Java 3D Examples

- Examples will often be taken from Andrew Davison, "Killer Game Programming", O'Reilly 2005
- Code and extra chapters available from http://fivedots.coe.psu.ac.th/~ad/jg/

- Java applications that embed the Canvas3D in a JPanel for inclusion within a Swing GUI
Checkers3D

The scene consists of
- a dark green and blue tiled surface (and red center)
- labels along the X and Z axes
- a blue background
- a floating sphere lit from two different directions
- the user (viewer) can move through the scene by moving the mouse

Checkers3D Scene Graph

Figure 77. Scene Graph for Checkers3D

( from Davison’s lectures)
Checkers3D Code Structure

- Windowed Java application
- Java Swing plus Java3D
- Checkers3D
  - Main class that extends JFrame
  - WrapCheckers3D JPanel added to JFrame
- WrapCheckers3D
  - Holds all the 3D code
  - Extends JPanel
  - Canvas3D added to JPanel
  - SimpleUniverse
  - createSceneGraph() method

Checkers3D Code

```java
public class Checkers3D extends JFrame {
    public Checkers3D() {
        super("Checkers3D");
        Container c = getContentPane();
        c.setLayout( new BorderLayout() );
        WrapCheckers3D w3d = new WrapCheckers3D();
        c.add(w3d, BorderLayout.CENTER);
        setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
        pack();
        setResizable(false);    // fixed size display
        setVisible(true);
    } // end of Checkers3D()

    public static void main(String[] args) {
        new Checkers3D();
    }
} // end of Checkers3D class
```
WrapCheckers3D Code

public class WrapCheckers3D extends JPanel
// Holds the 3D canvas where the loaded image is displayed
{

private SimpleUniverse su;
private BranchGroup sceneBG;
private BoundingSphere bounds; // for environment nodes

public WrapCheckers3D() // A panel holding a 3D canvas
{
    setLayout( new BorderLayout() );
    setOpaque( false );
    setPreferredSize( new Dimension(PWIDTH, PHEIGHT));

    GraphicsConfiguration config =
        SimpleUniverse.getPreferredConfiguration();
    Canvas3D canvas3D = new Canvas3D(config);
    add("Center", canvas3D);
    canvas3D.setFocusable(true); // give focus to the canvas
    canvas3D.requestFocus();

    su = new SimpleUniverse(canvas3D);

    createSceneGraph();
    initUserPosition(); // set user's viewpoint
    orbitControls(canvas3D); // viewpoint controls

    su.addBranchGraph( sceneBG );

} // end of WrapCheckers3D()
The Content Scene Graph

• The sphere, lights and background

private void createSceneGraph()
// initialise the scene
{
    sceneBG = new BranchGroup();
    bounds = new BoundingSphere(new Point3d(0,0,0), BOUNDSIZE);

    lightScene();       // add the lights
    addBackground();    // add the sky
    sceneBG.addChild( new CheckerFloor().getBG() );
    floatingSphere();   // add the floating sphere

    sceneBG.compile();  // fix the scene
} // end of createSceneGraph()

Bounds

bounds = new BoundingSphere(new Point3d(0,0,0), BOUNDSIZE);

• Much 3D scene rendering is computationally expensive
  - The effect of lights
  - Behaviours that animate objects

• Bounds define when to bother calculating lighting, animation etc
  - setInfluencingBounds for lights
  - setSchedulingBounds for behaviours

• Calculations only done if the current view intersects the bounds
Visible Object - The Sphere

private void floatingSphere()
// A shiny blue sphere located at (0,4,0)
{
// Create the blue appearance node
Color3f black = new Color3f(0.0f, 0.0f, 0.0f);
Color3f blue = new Color3f(0.3f, 0.3f, 0.8f);
Color3f specular = new Color3f(0.9f, 0.9f, 0.9f);

Material blueMat = new Material(blue, black, blue,
specular, 25.0f);
// sets ambient, emissive, diffuse, specular, shininess
blueMat.setLightingEnable(true);

Appearance blueApp = new Appearance();
blueApp.setMaterial(blueMat);

The Sphere (2)

// position the sphere
Transform3D t3d = new Transform3D();
t3d.set( new Vector3f(0,4,0));
TransformGroup tg = new TransformGroup(t3d);

tg.addChild(new Sphere(2.0f, blueApp));
// set its radius and appearance

sceneBG.addChild(tg);
} // end of floatingSphere()
private void lightScene()
/* One ambient light, 2 directional lights */
{
    Color3f white = new Color3f(1.0f, 1.0f, 1.0f);

    // Set up the ambient light
    AmbientLight ambientLightNode = new AmbientLight(white);
    ambientLightNode.setInfluencingBounds(bounds);
    sceneBG.addChild(ambientLightNode);

    // Set up the directional lights
    Vector3f light1Direction = new Vector3f(-1.0f, -1.0f, -1.0f);
    // left, down, backwards
    Vector3f light2Direction = new Vector3f(1.0f, -1.0f, 1.0f);
    // right, down, forwards
    DirectionalLight light1 =
        new DirectionalLight(white, light1Direction);
    light1.setInfluencingBounds(bounds);
    sceneBG.addChild(light1);
    DirectionalLight light2 =
        new DirectionalLight(white, light2Direction);
    light2.setInfluencingBounds(bounds);
    sceneBG.addChild(light2);
} // end of lightScene()
Coloured Background

```java
private void addBackground()
// A blue sky
{ Background back = new Background();
    back.setApplicationBounds( bounds );
    back.setColor(0.17f, 0.65f, 0.92f); // sky colour
    sceneBG.addChild( back );
} // end of addBackground()
```

Viewpoint Control

```java
private void orbitControls(Canvas3D c)
/* OrbitBehaviour allows the user to rotate around
the scene and zoom in and out. */
{
    OrbitBehavior orbit =
        new OrbitBehavior(c, OrbitBehavior.REVERSE_ALL);
    orbit.setSchedulingBounds(bounds);
    ViewingPlatform vp = su.getViewingPlatform();
    vp.setViewPlatformBehavior(orbit);
} // end of orbitControls()
```
Initial Viewer Position

- Viewer position needs to be set
- Use the lookAt() method
  - Viewer position
  - A point they are looking at
  - A vector specifying which way is up
- Apply to viewer transform group

Initialise Viewer Position

```java
private void initUserPosition()
// Set the user's initial viewpoint using lookAt()
{
    ViewingPlatform vp = su.getViewingPlatform();
    TransformGroup steerTG = vp.getViewPlatformTransform();

    Transform3D t3d = new Transform3D();
    steerTG.getTransform(t3d);

    // args are: viewer posn, where looking, up direction
    t3d.lookAt(USERPOSN, new Point3d(0,0,0), new Vector3d(0,1,0));
    t3d.invert();

    steerTG.setTransform(t3d);
}  // end of initUserPosition()
```
Alternative Viewer Position

- `setNominalViewingTransform()` method
  - `ViewPlatform` utility method
- Moves viewer position back along the Z-axis so that objects at the origin of size -1 to 1 along the x axis are fully viewable

```java
SimpleUniverse simpleU = new SimpleUniverse(canvas3D);

// This will move the ViewPlatform back a bit so the objects in the scene can be viewed.
simpleU.getViewingPlatform().setNominalViewingTransform();
```

The End