

# Principles of Computer Game Design and Implementation

## Lecture 5

# We already knew

- Introduction to this module
- History of video
- High-level information of a game
- Designing information for a game
- Execution of a game (Game loop)

# jMonkeyEngine

Architecture and Mathematical  
Concepts

# jMonkeyEngine

- A high performance scene graph based graphics API
- Completely open source (BSD License)
- Written in pure Java



jMonkeyEngine graphics

# jMonkeEngine History

- Started in 2003 by Mark Powell inspired by a C++ book *“3D Game Engine Design”*
- 2008 jMonkeyEngine v. 2.0
- 2009 Development stalled. Project forked.



jMonkeEngine v. 3.0

Ardor 3D

- |   |   |
|---|---|
| <ul style="list-style-type: none"><li>• Community-driven project</li><li>• New people joint</li><li>• Integration with free tools</li></ul> | <ul style="list-style-type: none"><li>• Commercial development</li><li>• Neater but less features</li></ul> |
|---|---|

# Version Differences

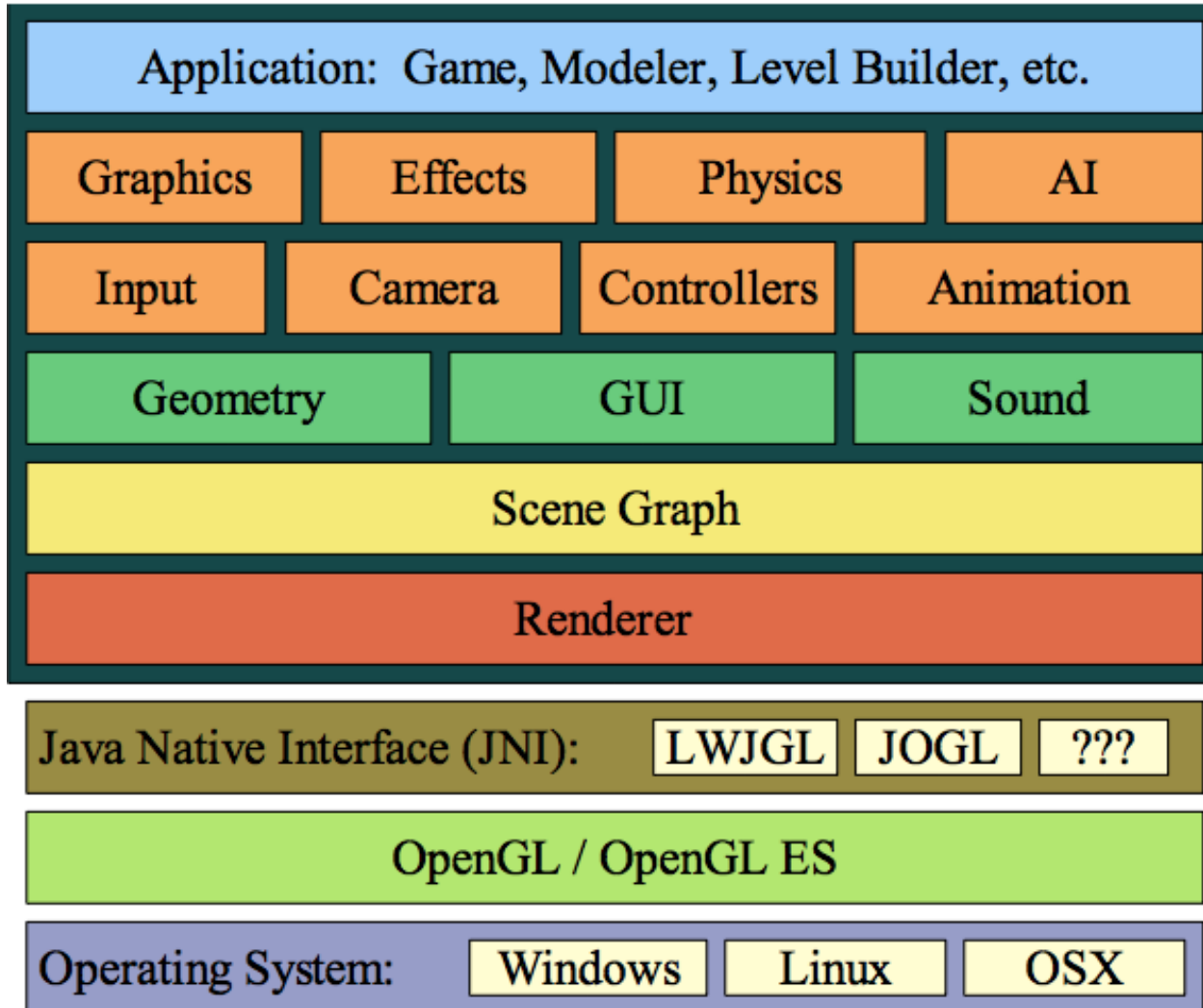
- jME v2.0
  - Stable
  - Uses OpenGL 1. Runs on *any* hardware
- jMe v2.1
  - Final release in the 2.x branch
- jME v3.0
  - Uses OpenGL 2. Runs well on *modern* hardware
  - **Shader** based
  - Physics engine integrated
  - jMonkeyPlatform

# jME Documentation

- Official site:

<http://www.jmonkeyengine.org>

# jME Architecture

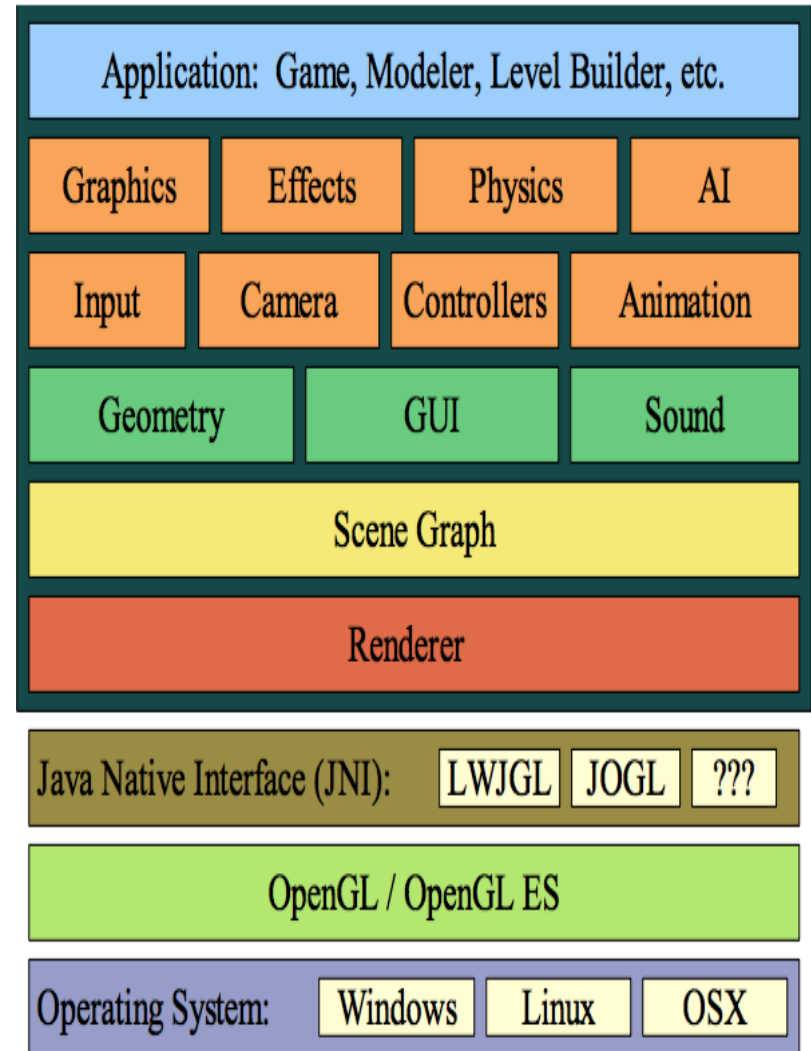


the picture is largely out-dated,  
but it still conveys the idea

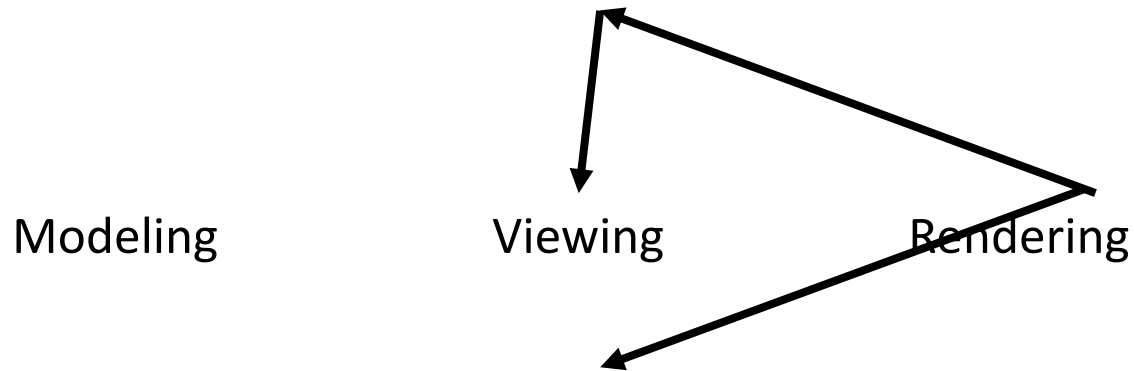


# Where Will It Run?

- jME is 100% Java.
- It depends on a JNI platform.
  - LWJGL is currently the only supported JNI platform.
  - LWJGL runs on Linux, OSX, and Win32.
- Implemented over OpenGL



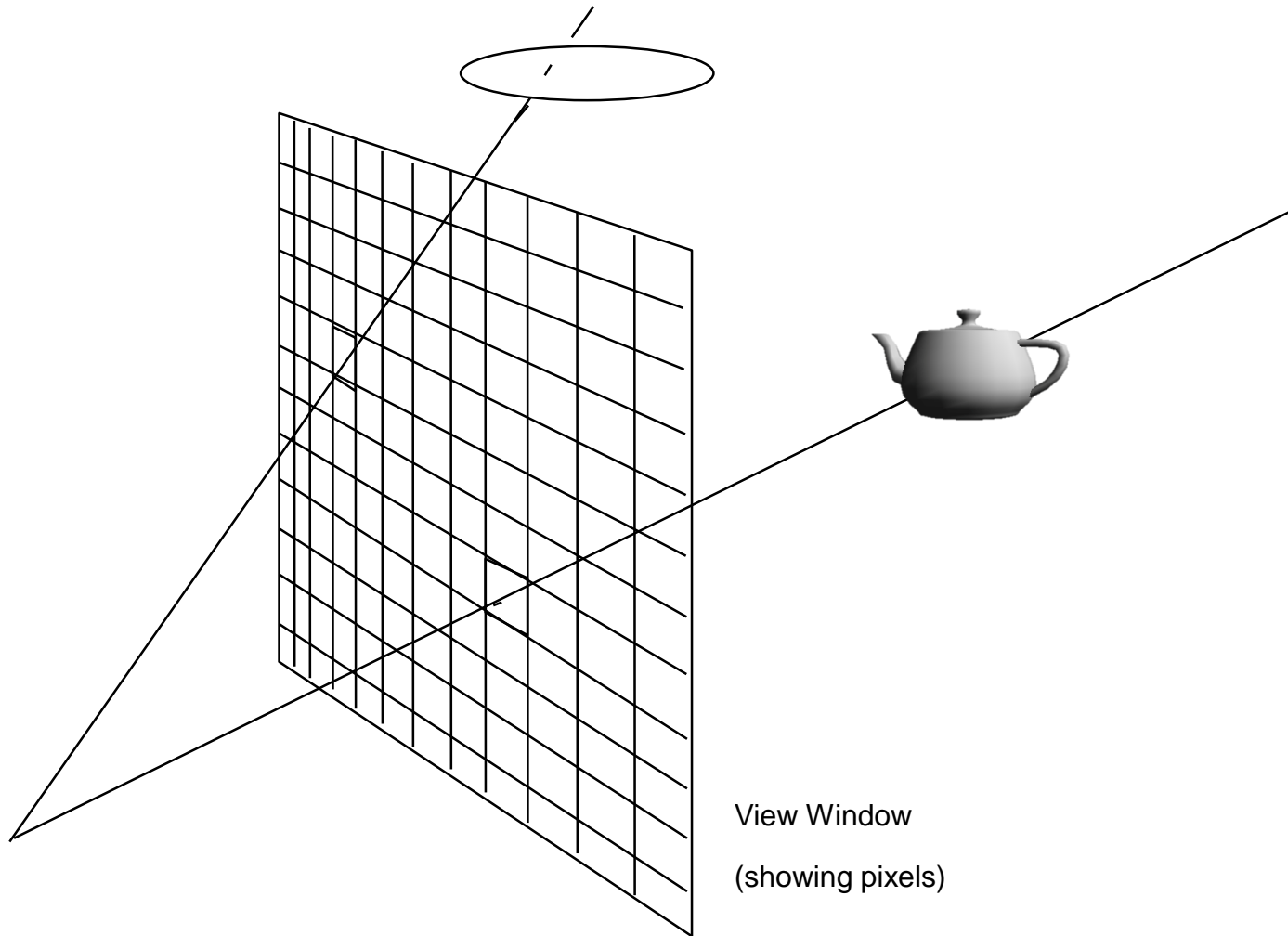
# Image Synthesis



## Separation of **Scene Specification**, **Viewing** and **Rendering**

- **Scene** is modelled independent of any view
- **Views** are unconstrained
- There are many possible **rendering** methods given a scene and a view

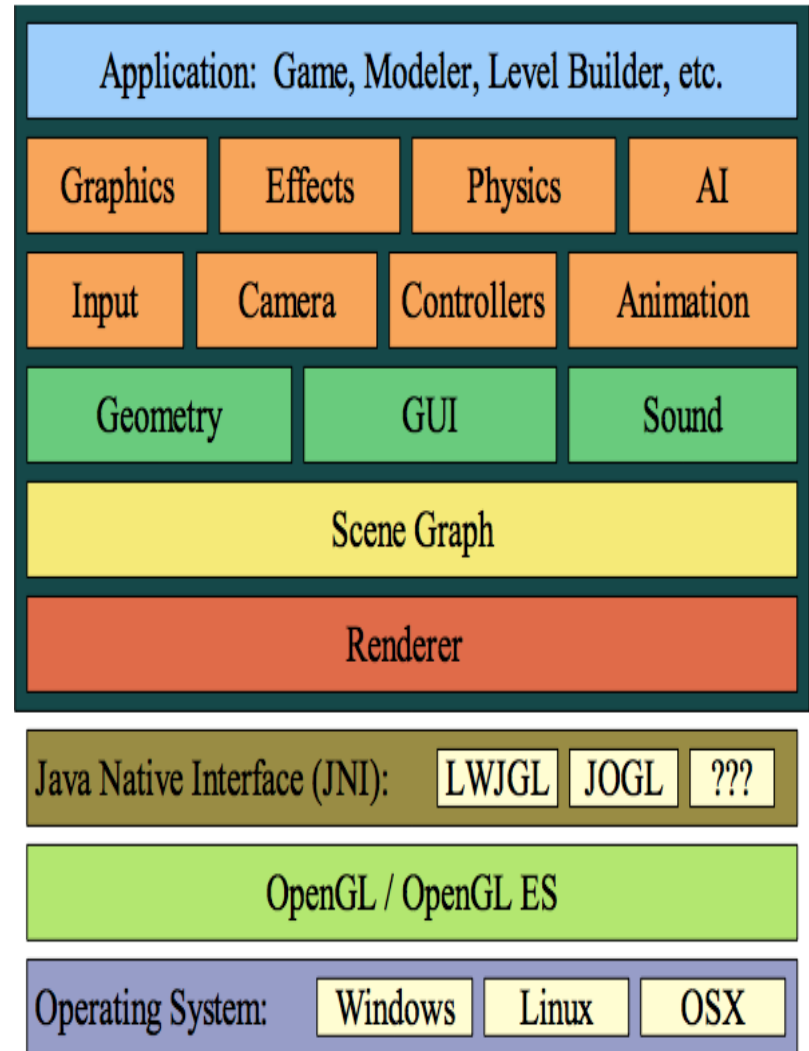
# Model to Screen



# Renderer

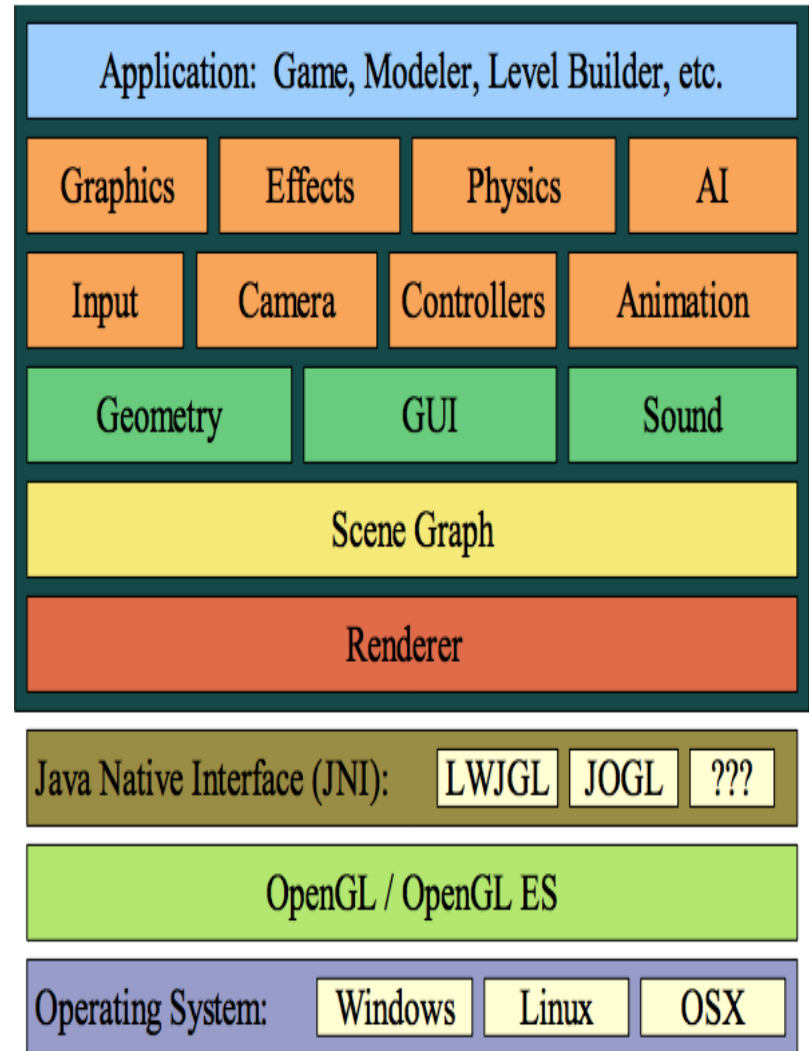
- Transforms geometry from world space to screen space
- Eliminates “hidden” objects
- Draws the transformed scene

More to follow



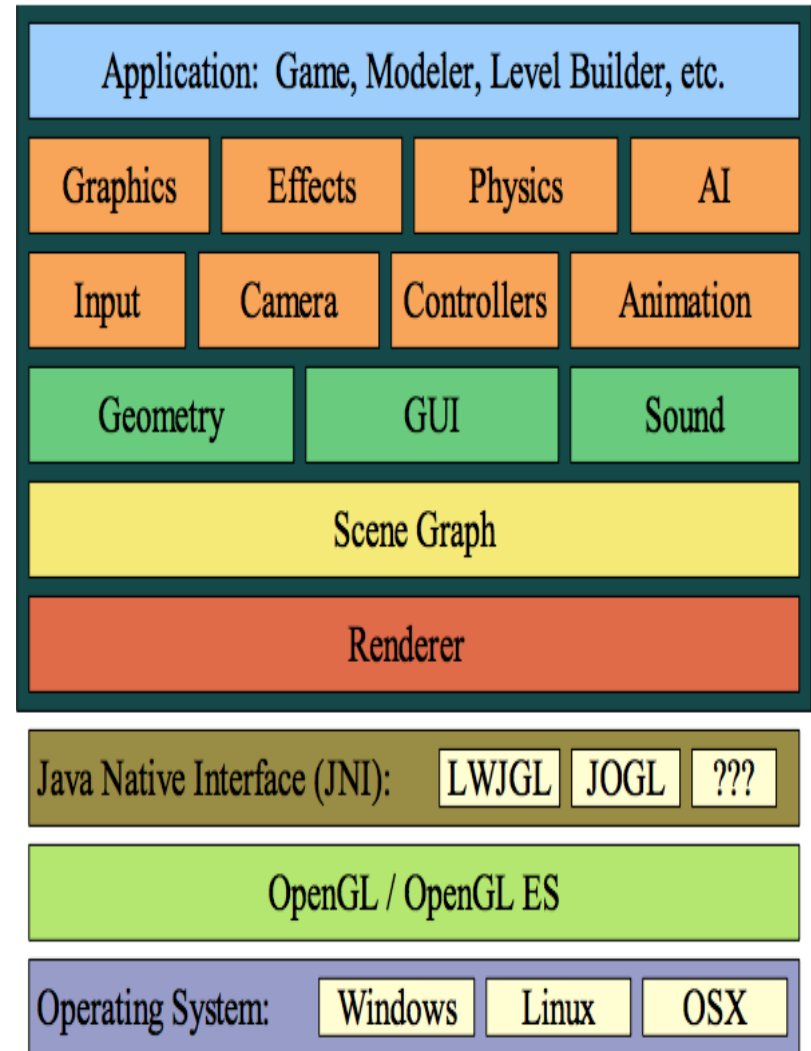
# Scene Graph

- A hierarchical data structure used to group data
  - Simplifies management
  - Groups objects into the same spatial region
  - Facilitates transformations & rotations of compound objects



# Geometry

- Geometry
  - Geometric data for rendering objects
- GUI
  - Widgets
- Sound
  - Similar to renderer
  - 3D effects



# Setting Up jME 3.0

- Download the appropriate version of the jMonkeyEngine SDK
  - <http://jmonkeyengine.org/downloads/>
- Run the installer

(already available in the labs)

- File -> New Project -> BasicGame

# Simplest jME Program

```
package mygame;
```

```
import com.jme3.app.SimpleApplication;
```

```
public class Main extends SimpleApplication {
```

```
    public static void main(String[] args) {
```

```
        Main app = new Main();
```

```
        app.start();
```

```
    }
```



# A Default Blue Box

```
package mygame;
```

```
import com.jme3.app.SimpleApplication;
```

```
import com.jme3.material.Material;
```

```
import com.jme3.math.ColorRGBA;
```

```
import com.jme3.scene.Geometry;
```

```
import com.jme3.scene.shape.Box;
```

```
public void simpleInitApp() {
```

# Let's Run It

## Demo

# Game Loop

```
package mygame;
```

```
import com.jme3.app.SimpleApplication;
```

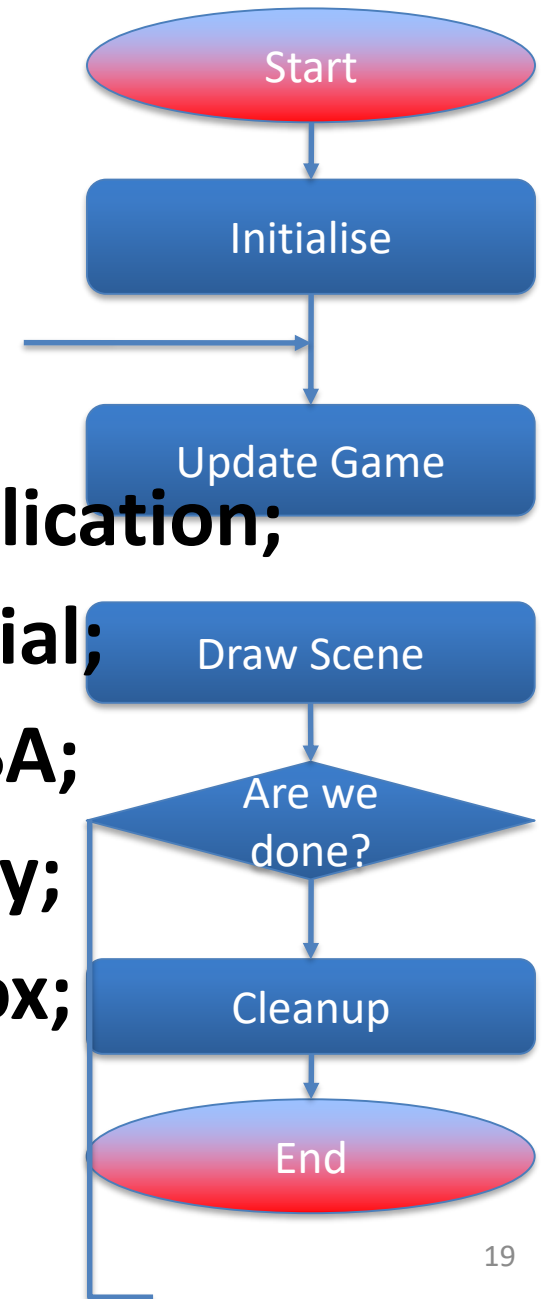
```
import com.jme3.material.Material;
```

```
import com.jme3.math.ColorRGBA;
```

```
import com.jme3.scene.Geometry;
```

```
import com.jme3.scene.shape.Box;
```

```
public void simpleInitApp() {
```

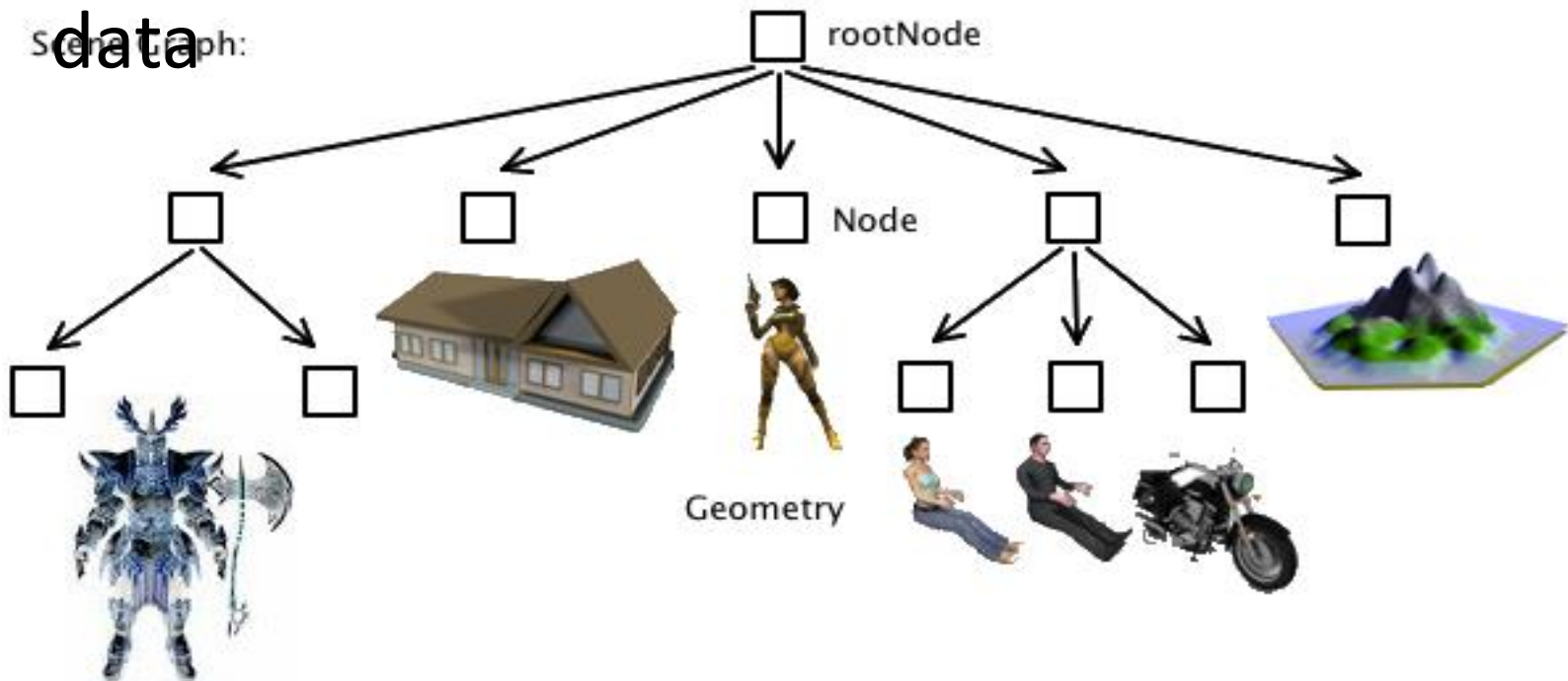


# SimpleApplication Provides

- Options dialog (when you first run it)
  - Can ask for it to be always on
- Input handler
- Standard camera
- A timer to compute the frame rate and provide smooth movements
  
- rootNode

# Scene Graph

- The scene graph represents the 3D world
- Leaf nodes (**Geometry**) represent data
- Internal nodes (**Nodes**) group and manage



# Two Geometries

```
public void simpleInitApp() {  
    Material mat = new Material(assetManager,  
    "Common/MatDefs/Misc/Unshaded.j3md");  
    mat.setColor("Color", ColorRGBA.Blue);  
  
    Box b = new Box(1, 1, 1);  
    Geometry geom = new Geometry("Box", b);  
    geom.setMaterial(mat);  
  
    Sphere s = new Sphere(60, 60, 1.5f);  
    Geometry sgeom = new Geometry("Sphere", s);  
    sgeom.setMaterial(mat);  
  
    rootNode.attachChild(geom);  
    rootNode.attachChild(sgeom);  
}
```

# Graphical Model

- Items arranged spatially (grouped together)
  - Placing something (e.g. a light) in a branch affects all branch elements
- A node is a reference point to its children
  - Simplifies rendering
  - Simplifies manipulation
- Simplifies importing models

# Rendering Scene Graph

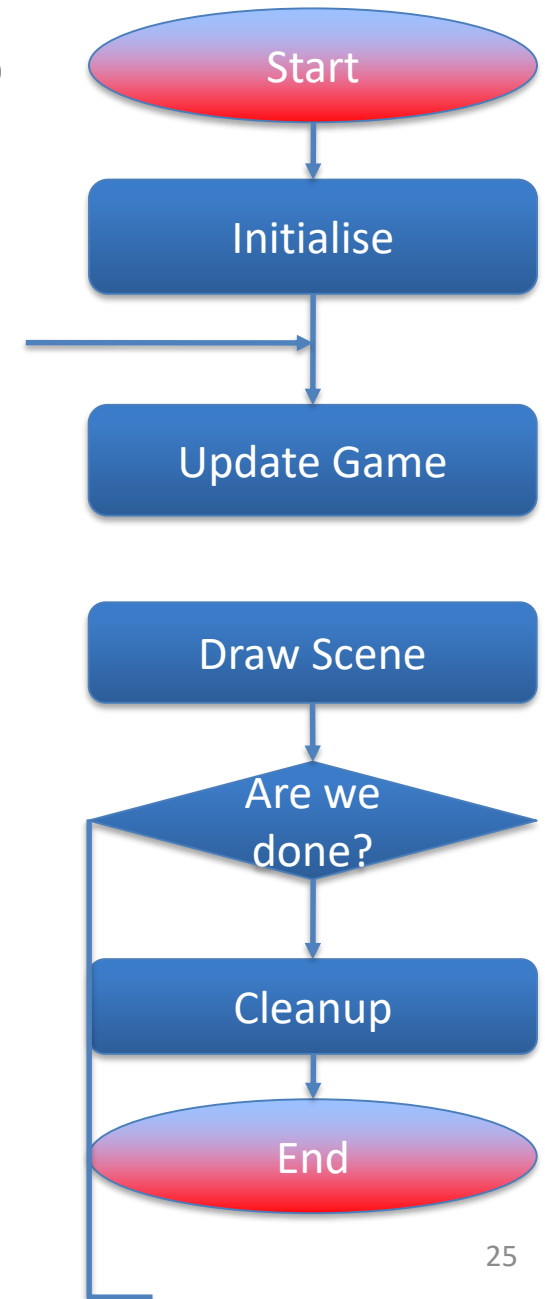
- Every node (Nodes and Geometry) defines
  - Transform(ation)s
    - orientation, location and scale
  - BoundingVolume
    - An area containing all sub-nodes
  - Render state
    - Defines how geometry is displayed



# Meaningful Game Loop

```
...  
protected Geometry geom;  
...  
public void simpleUpdate(float tpf) {  
    geom.move (0.001f, 0, 0);  
}
```

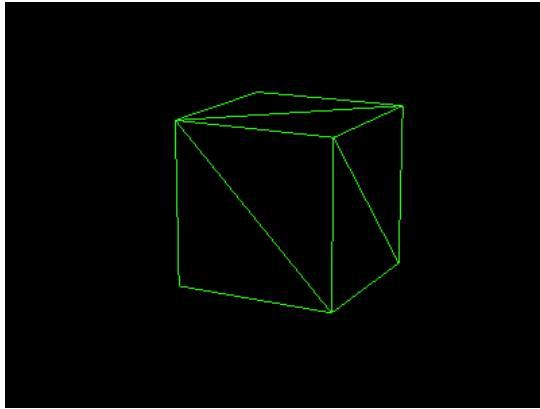
Changes the scene graph



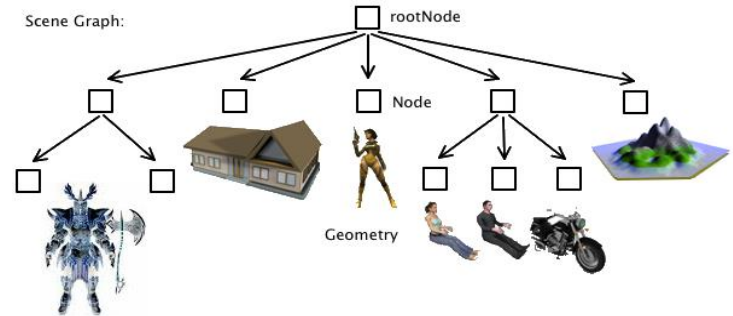
# Let's Run It

## Demo

# Meshes and Geometries



A collection of *polygons*  
that can be drawn



Everything that is rendered

Box mesh = new Box(1, 1, 1);

Geometry geom = new Geometry("Box", mesh);

But *how* is it rendered

# Meshes, Geometries and Materials

- All Geometries must have *Materials* that defines colour or texture.
- Each Material is based on a Material Definition file (.j3md)
  - Lighting.j3md, Unshaded.j3md

All Materials (except "Unshaded" ones) are **invisible** without a light source.

# Example

```
public void simpleInitApp() {  
    Box b = new Box(1, 1, 1);  
    geom = new Geometry("Box",  
b);  
    rootNode.attachChild(geom);  
    Material mat = new  
Material(assetManager,  
"Common/MatDefs/Light/Lighting.j  
3md");
```

# 3D Models and Games

- While it is possible to specify the geometry based on basic shapes (we do it), most games *import* scene graphs from a 3D modelling tool
  - Maya
  - 3D Max
  - Blender
  - ...

# Summary

- jMonkeyEngine is a simple yet powerful Java game engine
- Basic shapes can be combined in a scene graph to create a 3D model
- We need some Maths to manipulate entities