JAVA 3D

Announcement

- Professor Wang's office hours for the week of Nov 14th and Nov 21th
  - Week of Nov 14th, Wednesday 9am - 10am, Thursday 8:30am - 9:30am
  - Week of Nov 21st, Tuesday 8:30am - 9:30am

Some Application Areas

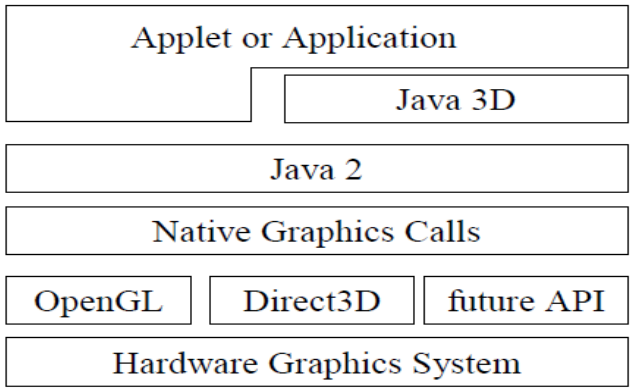
- Scientific/Medical/Data visualization
- Geographical information systems (GIS)
- Computer-aided design (CAD)
- Simulation
  - HW is here
- Computer-aided education (CAE)
- Games

Java 3D Overview

- A high-level Java API for building interactive 3D applications and applets
- Fast and efficient impl. on a variety of platforms
  - built on top of OpenGL and DirectX
- Other features:
  - object behaviors
  - user interaction
  - loaders for many 3D file formats

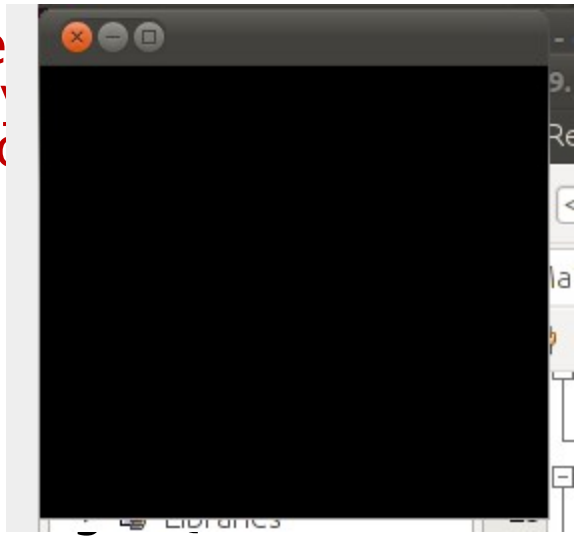
Java 3D API

- 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.)
- Simple to use :-)



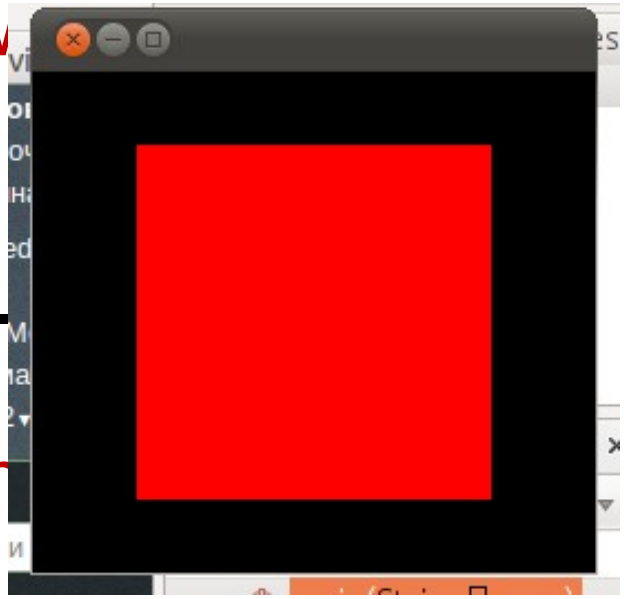
Let’s Draw smth. simple

```
package java3d;

    import
    com.sun.j3d.utils.universe.Simple
    import com.sun.j3d.utils.geometry
    import javax.media.j3d.BranchGro



    public class Main {


        public static void main(String[]

            SimpleUniverse uni =new SimpleUniverse();

    uni.getViewingPlatform().setNominalViewingTran
    sform();
        }

    }
```



Adding Cube

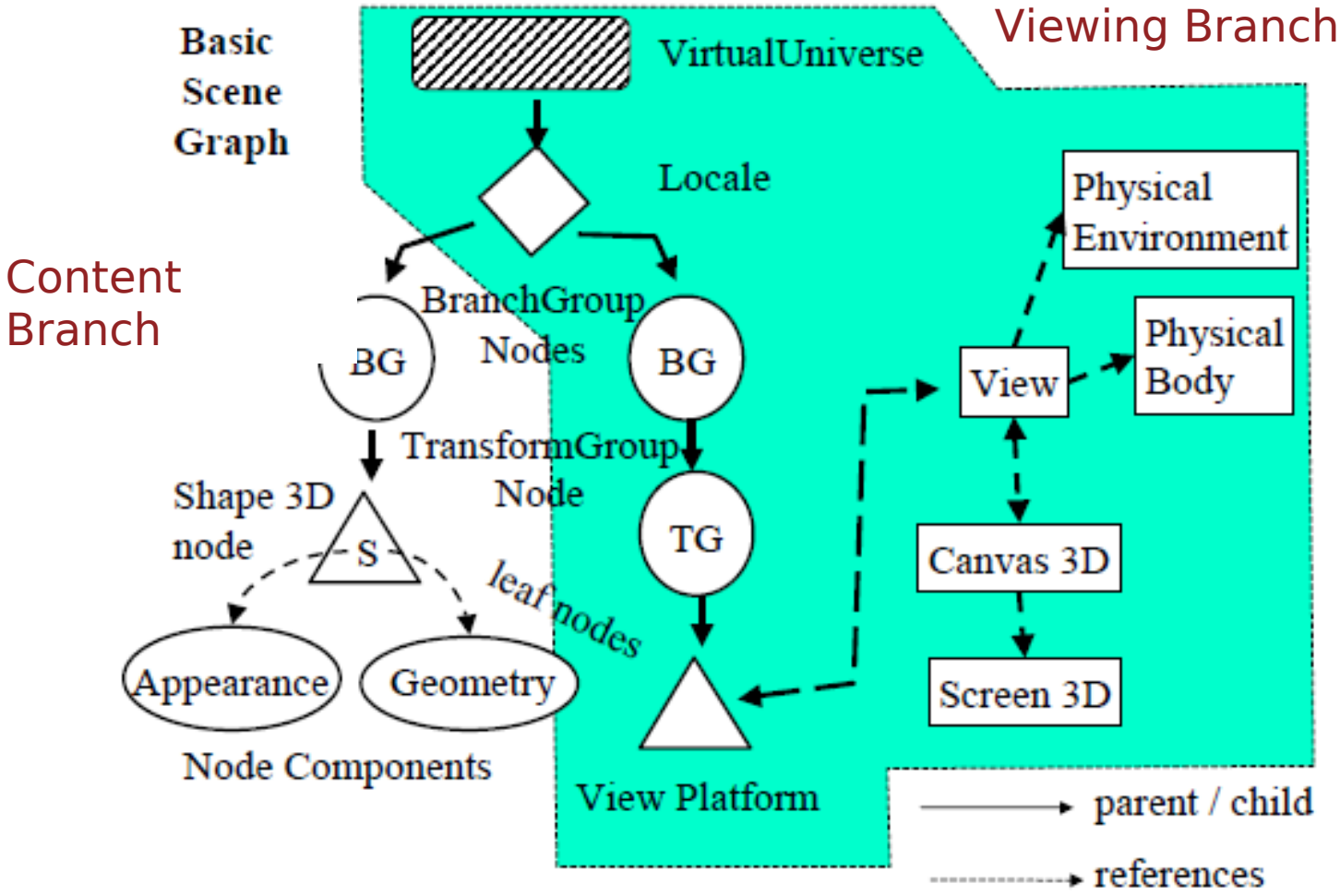
```
public static void main(String[] args) {
        SimpleUniverse uni =new
 SimpleUniverse();
        BranchGroup groups = new
 BranchGroup();
        groups.addChild(nev
 ColorCube(0.5));

 uni.getViewingPlatform().
 ingTransform();
        uni.addBranchGraph
    }
```



What is a Scene Graph?

- A *scene graph* is a tree-like data structure(DAG) that stores, organizes, and renders 3D scene information (3D objects, materials, lights, behaviours).

- The scene graph makes 3D programming considerably easier than directly coding in OpenGL or DirectX.
- One path from the locale to a leaf node path describes how the leaf is rendered



1)Scene graph is a DAG -- there is one path from the locale to a leaf

2)Nodepath describes how the leaf is rendered.

is where all the action will be taking place

Canva3D is where all the action will be taking place.

Writing a Java 3D Program

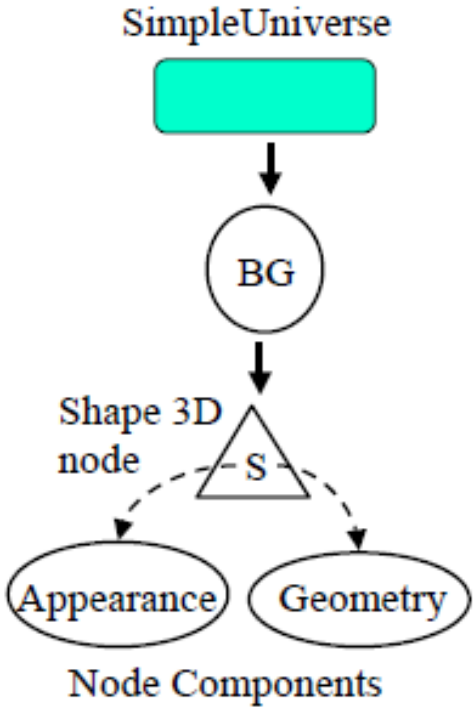
1. create a Canvas3D
2. create a VirtualUniverse
3. create a Locale object, attach to VirtualUniverse
4. construct a view branch graph
   a. create View object
   b. create ViewPlatform
   c. create a PhysicalBody
   d. create a PhysicalEnvironment
   e. attach ViewPlatform, PhysicalBody, PhysicalEnvironment, Canvas3D to View

Viewing Branch

5. construct content branch graph
6. compile branch graph
7. insert subgraphs into Locale

Simple Universe

- SimpleUniverse -- convenience, beginning
  - ignore viewing branch graph



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 Java 3D 5
  - SimpleUniverse()
  - SimpleUniverse(Canvas3D canvas3D) // references canvas3D
  - void addBranchGraph(BranchGroup gb) // add content to Locale
- BranchGroup methods
  - BranchGroup();
  - addChild()
  - 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

Example

- 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.ColorCu
  be;
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
```

standard Java code for BorderLayout

```
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
```

don't worry, after adding it

```
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
```

add Canvas3D to center of BorderLayout

```
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
```

create your SceneGraph contents

```
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
```

optimize your SceneGraph, not necessary, allow your program to run faster.

```
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
```

setup the SimpleUniverse, attach the Canvas3D

```
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().setNominalViewi
```

set the ViewingPlatform (where the User is)

```
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
```

add your SceneGraph to the SimpleUniverse

```
....
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);
}
```

Create cube

```
....
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);
}
```

if called as an application, a 500x500 window will be opened

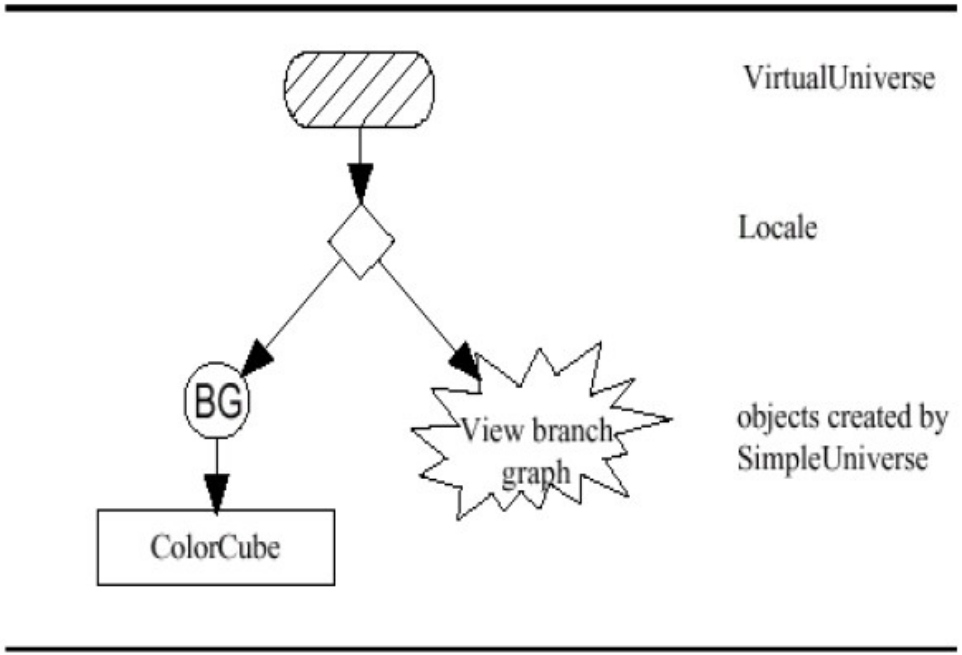


Figure 1-11 Scene Graph for HelloJava3Da Example

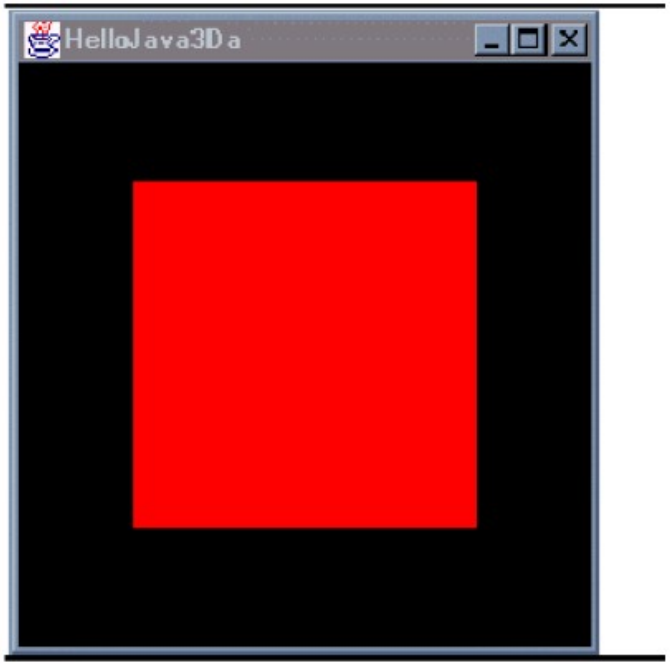
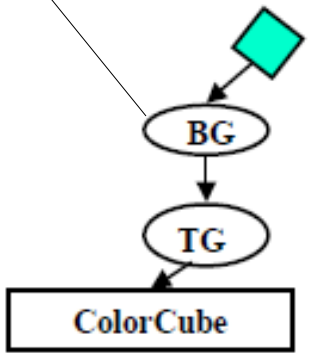


Figure 1-12 Image Produced by HelloJava3Da

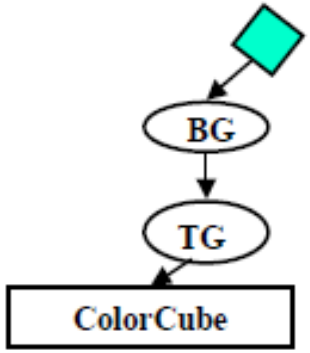
Adding translations

- 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)

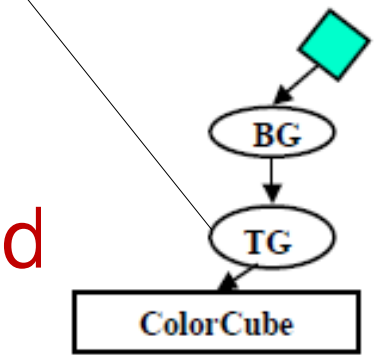
```
public BranchGroup
  createSceneGraph() {
  BranchGroup objRoot = new
  BranchGroup();
  Transform3D r1 = new
  Transform3D(), r12 = new
  Transform3D();
  r1.rotX(Math.PI/4.0d);
  r2.rotY(Math.PI/5.0d);
  r1.mul(r2);
  TransformGroup objRotated
  TransformGroup(r1);
  objRotated.addChild(new
```



```
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
  TransformGroup(r1);
  objRotated.addChild(new
```



```
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
  TransformGroup(r1);
  objRotated.addChild(new
```



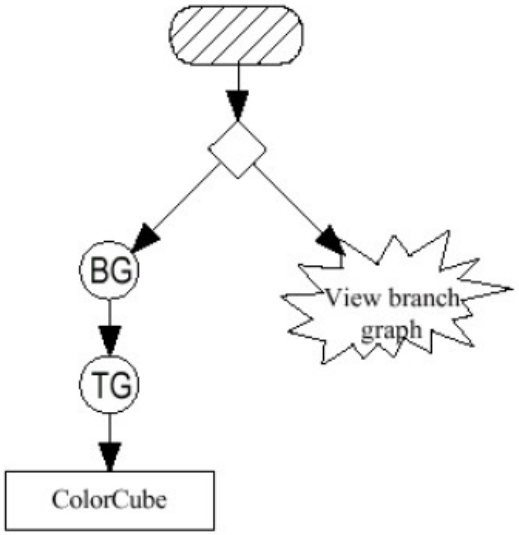


Figure 1-14 Scene Graph for HelloJava3Db Example

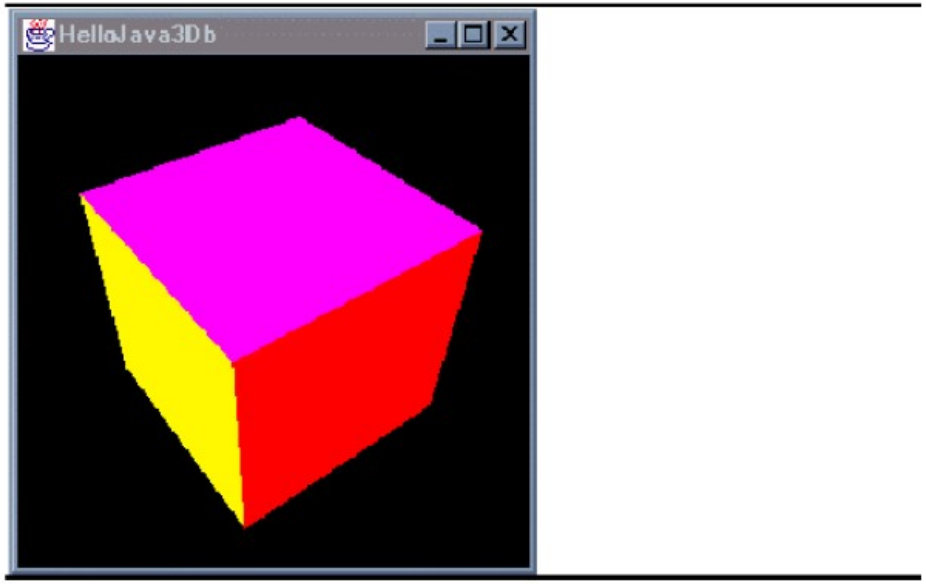


Figure 1-15 Image of the Rotated ColorCube Rendered by HelloJava3Db

Java 3D Installation

- Install Java
  - http://www.java.com/en/download/inc/windows_upgrade_xpi.jsp
  - Download
  - Follow installation instructions
- Install Java 3D jars
  - http://www.oracle.com/technetwork/java/javase/tech/inde
  - Download
  - Follow installation instructions

References

- Tutorials
  - http://www.vrupl.evl.uic.edu/LabAccidents/jav
    (VERY GOOD!)
  - http://www.youtube.com/watch?v=pW5pPHQ-
- Simple Simulation creation
  - http://fivedots.coe.psu.ac.th/Software.coe/LAB/Java3D/
  - www.cg.tuwien.ac.at/courses/CG2/SS2002/Java3D_slid