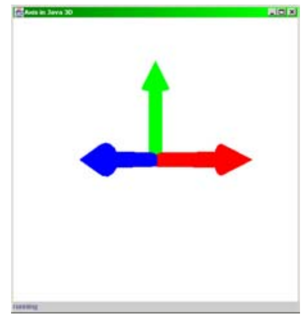
BackgroundSound (unattenuated), PointSound (radiates uniformly), ConeSound (directed), SoundScape (reverb, air...)

Example Geometry, Appearance

Axis.java




```
Graphics primitives Cylinder
and Cone.
```



Input, Behavior and Picking

Java3D has access to keyboards and mice using the Java API.

Java3D also provides access to continuous input devices, 6 DOF trackers and joysticks via an **abstract** InputDevice Interface.

InputDevice or sensors must be implemented for actual devices.

Input data from the sensor data can be read and processed.

Behavior nodes contain:

a **scheduling region** that “activates” node (intersects view platform)
an **initialization** method called when live, sets wakeup (event)
and a **processStimulus** method called when active & “woke up”

ProcessSimulus( ) receives and processes on going messages, sets new wakeup criteria, and sets the next wakeup condition before exiting

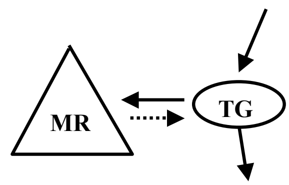
Mouse interaction

Java3D provides 4 utility classes for mouse interaction.

abstract class MouseBehavior
defines initialize, processStimuli etc for subClasses
MouseRotate, MouseTranslate, MouseZoom

MouseRotate
a Behavior to set for a TransformGroup
drag the left mouse

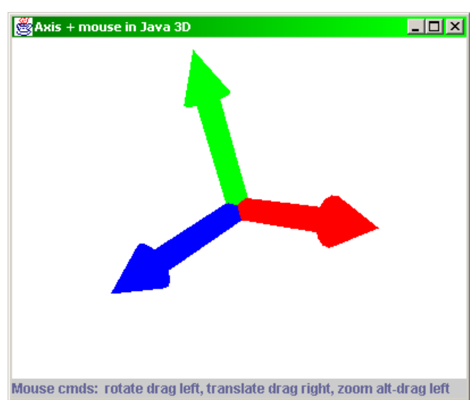
```
import com.sun.j3d.utils.behaviors.mouse.*;
...
MouseRotate behavior = new MouseRotate();
behavior.setTransformGroup(objTrans);
objTrans.addChild(behavior);
behavior.setSchedulingBounds(bounds);
...
```



MouseTranslate
drag the right mouse

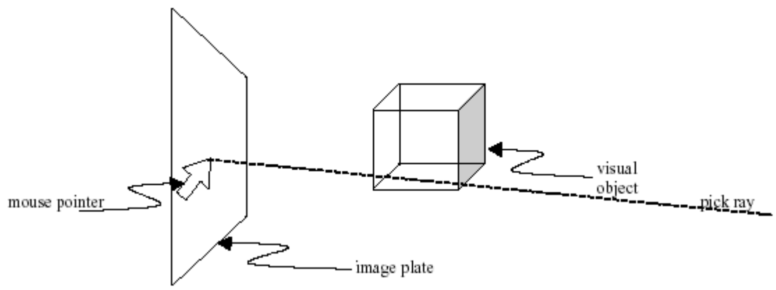
MouseZoom
alt-drag the left mouse

see AxisMouse.java example
or Sun tutorials on Interaction



Generalized picking and Pick Utility classes

Generalized picking is ray based:



Using the rotate, translate and zoom pick utilities

1. Create your scene graph.
2. Create this behavior with root and canvas

```
PickRotateBehavior behavior =
   new  PickRotateBehavior(canvas, root, bounds);
root.addChild(behavior);
```


A picking ray is projected from the screen along Z.
The sceneGraphPath of objects (closest or all) intersecting ray is created. The object is obtained by searching the sceneGraphPath.

Example mouse picking behavior -- see also Sun's MousePickApp.java in java3D tutorials

```
WakeupCriterion[] mouseEvents;
WakeupOr mouseCriterion;
Positions positions;
PickRay pickRay = new PickRay();
SceneGraphPath sceneGraphPath[];
...
public void initialize() {
   ...
   mouseEvents = new WakeupCriterion[2];
   mouseEvents[0] = new
      WakeupOnAWTEvent(MouseEvent.MOUSE_DRAGGED);
   mouseEvents[1] = new
      WakeupOnAWTEvent(MouseEvent.MOUSE_PRESSED);
   mouseCriterion = new WakeupOr(mouseEvents); // any condition
   wakeupOn (mouseCriterion);
   }
```

```
public void processStimulus (Enumeration criteria) {
   WakeupCriterion wakeup;
   AWTEvent[] event;
   ...
   while (criteria.hasMoreElements()) {
      wakeup = (WakeupCriterion) criteria.nextElement();
      if (wakeup instanceof WakeupOnAWTEvent) {
         event = ((WakeupOnAWTEvent)wakeup).getAWTEvent();
         for (int i=0; i<event.length; i++) {
            id = event[i].getID();
            if (id == MouseEvent.MOUSE_DRAGGED) {
            ... }
            else if (id == MouseEvent.MOUSE_PRESSED) {
            ... }
      ...
      pickRay.set(mousePos, mouseVec);
      sceneGraphPath = branchGroup.pickAllSorted(pickRay);
      ...
      if (sceneGraphPath != null) {
         for (int j=0; j<sceneGraphPath.length; j++) {
            if (sceneGraphPath[j] != null) {
               Node node = sceneGraphPath[j].getObject();
               ... // do something with node picked
   wakeupOn (mouseCriterion);  ... }
```


Navigation w/ mouse using SimpleUniverse

```
TransformGroup viewTG = new TransformGroup();
viewTG =
   su.getViewingPlatform().getViewPlatformTransform();
...
// For each mouse behavior
MouseRotate myMouseRotate = new
MouseRotate(MouseBehavior.INVERT_INPUT);
myMouseRotate.setTransformGroup(viewTG);
myMouseRotate.setSchedulingBounds(mouseBounds);
edgeBG.addChild(myMouseRotate);
...
```

see AxisView.java