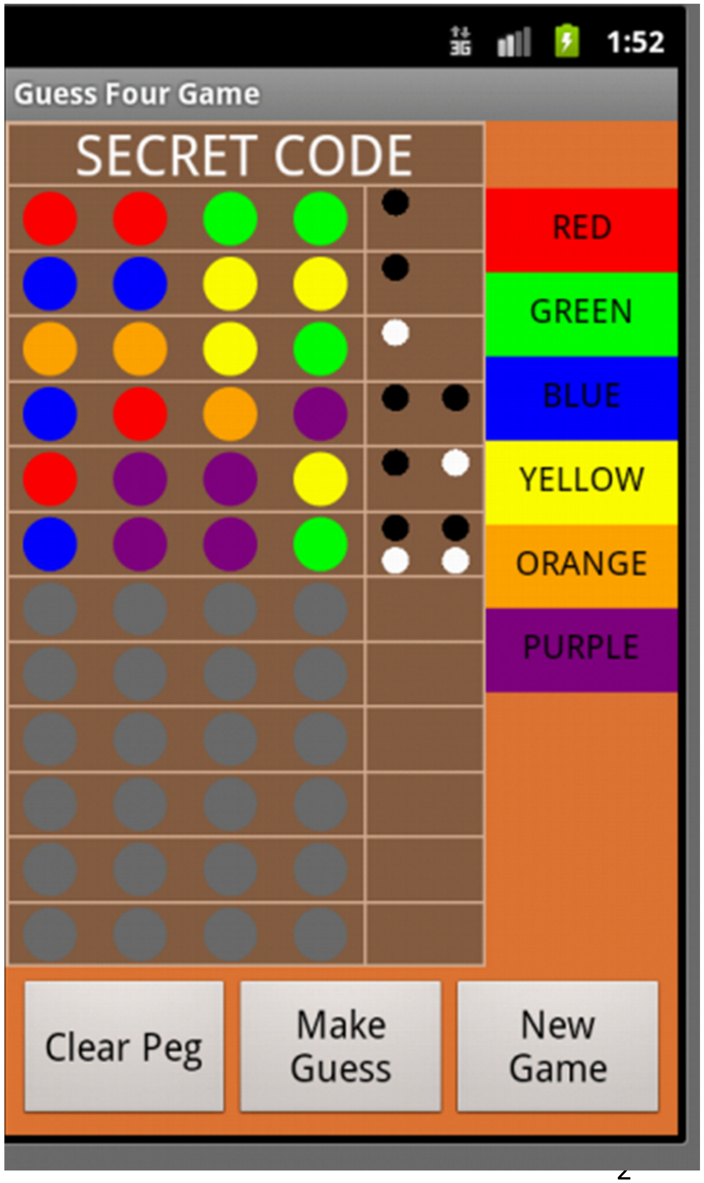
CS324e - Elements of Graphics and Visualization

Java 3D Intro

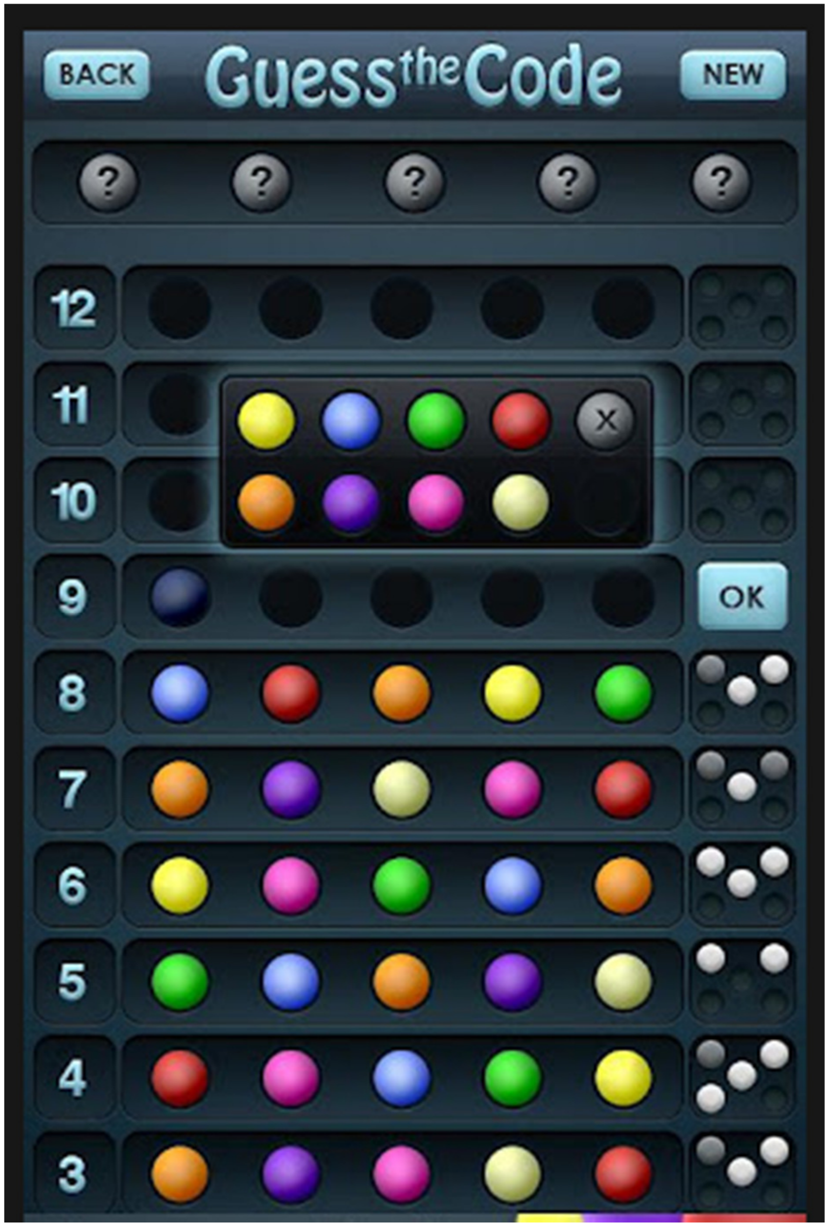
Java 2D

- Java2D and Swing part of standard Java
- Various attempts to make two d graphics appear more "lifelike" and 3 dimensional

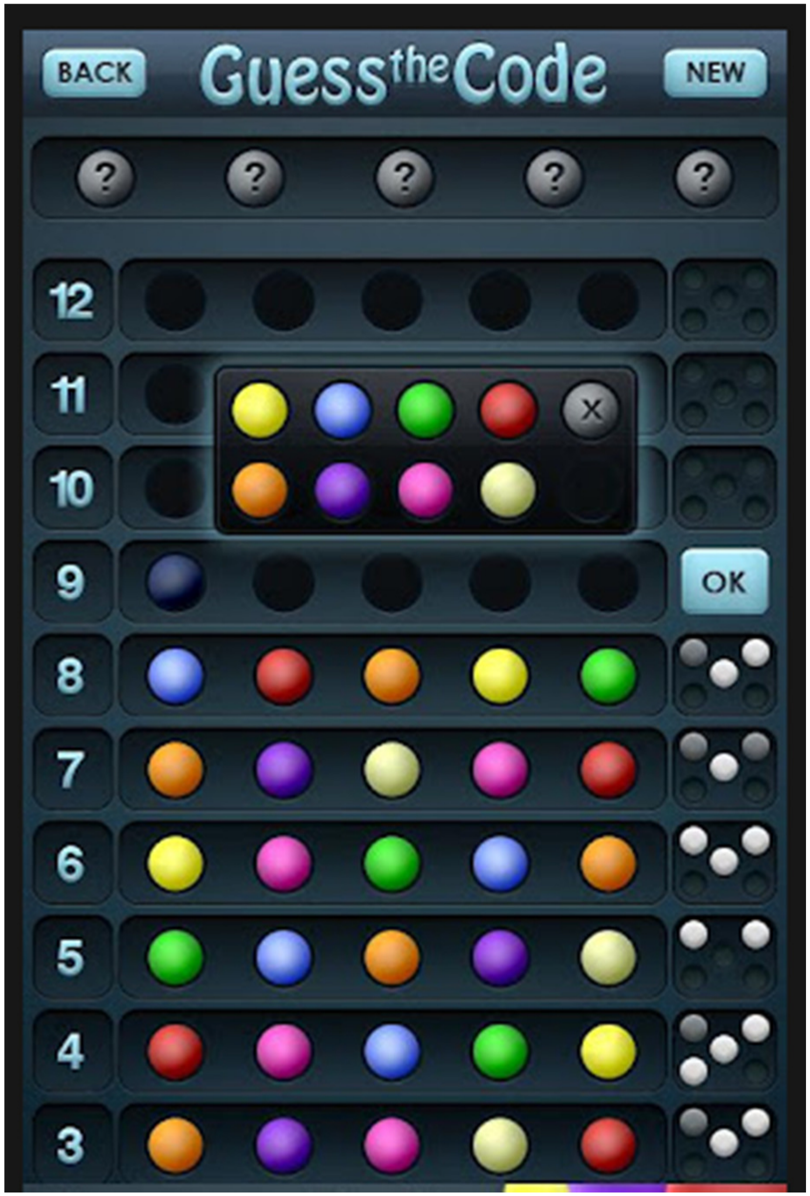


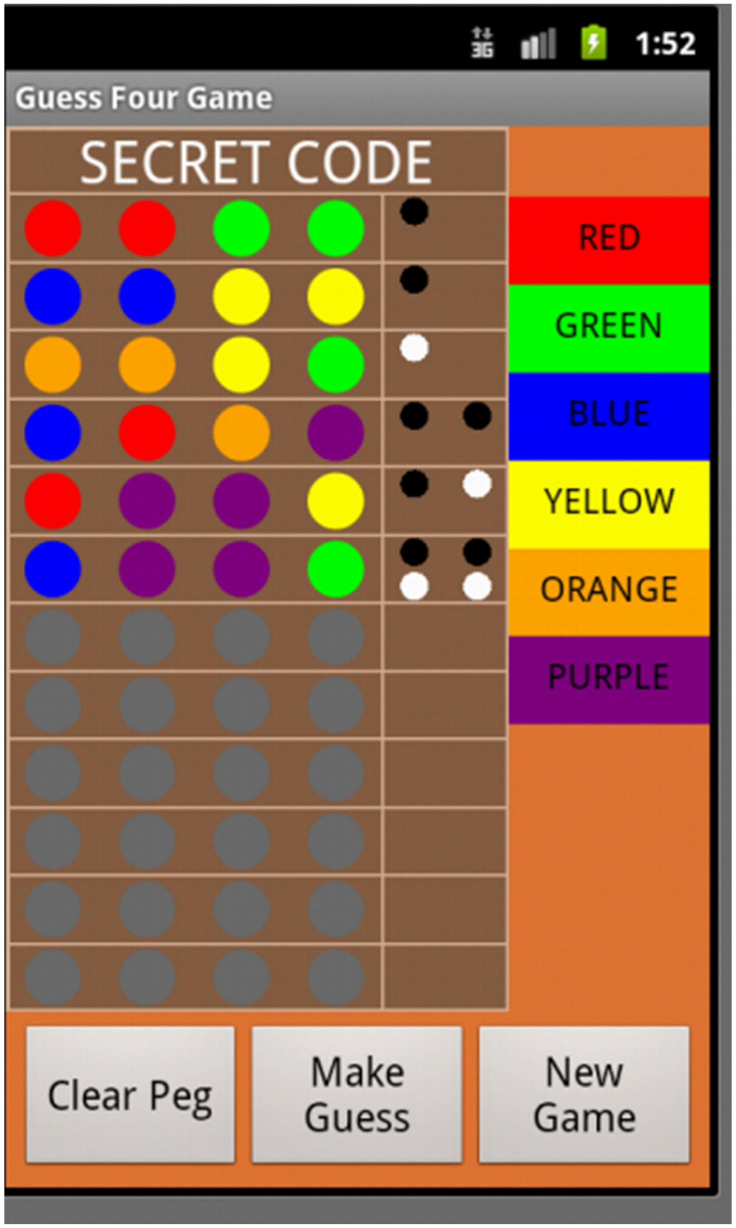
Gradients

- Gradient Paints can add depth to 2d primitives
- Notice the gradient paint on the pegs and shading on numbers



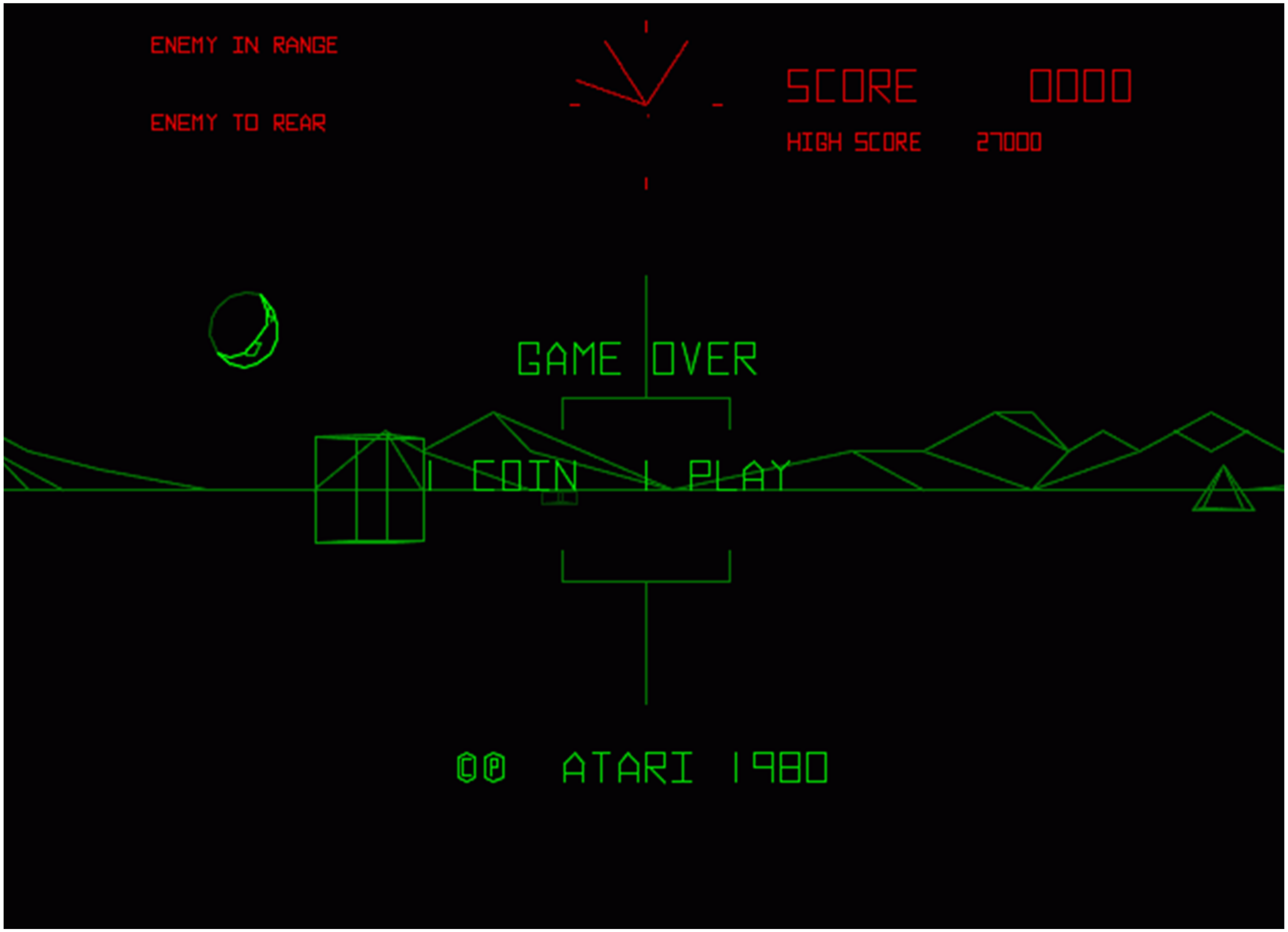
2D Graphics


BACK
GuesstheCode
NEW
?
12
11
10
9
OK
8
7
6
5
4
3


1:52
Guess Four Game
SECRET CODE
RED
GREEN
BLUE
YELLOW
ORANGE
PURPLE
Clear Peg
Make Guess
New Game

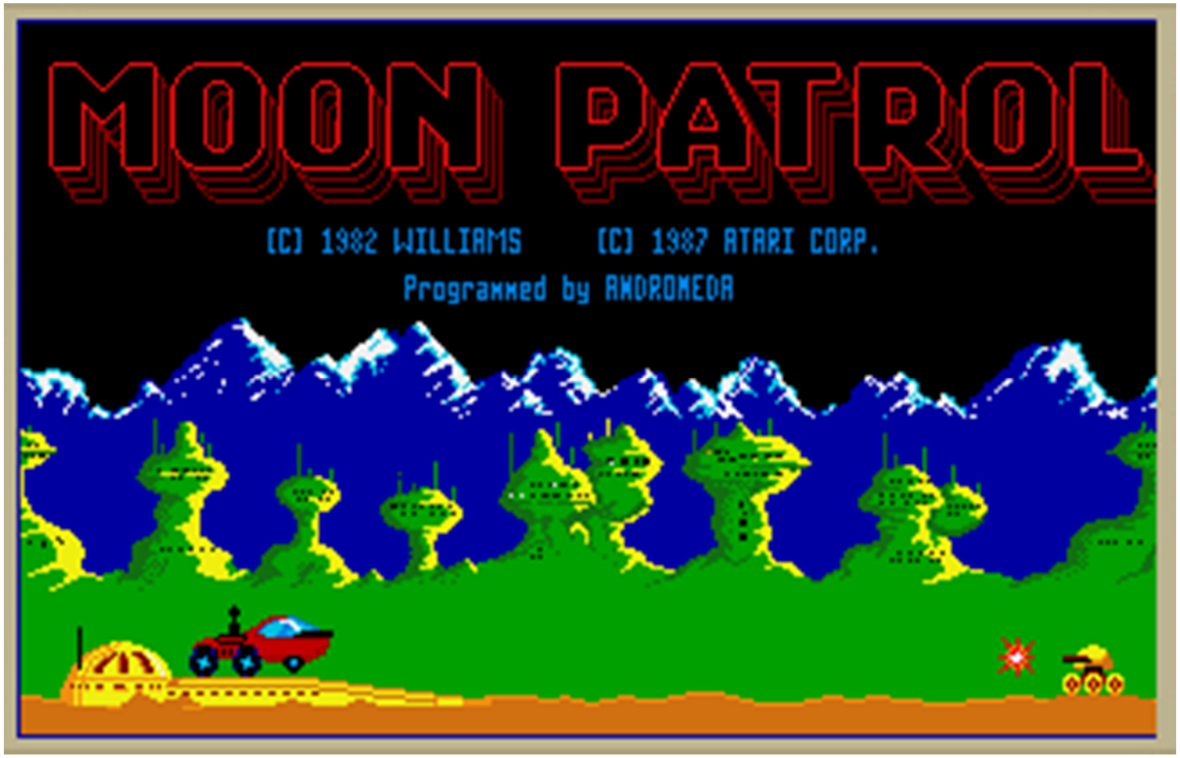
Wireframe Vector Graphics

- BattleZone - 1980

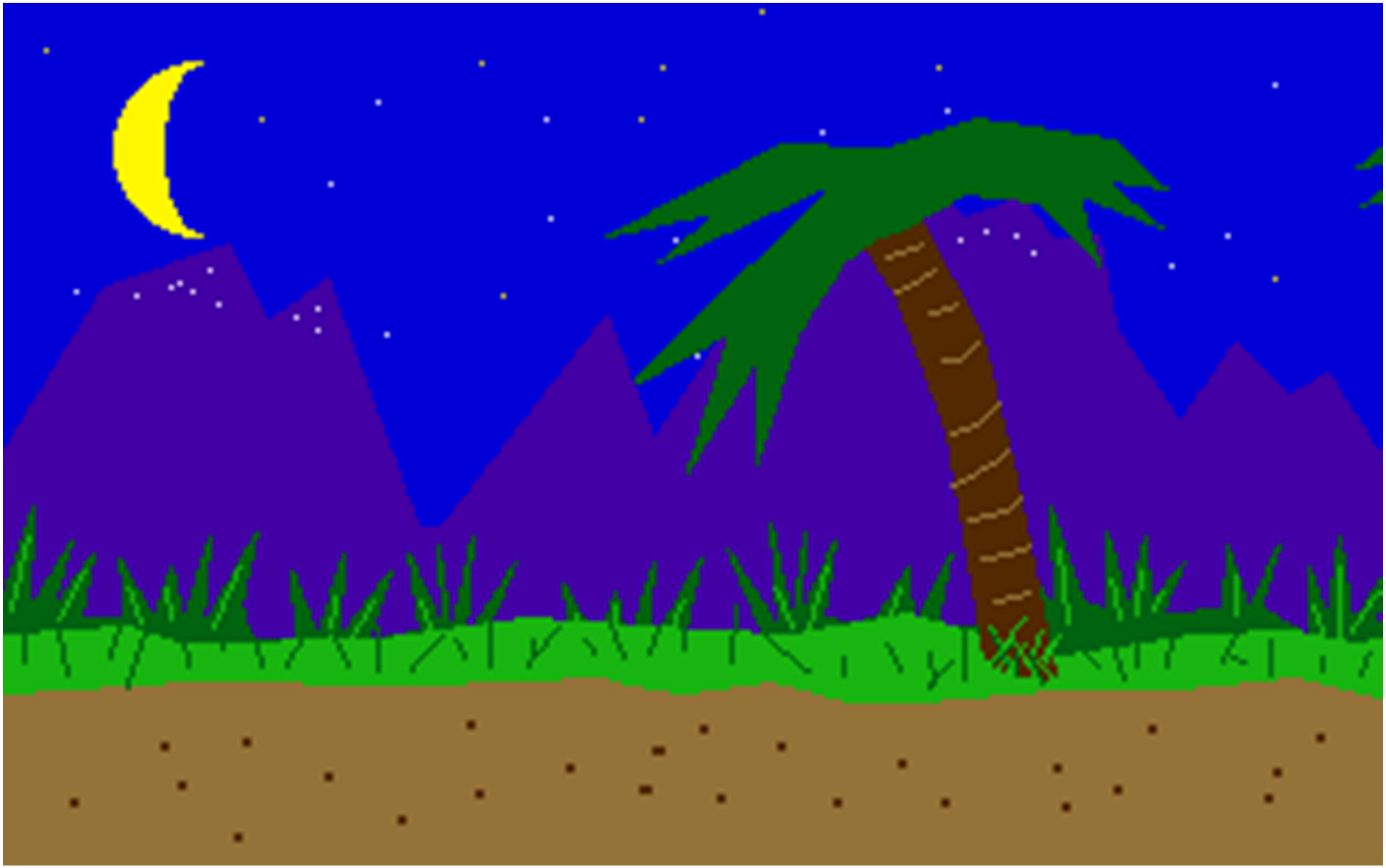


Parallax Scrolling

- multiple backgrounds
- backgrounds closer to view move at a faster speed than backgrounds farther away

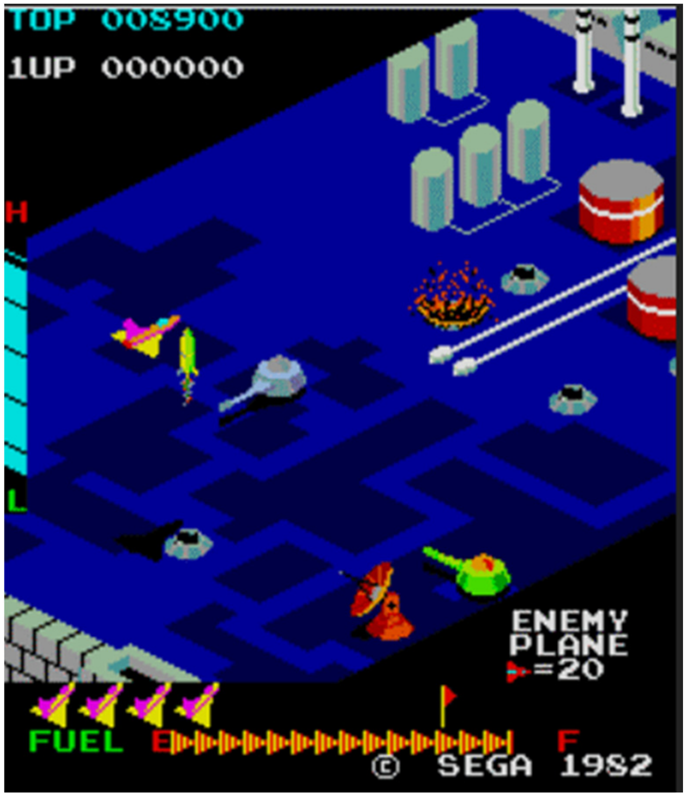


Parallax Scrolling Example

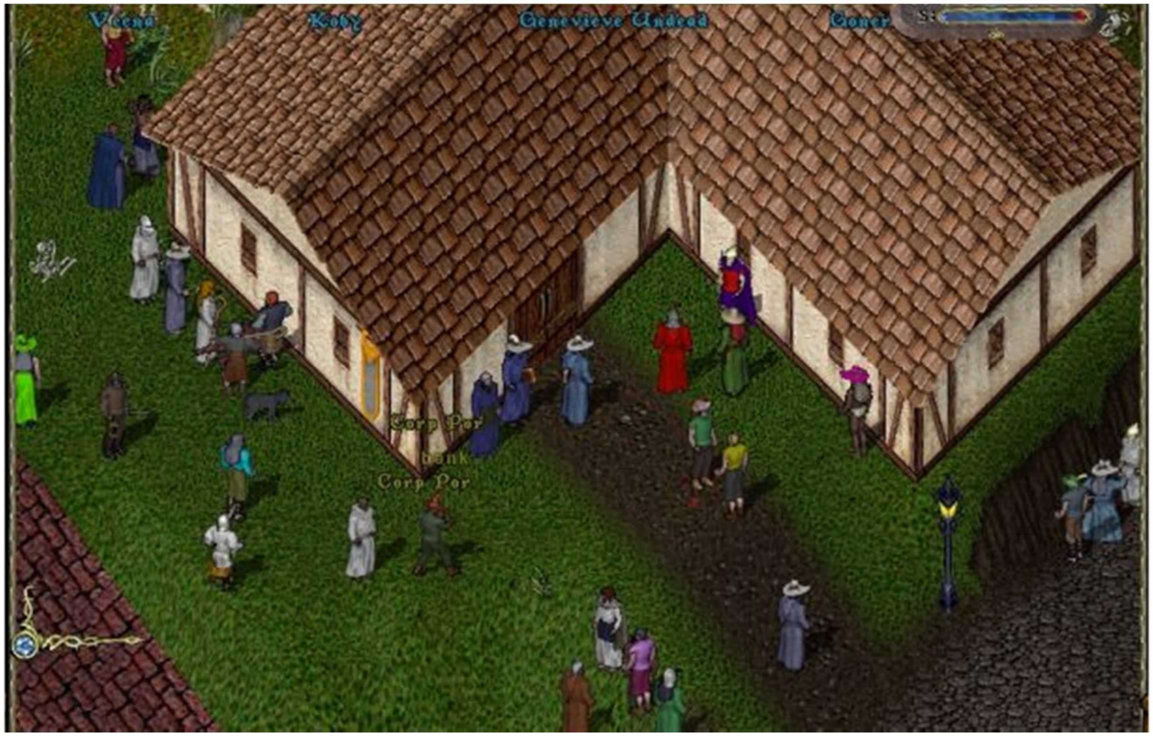


2.5D

- Isometric Graphics
- "rotate" object to reveal details on the side



Zaxxon



Ultima Online

3D Graphics

- Create 3D model
  - a small scene or a large world
- Model rendered into a 2D projection
- model includes
  - objects (boxes, cones, cylinders, sphere, user defined models)
  - lighting
  - cameras
  - textures
  - dynamic behaviors

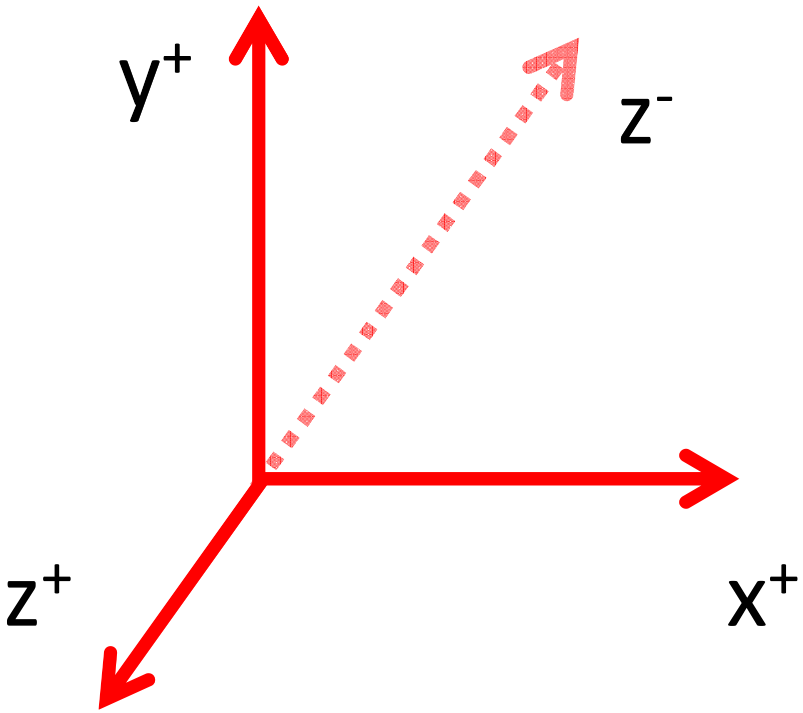
Java3D

- Not standard Java
- One of multiple non standard libraries to create 3d graphics in Java
  - others include
  - JOGL, jMonkey Engine, Ardor3D, JReality, LWJGL

- Java3D websites:
- http://java3d.java.net/
- http://www.oracle.com/technetwork/java/javase/tech/index-jsp-138252.html

Java 3D Coordinate System

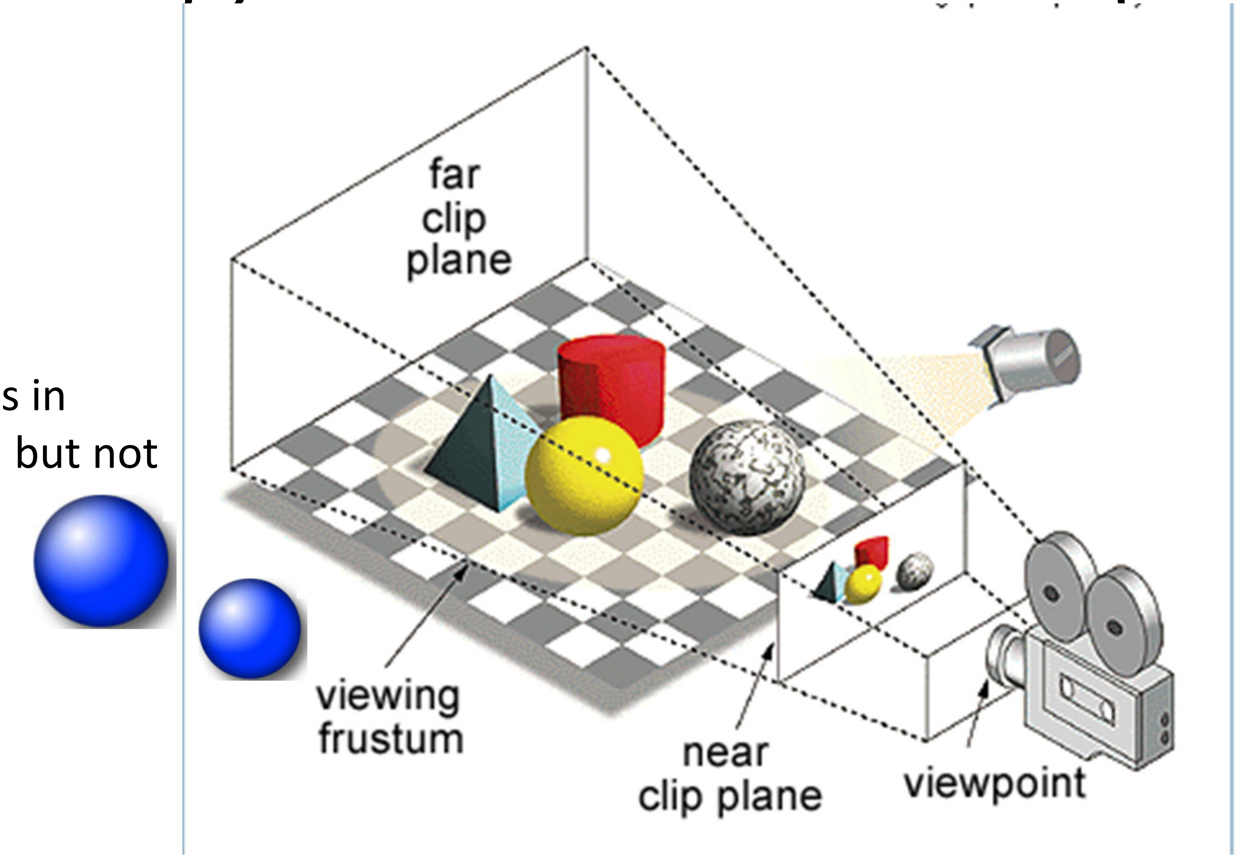
- x and y as expected (positive y is up, not down as in 2d graphics
- z axis - positive z is out of screen, negative z is into screen



Visual Portion

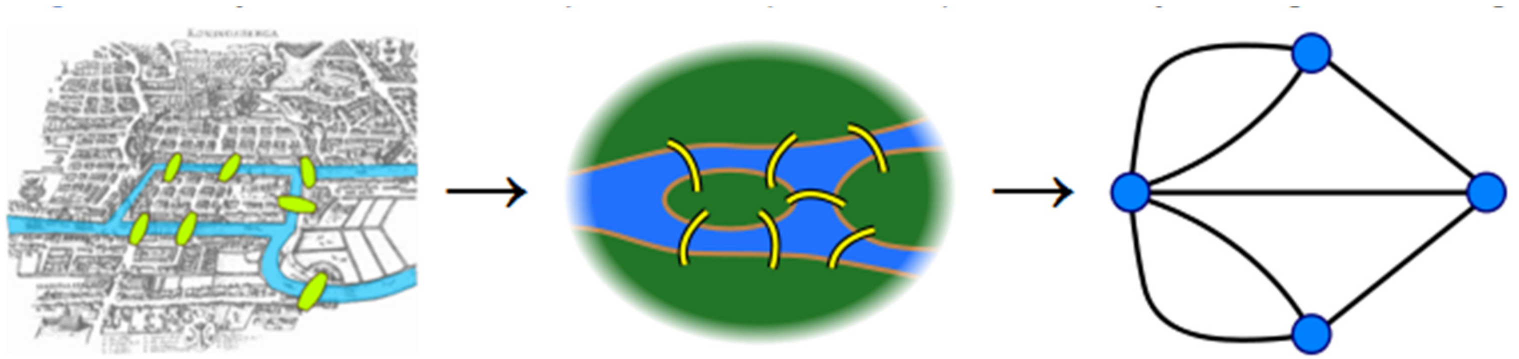
- Portion of 3D Scene that is rendered is contained in a *frustum* *(pro: frəstəm)*
  - a pyramid or cone with its top cut off

objects in scene, but not visible



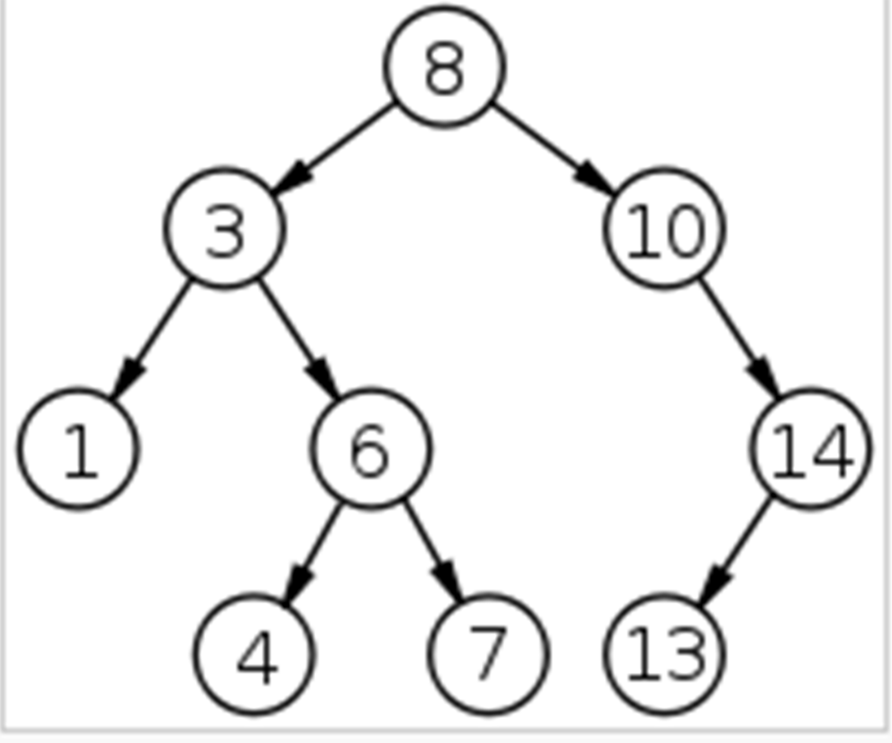
Scene Graphs

- The elements of a Java3D scene are stored in a data structure known as a *scene graph*
- Graph consist of *nodes* (aka *vertices*) that contain a piece of data and are connected to other nodes by *links* (aka *edges*)

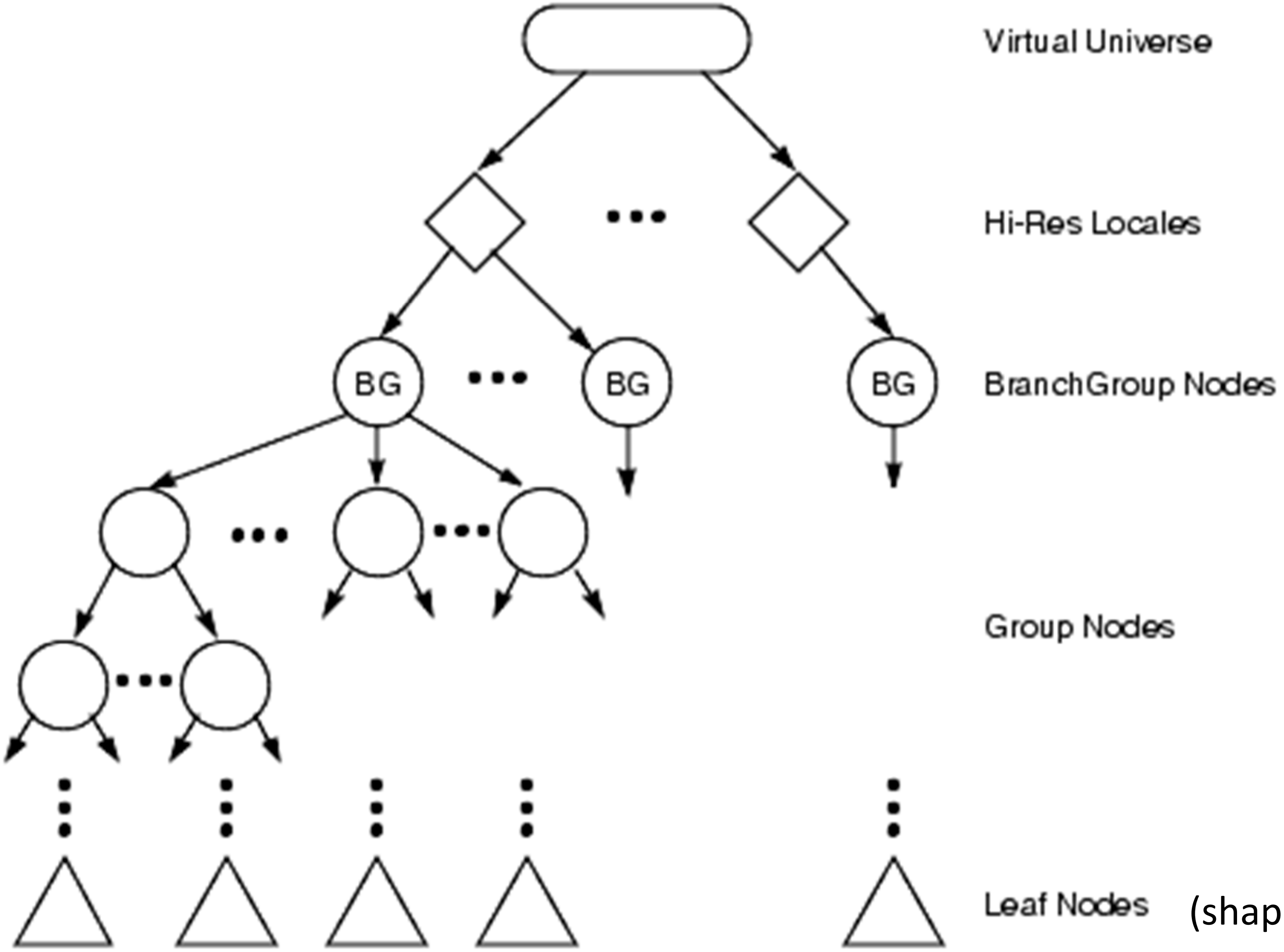


Trees - A Kind of Graph

- Trees and Binary Trees are special instances of Graphs
- root is node that contains 8
- leaves on the bottom



Java3D Scene Graphs



Virtual Universe

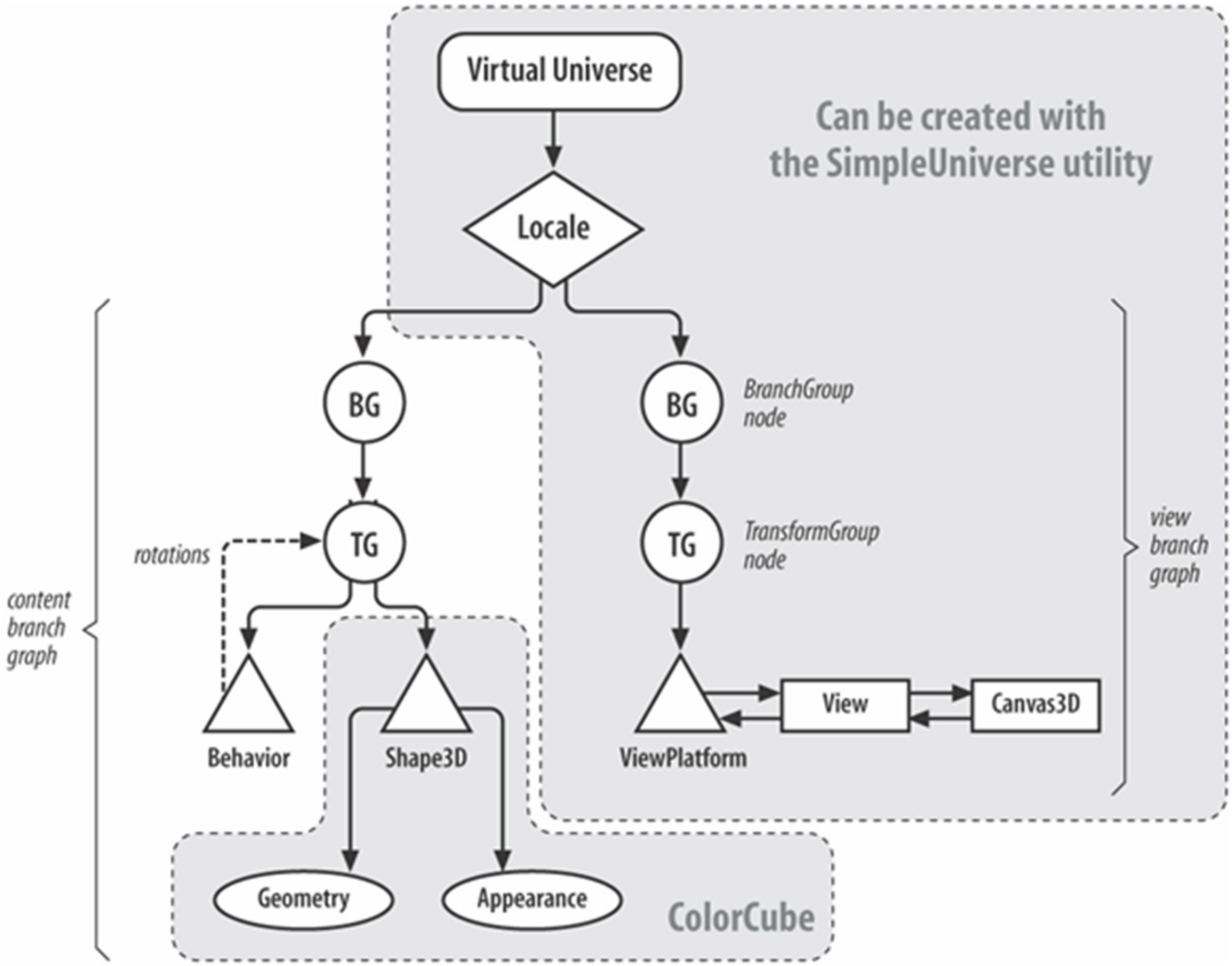
Hi-Res Locales

BranchGroup Nodes

Group Nodes

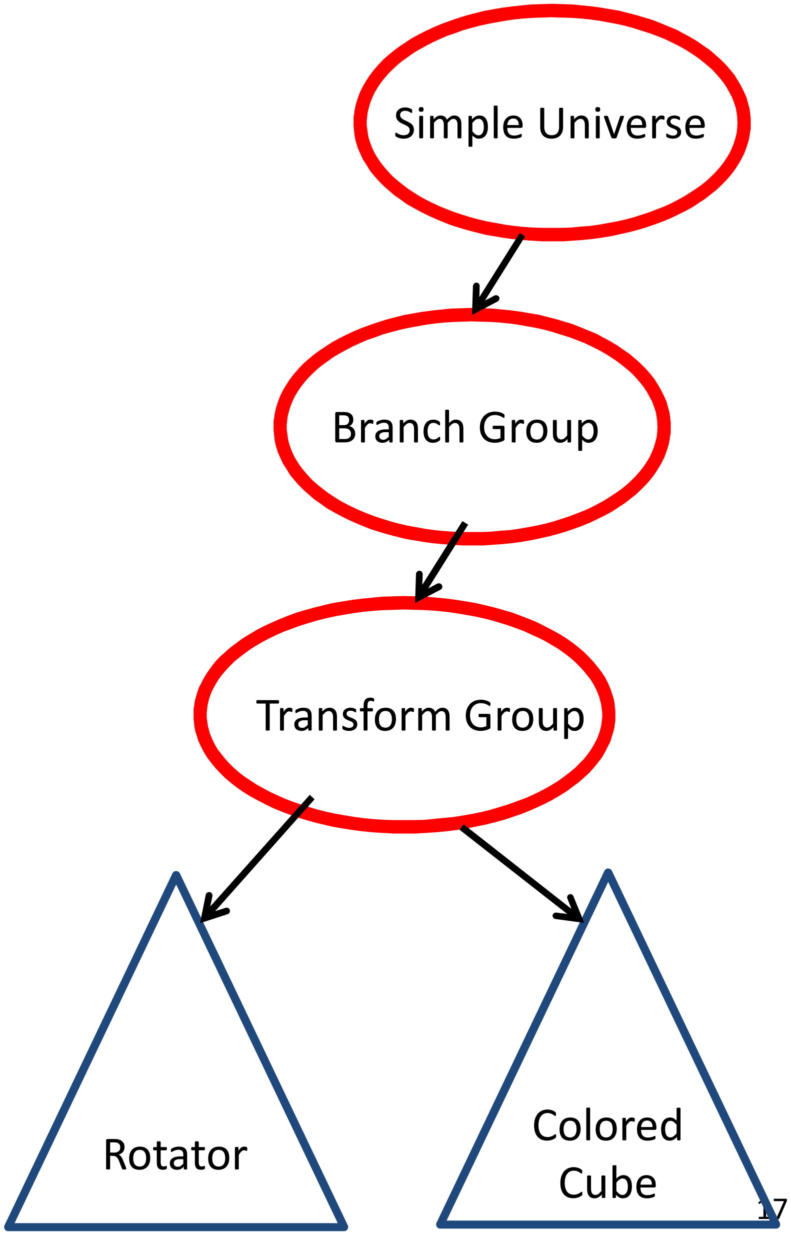
Leaf Nodes (shapes, lights)

Java3D Scene Graphs



HelloUniverse

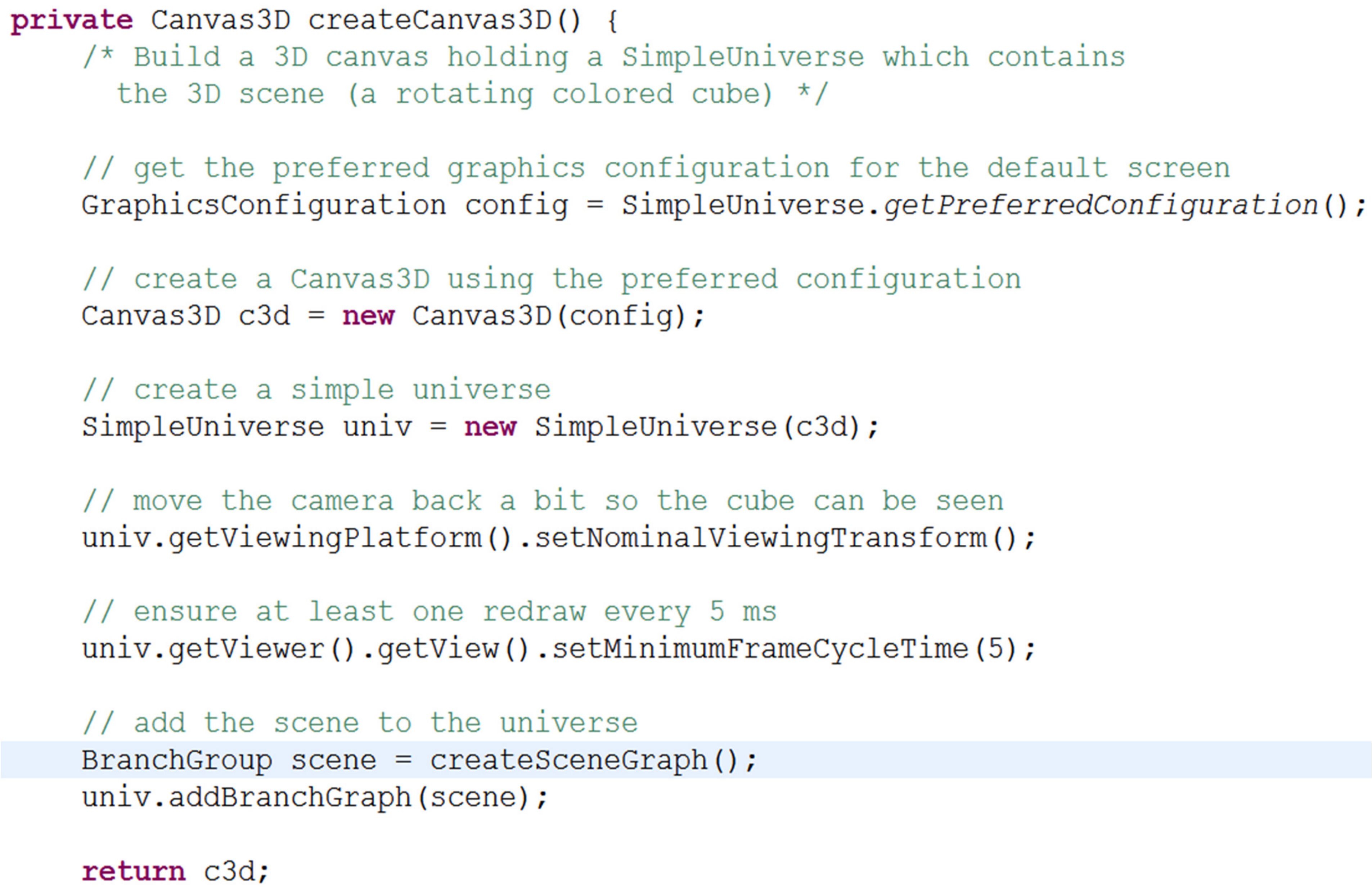
- Program to test installation of Java3D libraries
- Simple Scene Graph
- http://www.java2s.com/Code/Java/3D/HelloUniverse1.htm



HelloUniverse Code

- Root of Scene Graph is SimpleUniverse object
  - convenience class to set up ViewingPlatform, Locale, Viewer
- canvas3D
  - like a BufferedImage
  - once set up in graph we don't interact with in simple examples

HelloUniverse Code



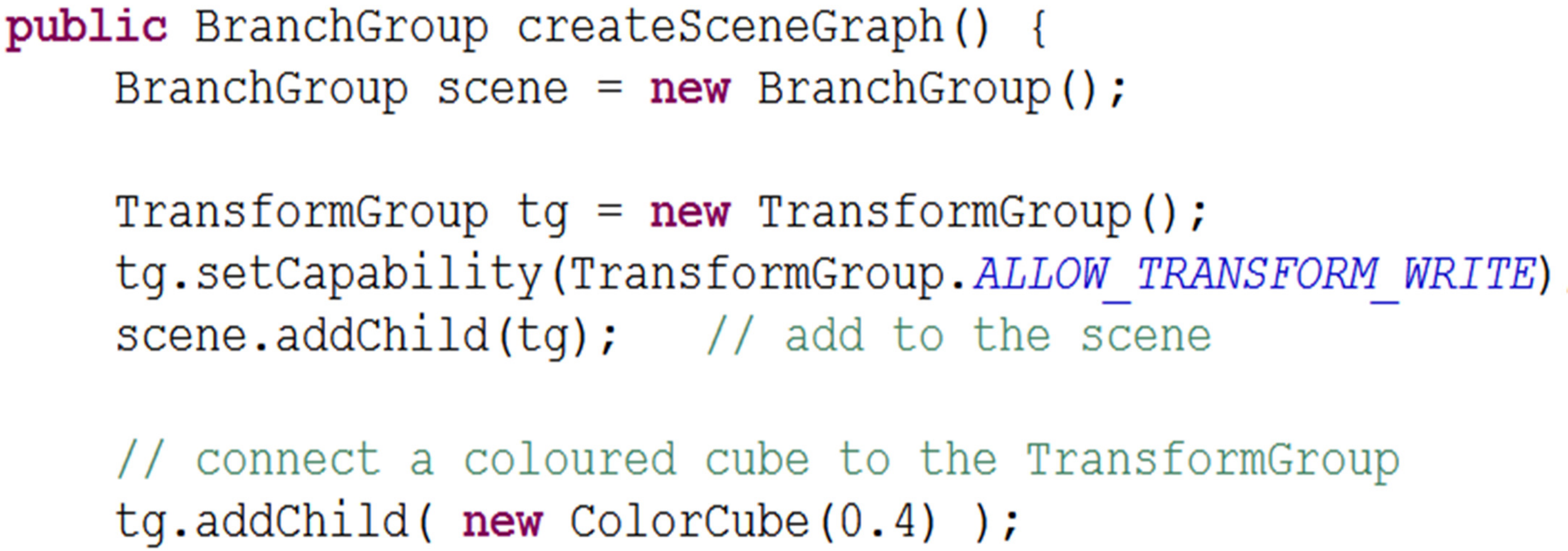
```
private Canvas3D createCanvas3D() {
    /* Build a 3D canvas holding a SimpleUniverse which contains
      the 3D scene (a rotating colored cube) */

    // get the preferred graphics configuration for the default screen
    GraphicsConfiguration config = SimpleUniverse.getPreferredConfiguration();

    // create a Canvas3D using the preferred configuration
    Canvas3D c3d = new Canvas3D(config);

    // create a simple universe
    SimpleUniverse univ = new SimpleUniverse(c3d);

    // move the camera back a bit so the cube can be seen
    univ.getViewingPlatform().setNominalViewingTransform();

    // ensure at least one redraw every 5 ms
    univ.getViewer().getView().setMinimumFrameCycleTime(5);

    // add the scene to the universe
    BranchGroup scene = createSceneGraph();
    univ.addBranchGraph(scene);

    return c3d;
```

Create the Objects

- Branch Groups used to group related objects together
- Transform Groups used to perform transforms on all objects in the group (children)
- ColoredCube a class to allow a simple shape to be displayed with out having to set up materials or color

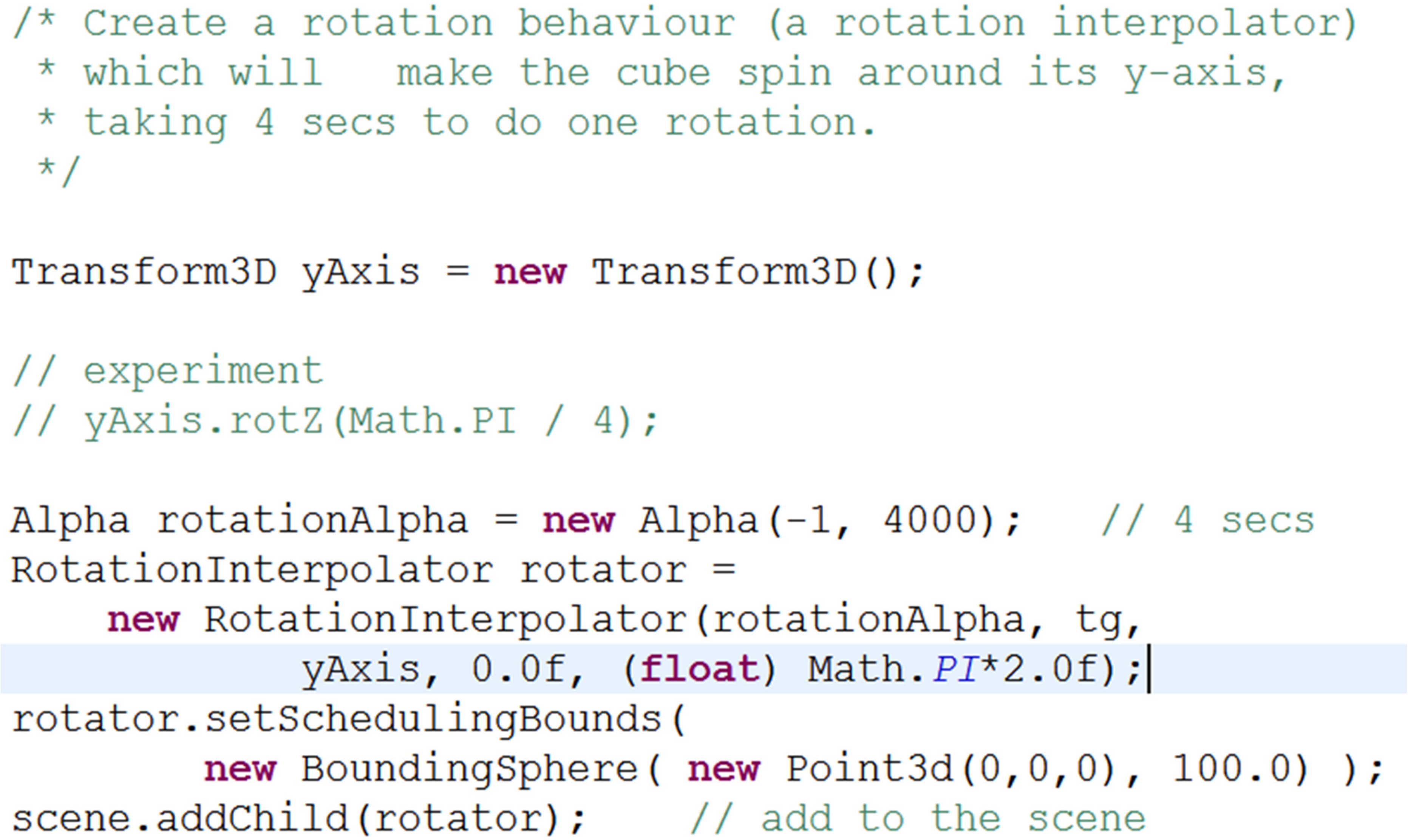
Adding ColoredCube

- 0.4 is size of cube
  - try different sizes when demoing program



```
public BranchGroup createSceneGraph() {
    BranchGroup scene = new BranchGroup();

    TransformGroup tg = new TransformGroup();
    tg.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE)
    scene.addChild(tg);   // add to the scene

    // connect a coloured cube to the TransformGroup
    tg.addChild( new ColorCube(0.4) );
```

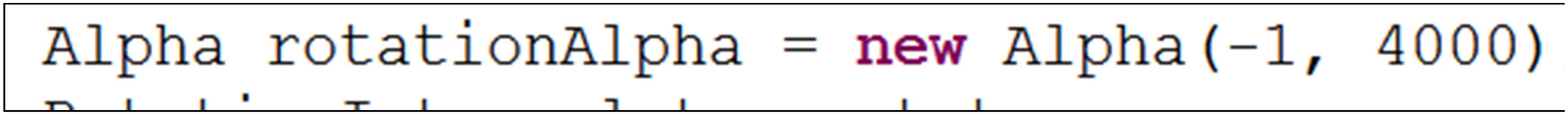
Adding Rotation Behavior



```
/* Create a rotation behaviour (a rotation interpolator)
 * which will   make the cube spin around its y-axis,
 * taking 4 secs to do one rotation.
 */

Transform3D yAxis = new Transform3D();

// experiment
// yAxis.rotZ(Math.PI / 4);

Alpha rotationAlpha = new Alpha(-1, 4000);   // 4 secs
RotationInterpolator rotator =
    new RotationInterpolator(rotationAlpha, tg,
            yAxis, 0.0f, (float) Math.PI*2.0f);
rotator.setSchedulingBounds(
        new BoundingSphere( new Point3d(0,0,0), 100.0) );
scene.addChild(rotator);    // add to the scene
```

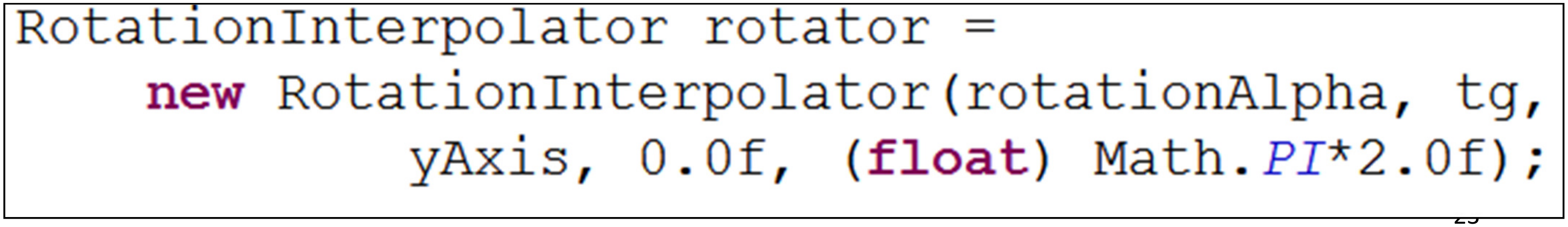
Rotation Behavior

- Alpha like the FRC Timing Framework interpolators
  - -1, loop continuously, 4000 milliseconds



```
Alpha rotationAlpha = new Alpha(-1, 4000)
```

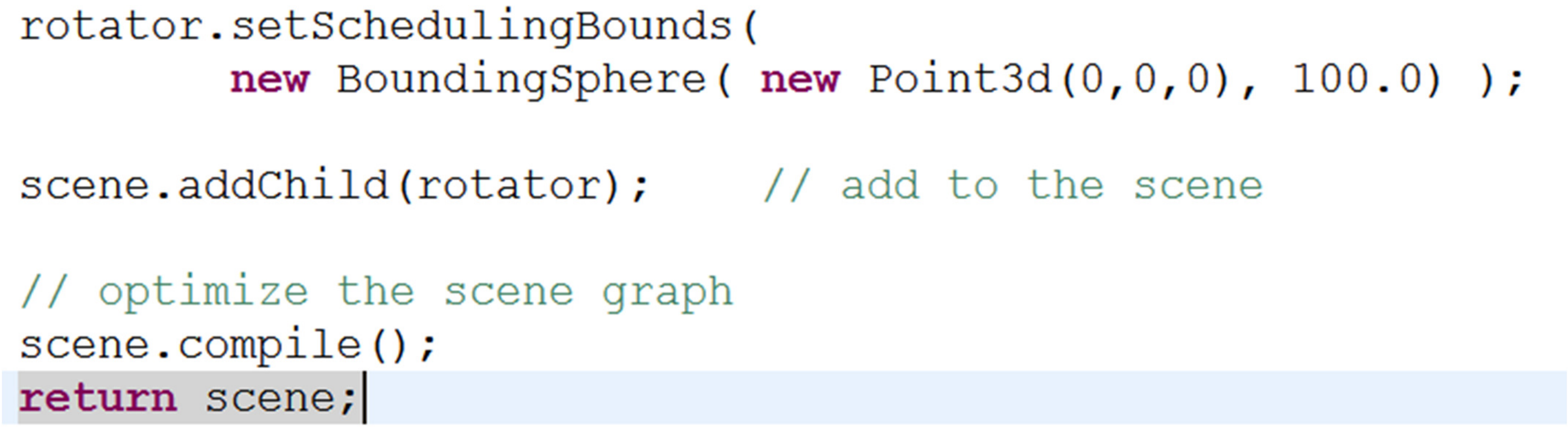
- alpha, transformGroup, transform3D (local coordinate system - rotation around y axis) , min angle, max angle



```
RotationInterpolator rotator =
    new RotationInterpolator(rotationAlpha, tg,
            yAxis, 0.0f, (float) Math.PI*2.0f);
```

Finishing SceneGraph

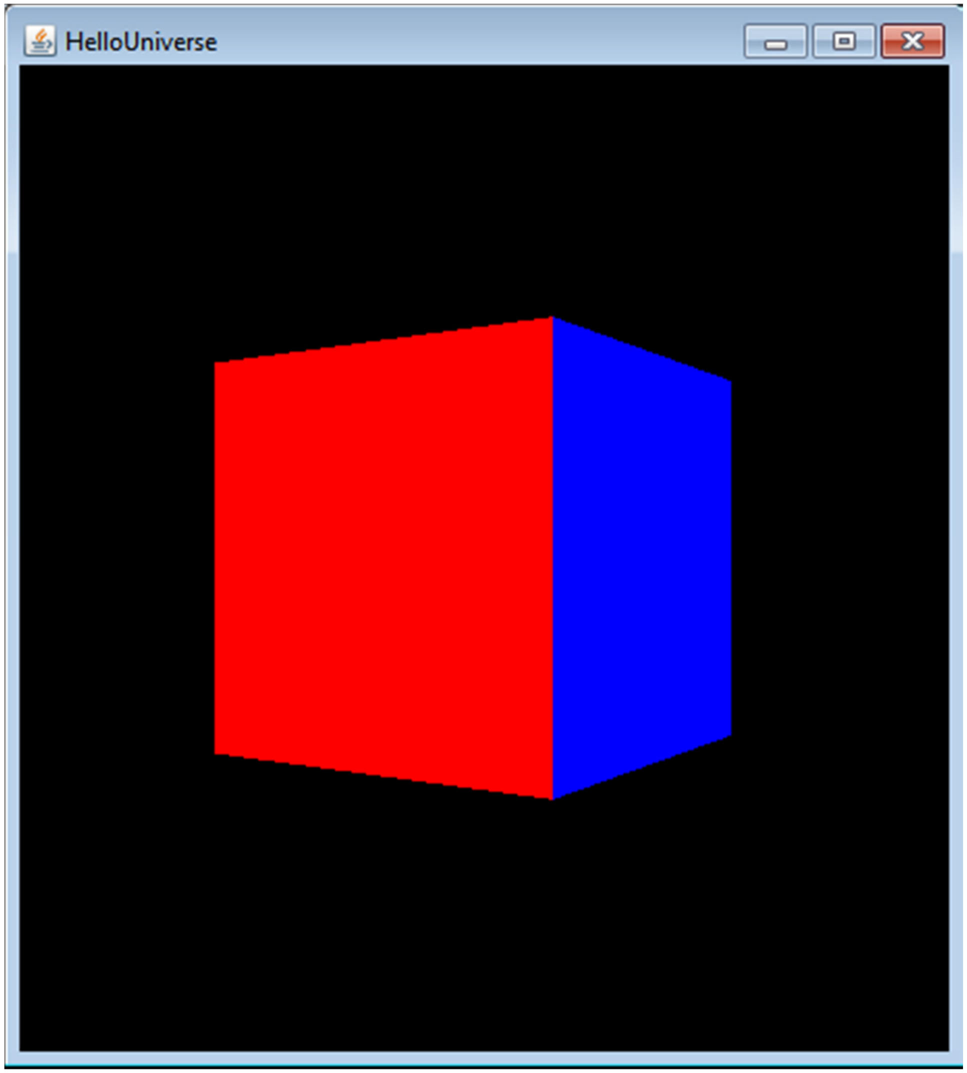
- Behaviors, such as rotation, have a bounds that must be set
- recall scene is the Branch Group



```
rotator.setSchedulingBounds(
        new BoundingSphere( new Point3d(0,0,0), 100.0) );

scene.addChild(rotator);    // add to the scene

// optimize the scene graph
scene.compile();
return scene;
```

Demo

- try making cube bigger
- try changing axis of rotation
- try adding another cube
- try changing position of cube
- why is background black?


HelloUniverse