

Introduction to the jMonkey Engine



A presentation by Gregg Patton

Contents

- An Overview of jME
- The History of jME
- Anatomy of the jME Project
- The jME Architecture
- Opportunities to Participate
- Gregg's Motivation
- The Future of jME
- References
- Links to Web Sites

An Overview of jME

- jME is a Java based 3D game engine.
- jME is open source, with a BSD license.
- jME can be used to create interactive, high performance 2D and 3D graphics systems.
- jME provides a scene graph architecture.
- jME provides real time rendering based on OpenGL.
- jME is intended to be a complete game creation solution.

jME Demonstration



Vertex Colors

The History of jME

- Project Founder
 - Mark Powel, aka - “mojomonk”.
- Mark's Motivation
 - Interest in game engines.
 - Curious about 3D graphics in Java.
 - No major game engines written in Java at the time.
- Notable Dates
 - jME project began in June 2003.

The History of jME (cont.)

- Notable Dates (cont.)
 - Rewrite in September 2003 using a scene graph architecture because:
 - The first jME was not robust enough.
 - It didn't allow for optimizations in the rendering pipeline.
 - It didn't allow for pluggable renderers.
 - jME became part of dev.java.net middleware in October of 2003.
 - Mark selected java.net because SourceForge CVS was down at the time.

The History of jME (cont.)

- Why did Mark choose open source?
 - The learning experience.
 - The enjoyment factor.
 - It's fun to work with others.
 - Needed help.
 - The open source community can provide the help.
 - Resume building.
 - Gives prospective employers something to look at.

The History of jME (cont.)

- Why did Mark choose BSD licensing?
 - It's the least constraining type of open source license without making it public domain.
 - It gets the authors' name on the source.
 - It let users do whatever they want to with it.



Anatomy of the jME Project

- Main web site
 - <http://www.mojomonkeycoding.com/>
 - jME discussion forums
 - <http://www.mojomonkeycoding.com/jmeforum/>
 - A few of the forums:
 - Graphics
 - Sound
 - GUI
 - Effects
 - etc.

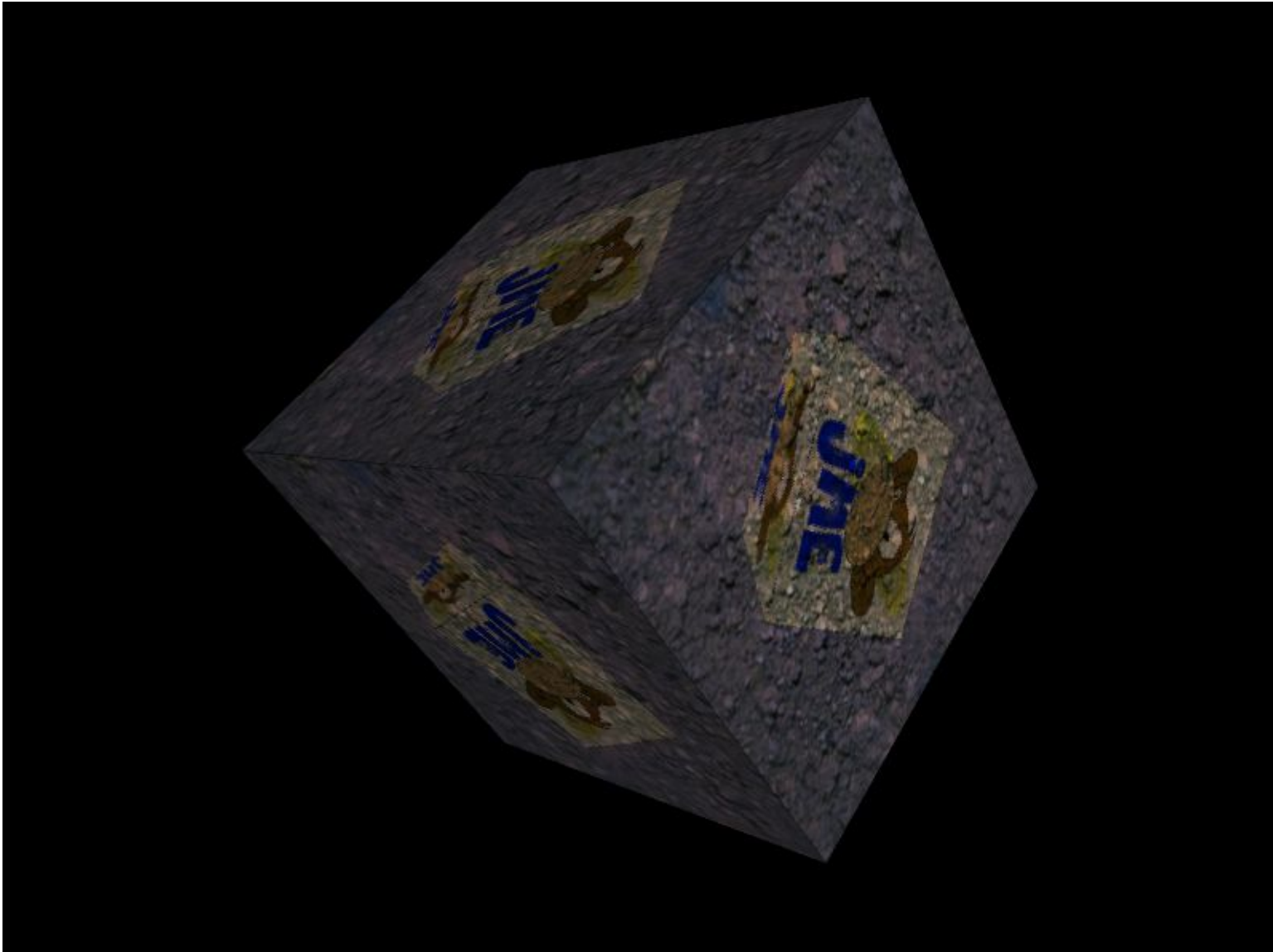
Anatomy of the jME Project (cont.)

- jME on java.net
 - <https://jme.dev.java.net/>
 - CVS repository
 - Release announcements





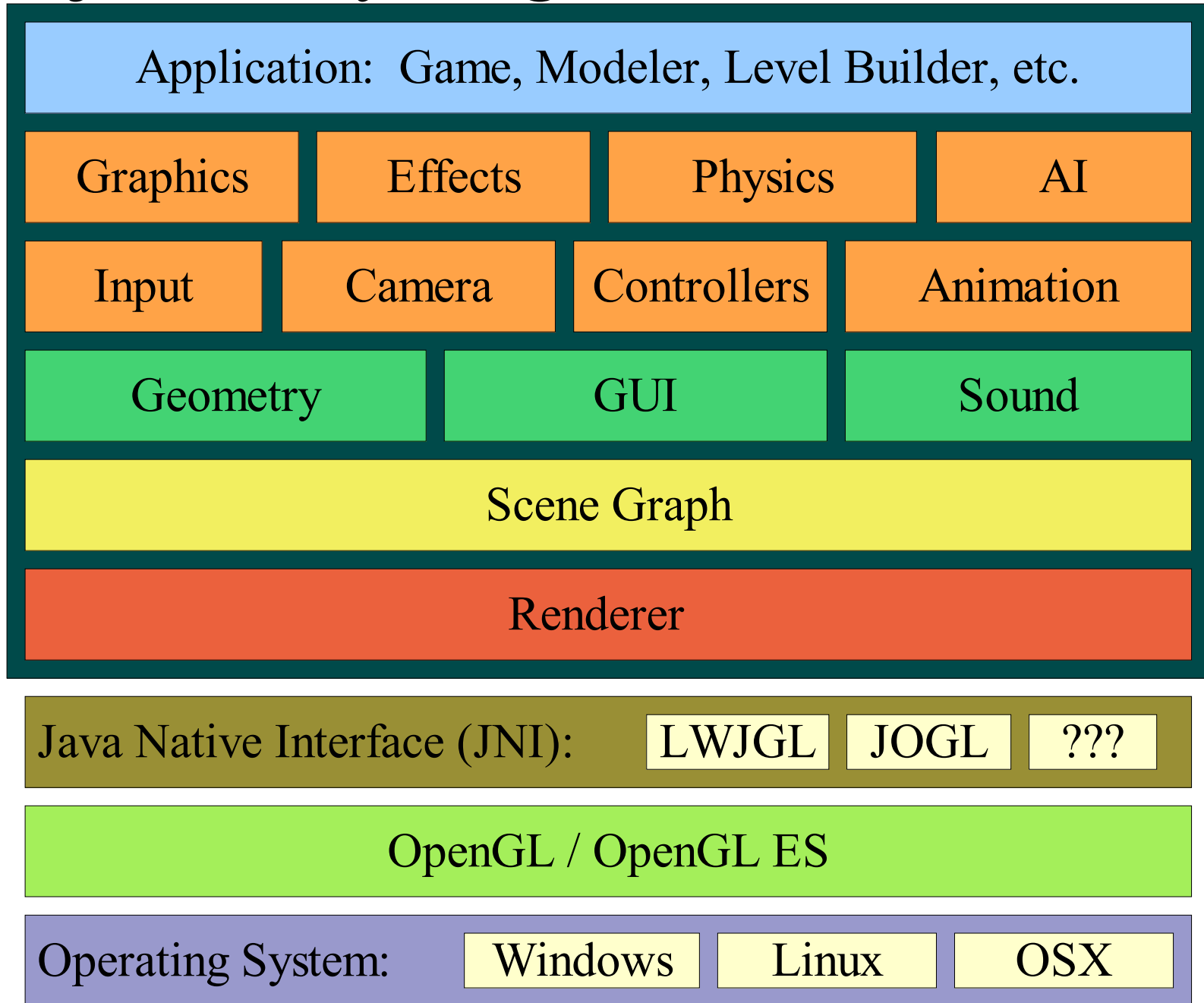
jME Demonstration



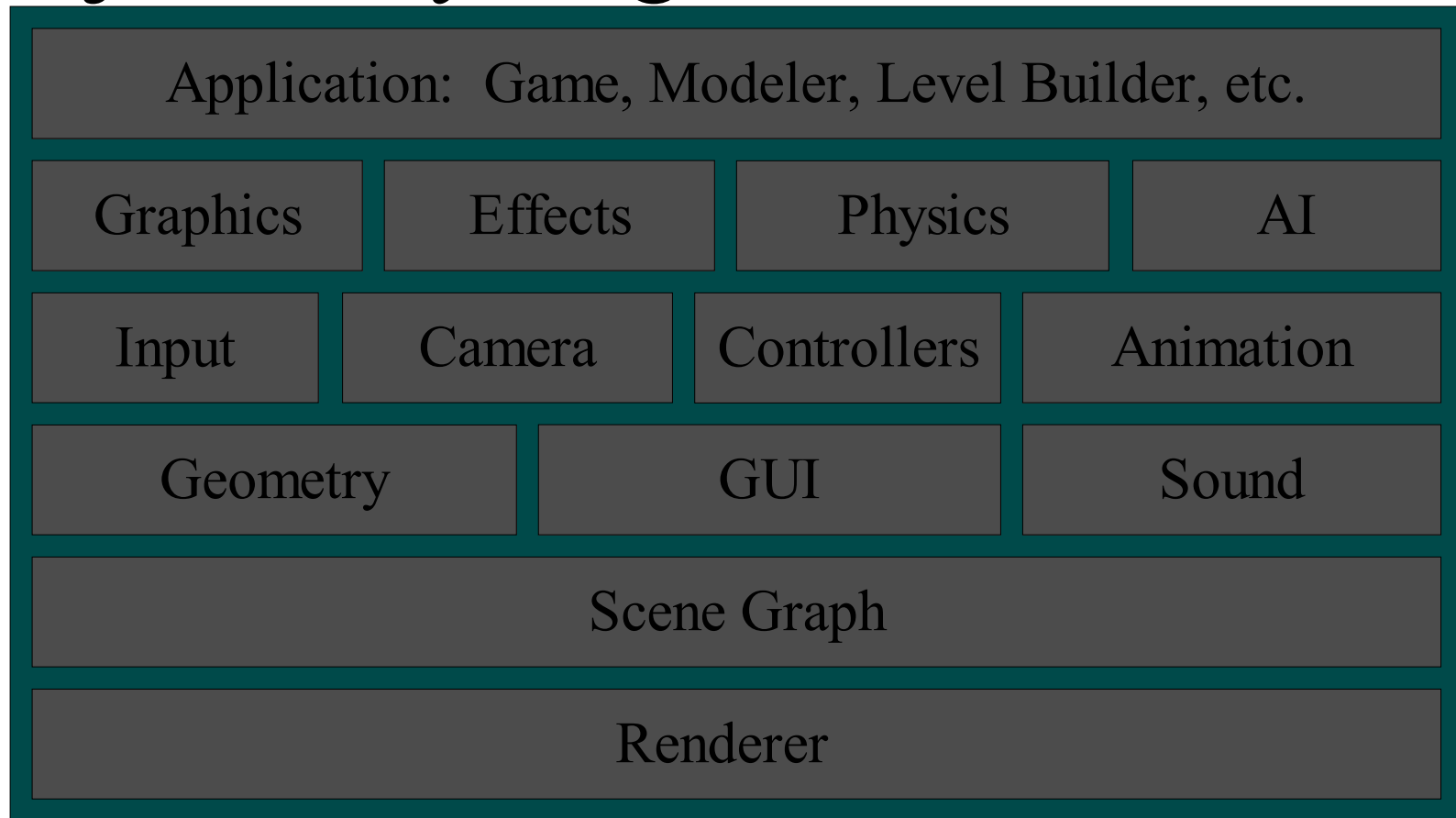
Render to Texture



jMonkey Engine Architecture



jMonkey Engine Architecture



Java Native Interface (JNI):

LWJGL

JOGL

???

OpenGL / OpenGL ES

Operating System:

Windows

Linux

OSX

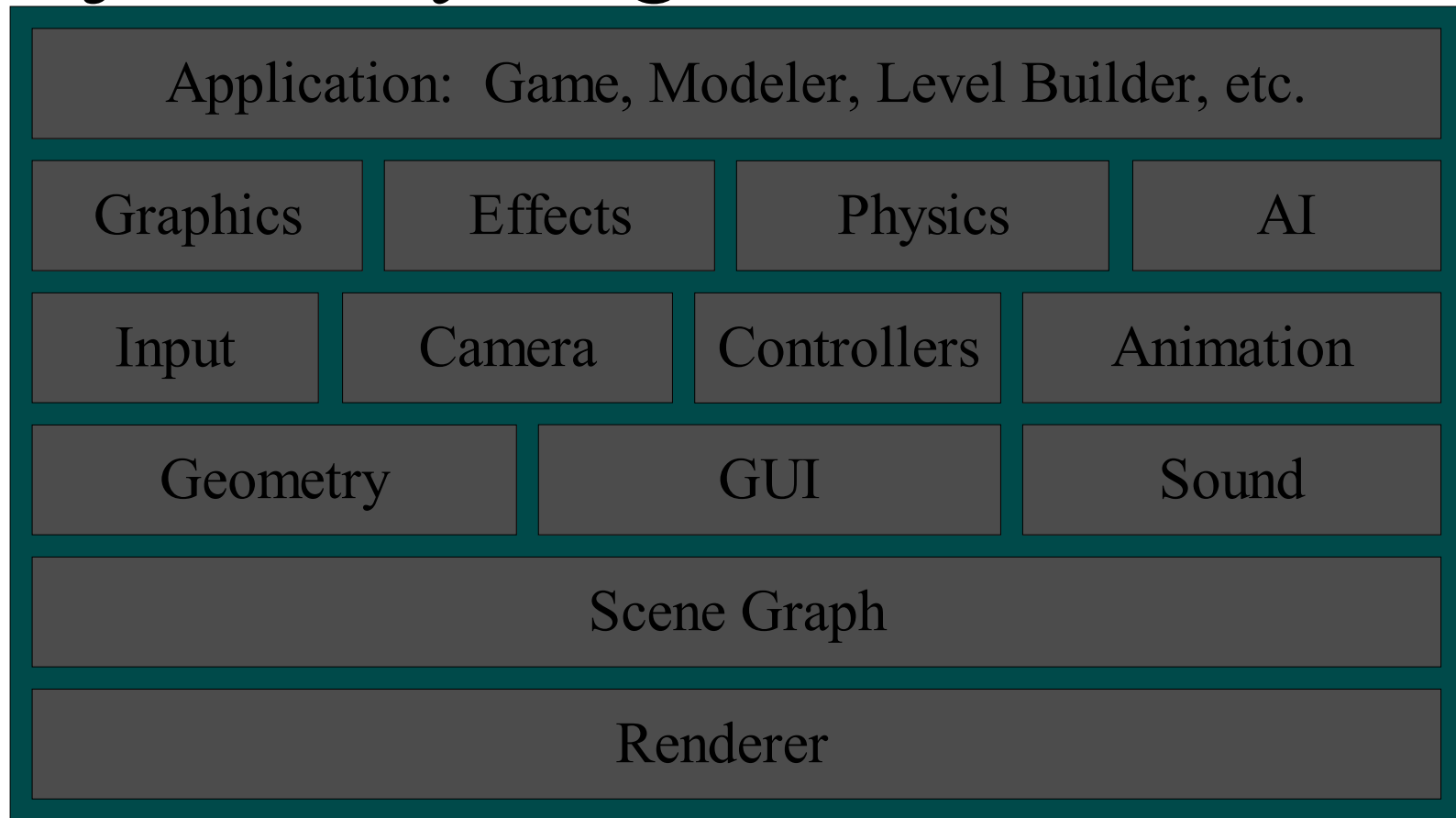


jME Supported Operating Systems

- 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.
- jME Verified on:
 - Windows
 - Linux?
 - OSX?



jMonkey Engine Architecture



Java Native Interface (JNI):

LWJGL

JOGL

???

OpenGL / OpenGL ES

Operating System:

Windows

Linux

OSX



OpenGL

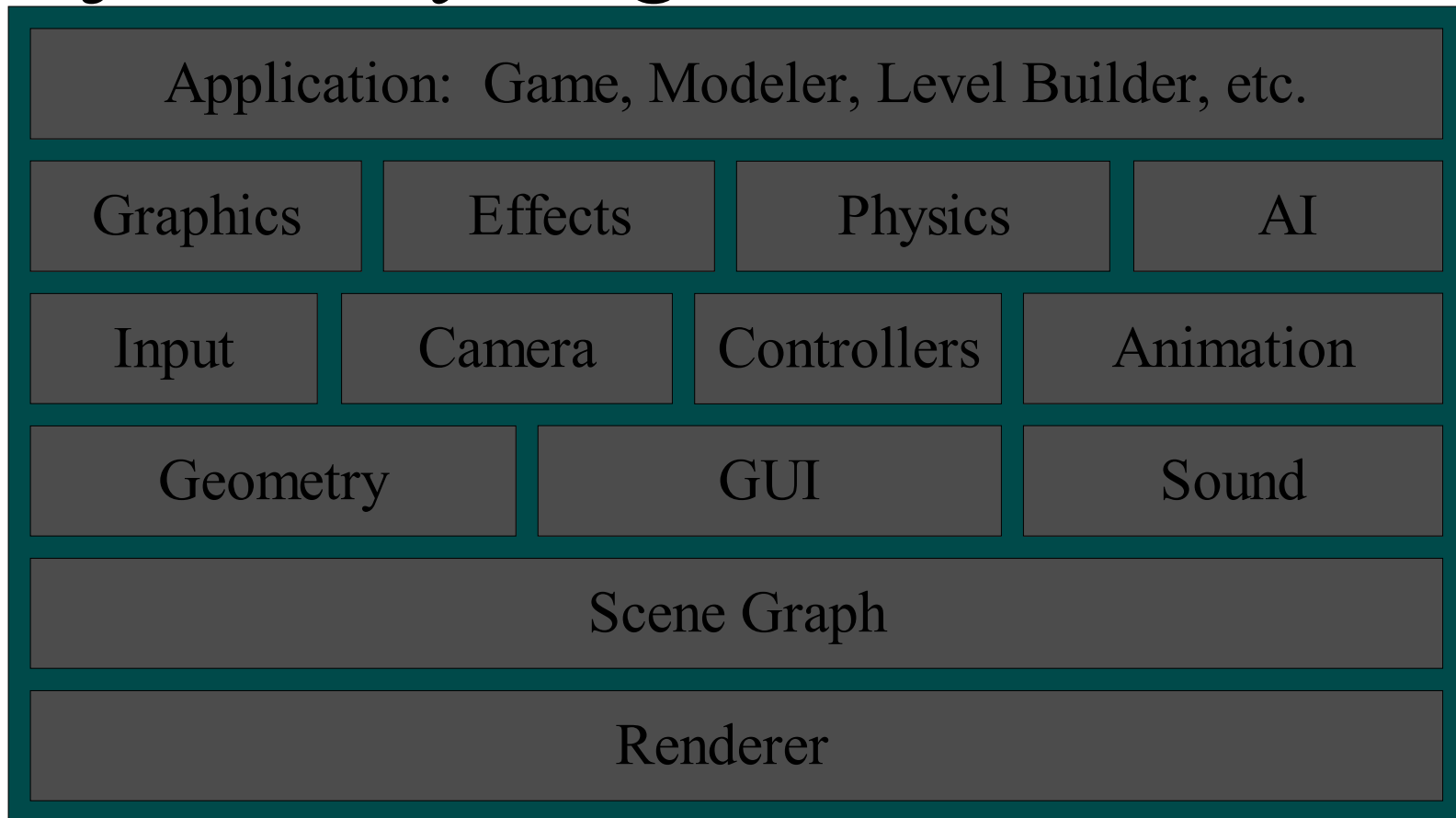
- A cross platform API for high performance 2D and 3D graphics.
- See <http://www.opengl.org/> for more information.

JME
MONKEY ENGINE

OpenGL ES

- Brings advanced 2D/3D graphics capabilities to a wide variety of mobile devices, appliances and embedded displays.
- The LWJGL project plans to support OpenGL ES meaning jME will also be able to support it.
- See <http://www.khronos.org/index.html> for more information.

jMonkey Engine Architecture



Java Native Interface (JNI):

LWJGL

JOGL

???

OpenGL / OpenGL ES

Operating System:

Windows

Linux

OSX

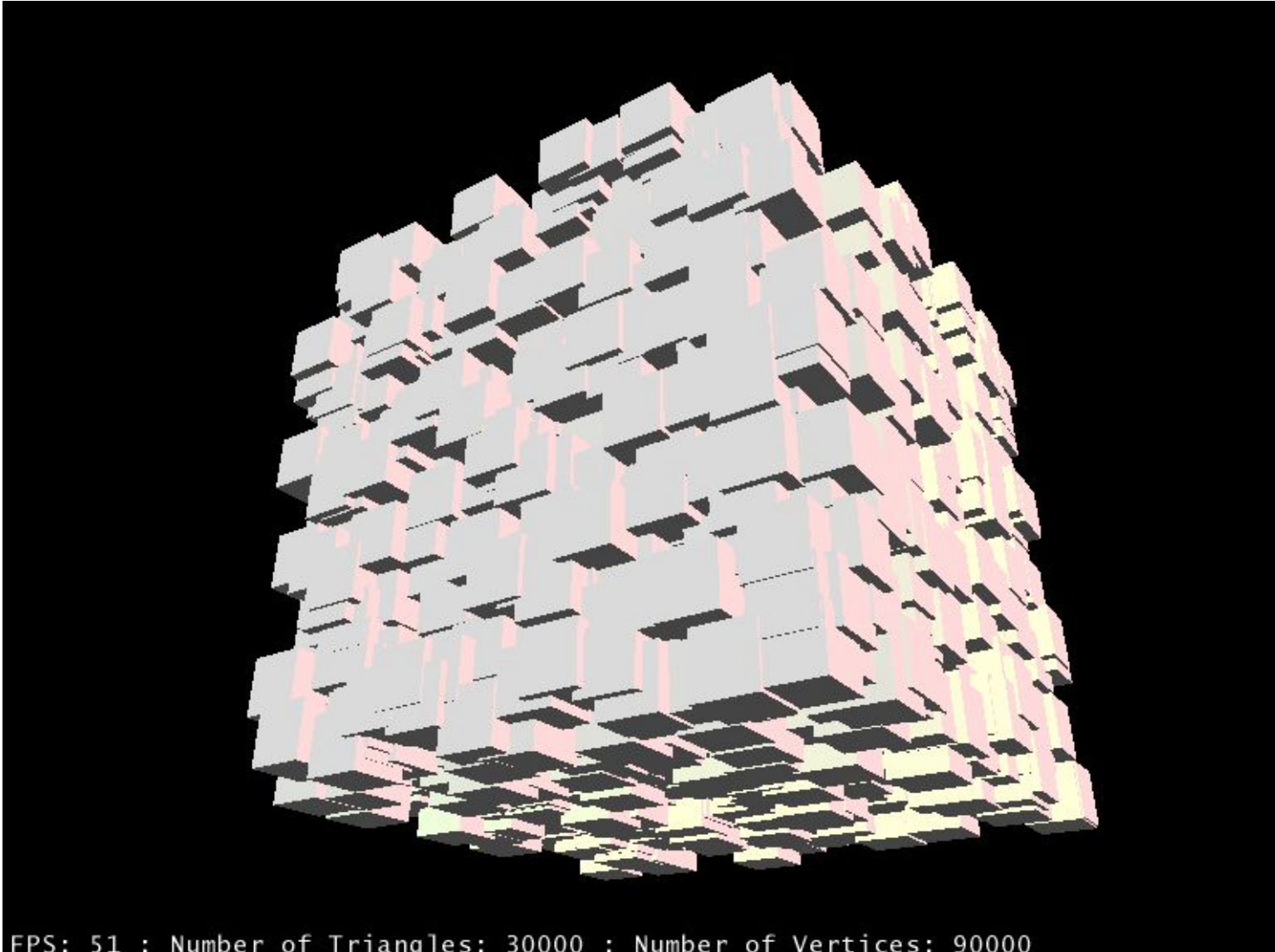


Java Native Interface to OpenGL

- Currently supported JNI implementations
 - LWJGL - Light Weight Java Graphics Library
 - See <http://www.lwjgl.org/> for more info.
- Future JNI support
 - JOGL
 - See <https://jogl.dev.java.net/> for more info.
 - ???

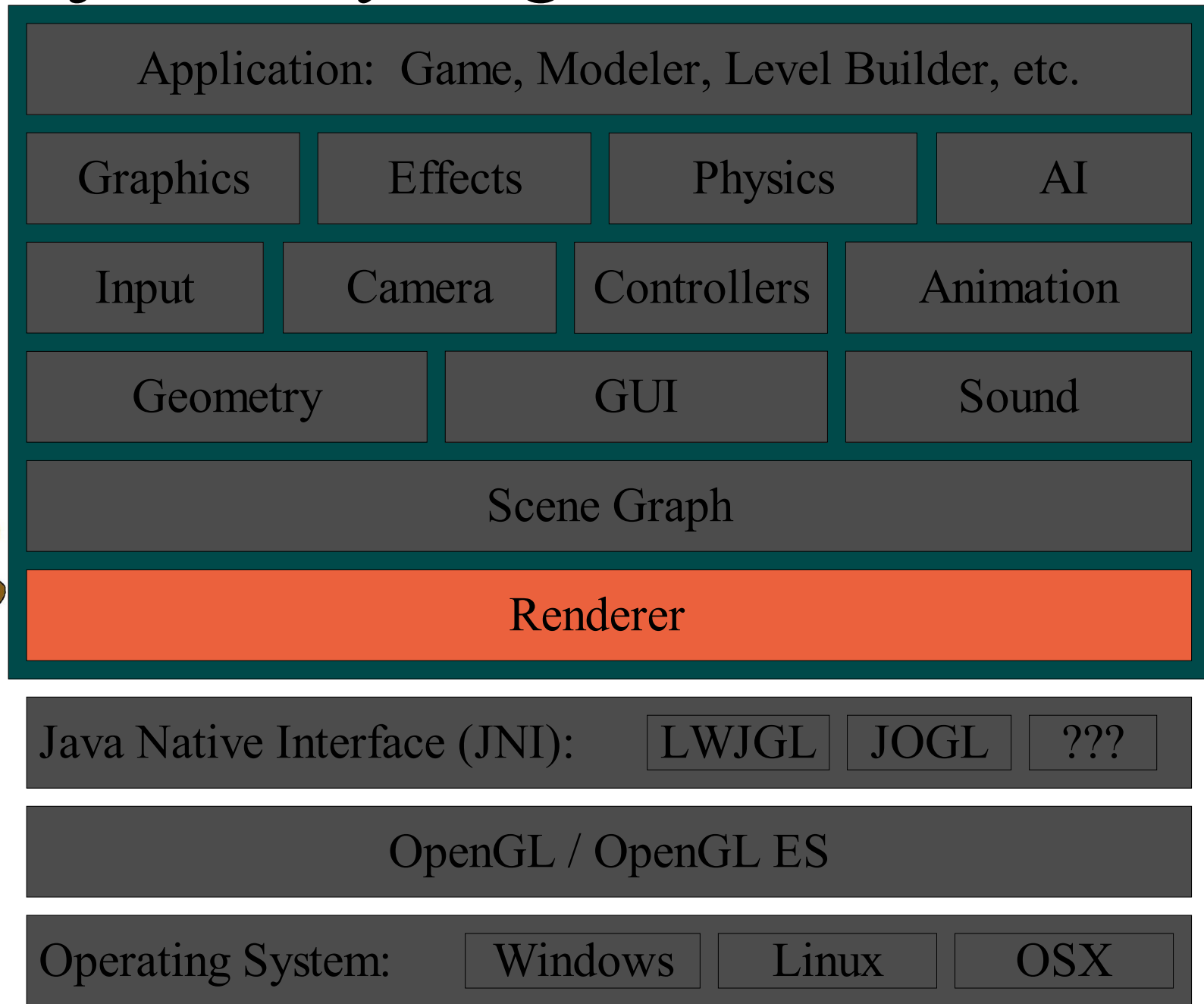


jME Demonstration



2,500 Box Test

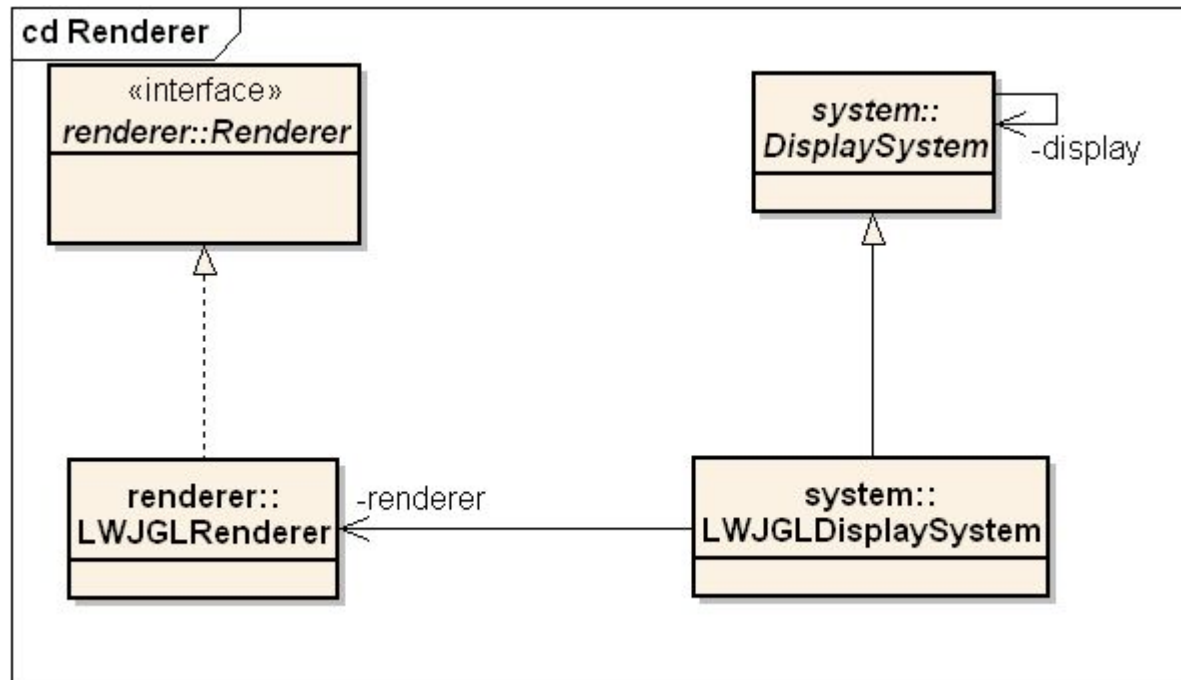
jMonkey Engine Architecture



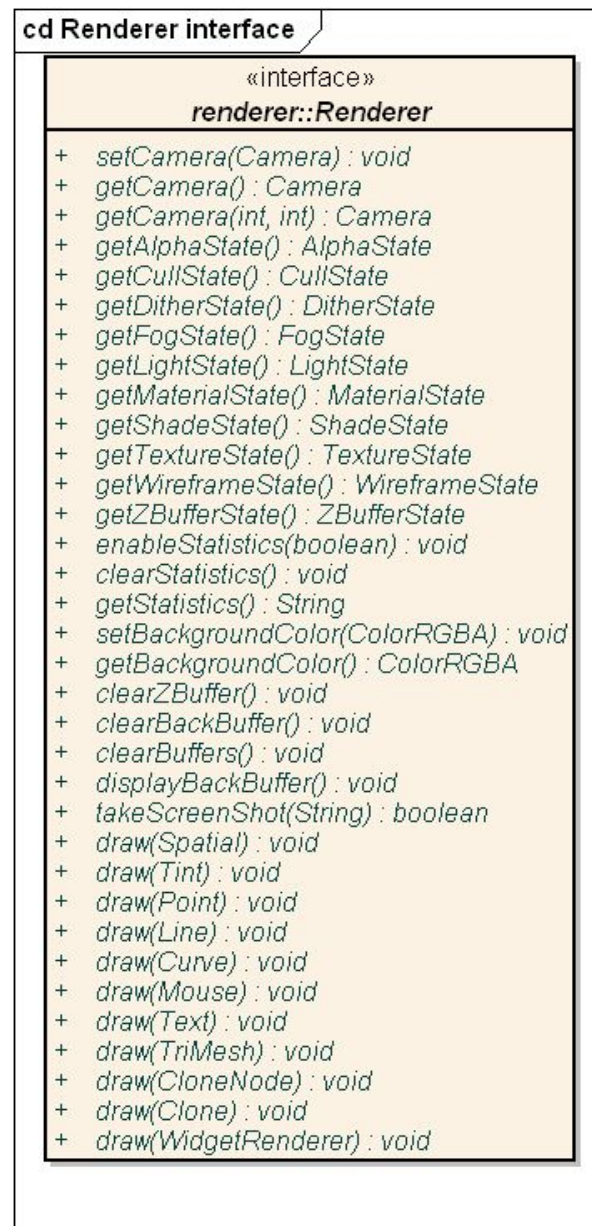
The jME Renderer

- Responsibilities of the Renderer
 - Transforms scene data from world space to screen space.
 - Eliminates parts of the scene graph from the rendering process to enhance performance by:
 - Culling
 - A process that determines if any part of an object is visible in the scene from the observer's perspective. If it's not, it won't be rendered.
 - Clipping
 - A process that splits an object into smaller pieces so the non-visible pieces can be discarded from the rendering process.
 - Draws the transformed scene to the screen.
 - Traverses the scene graph, rendering the visible portions of it, applying rendering states as it goes.

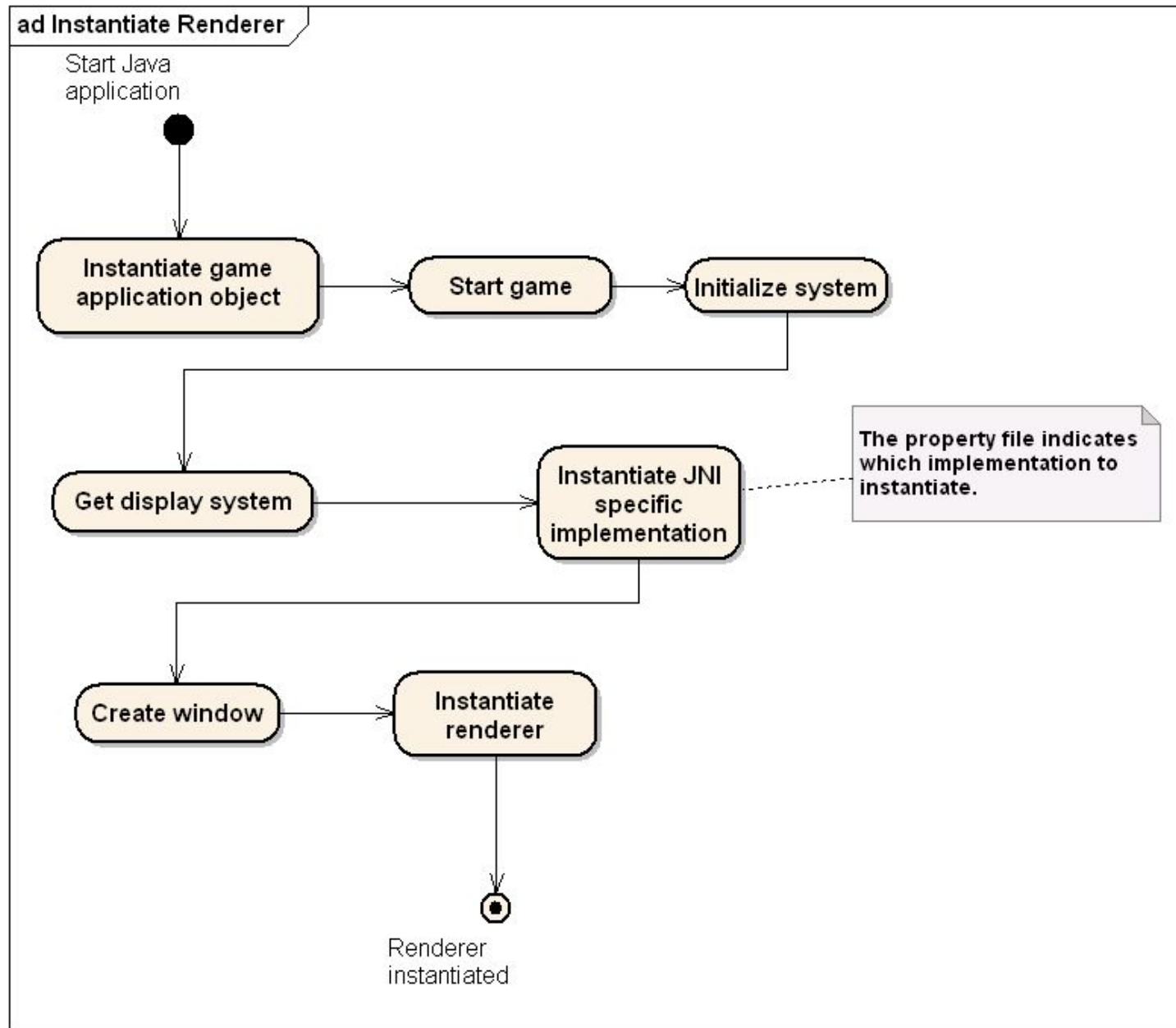
The jME Renderer Implementation



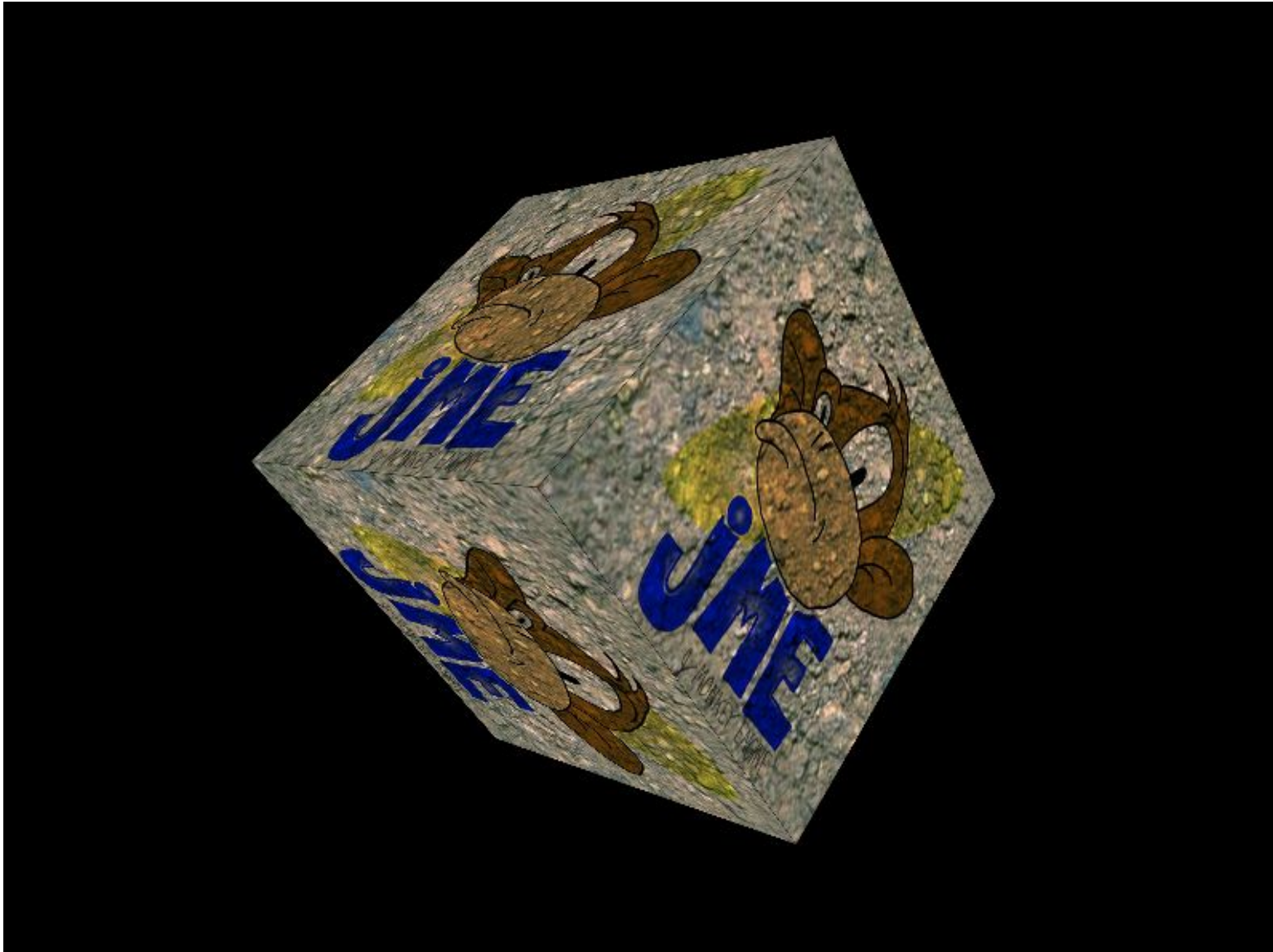
The jME Renderer Implementation (cont.)



The jME Renderer Implementation (cont.)

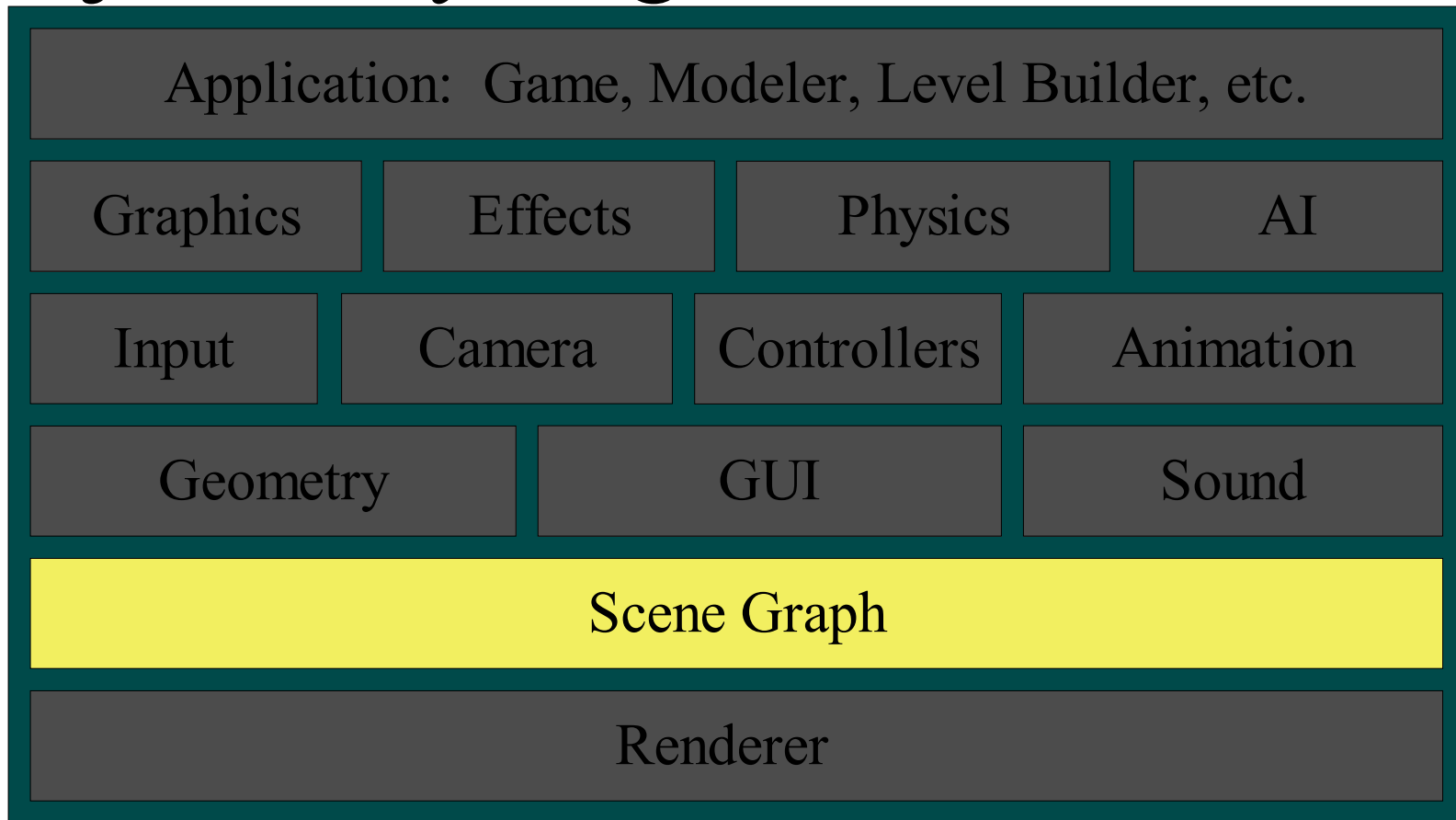


jME Demonstration



Multitexturing

jMonkey Engine Architecture



Java Native Interface (JNI):

LWJGL

JOGL

???

OpenGL / OpenGL ES

Operating System:

Windows

Linux

OSX

The jME Scene Graph

- What is a Scene Graph?
 - A hierarchical data structure used to group data.
 - Aka tree.
 - Comprised of:
 - Parent nodes containing any number of child nodes.
 - Child nodes containing one parent node.
 - The root node which has no parent.
 - Leaf nodes containing geometrical data.
 - Internal nodes to manage grouping.

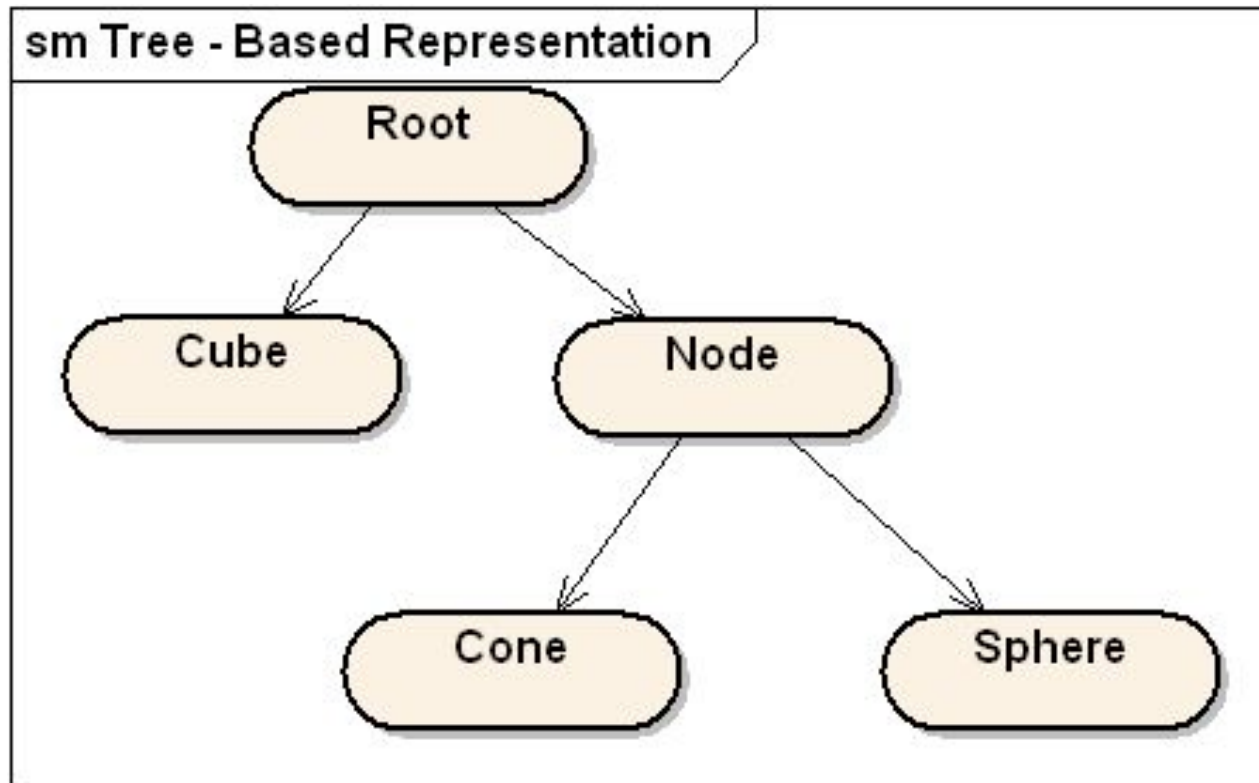
The jME Scene Graph (cont.)

- Benefits of a Scene Graph in a game engine.
 - Simplifies management of global attributes.
 - Setting a light to illuminate a single subtree in the graph.
 - Allows grouping of objects in the same spatial region.
 - Helps to quickly eliminate regions of the scene that aren't visible and don't need to be rendered.
 - Facilitates orientation of hierarchical models like humanoid characters.
 - The torso determines the position of the head, arms and legs. The arms determine the location of the hands. Etc...
 - Aids in persisting the state of a game.
 - The root node can be told to save itself and it's descendants can save themselves recursively

The jME Scene Graph (cont.)

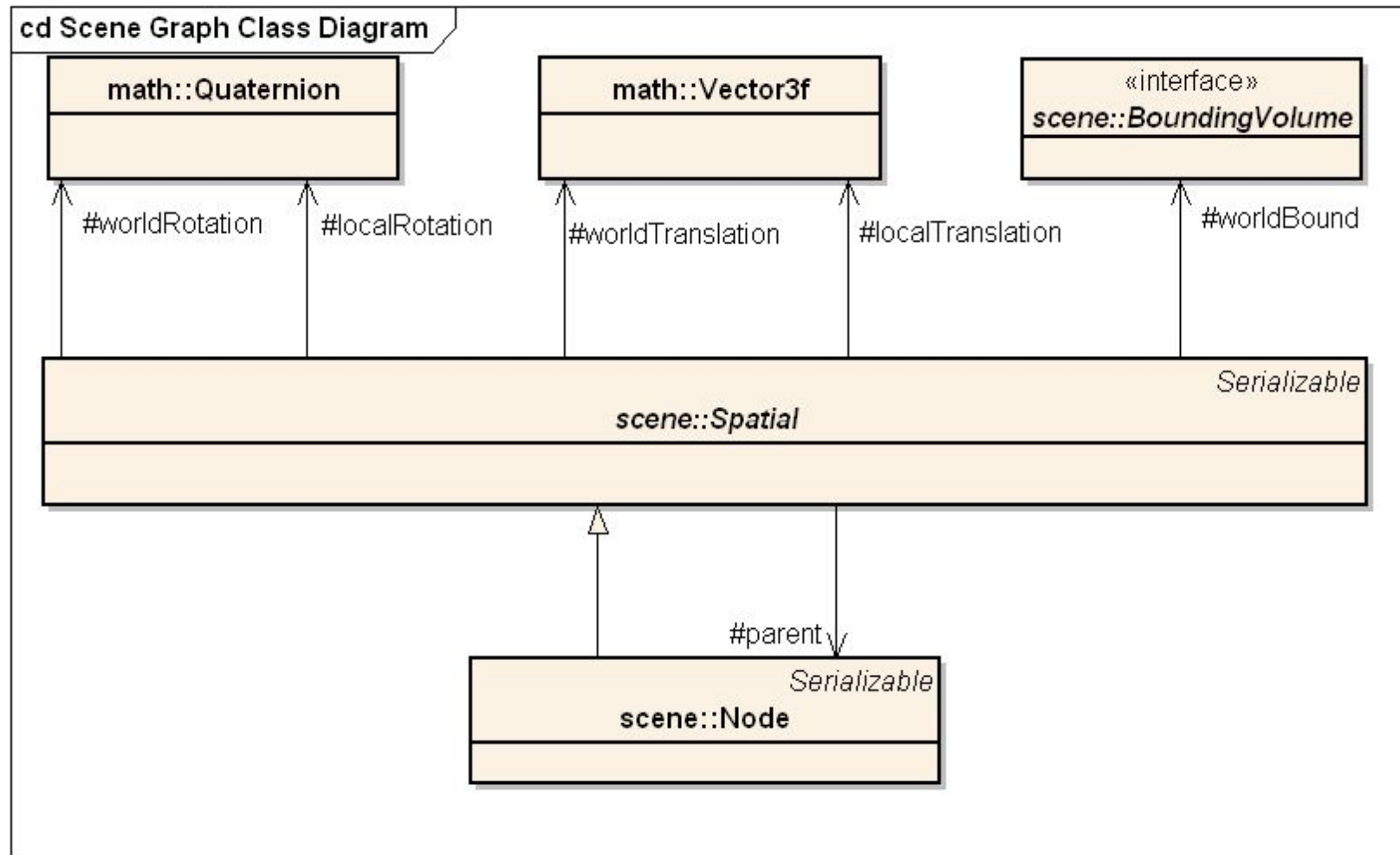
- A tree based representation that provides:
 - Nodes to maintain spatial and semantic information.
 - Transforms
 - Used to position, orient, and size objects.
 - Bounding volumes
 - Used for hierarchal culling and intersection testing.
 - Render state
 - Used to set up the renderer to draw objects.
 - Animation state
 - Used to represent time varying node data.

The jME Scene Graph (cont.)

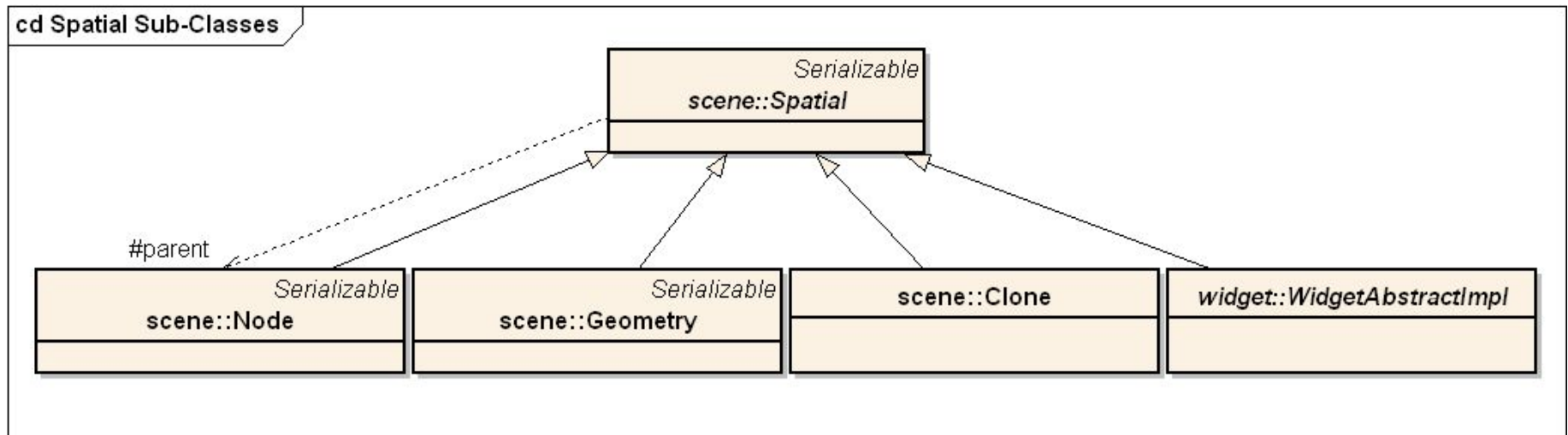




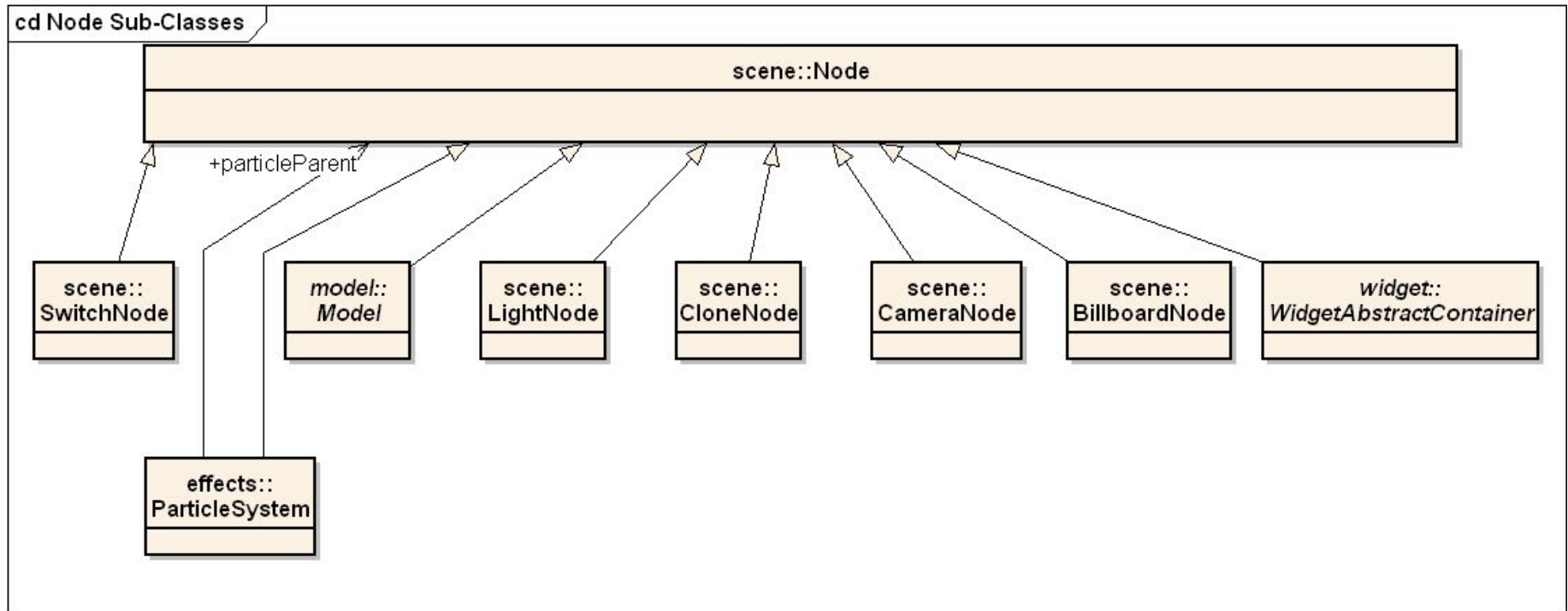
The jME Scene Graph Implementation



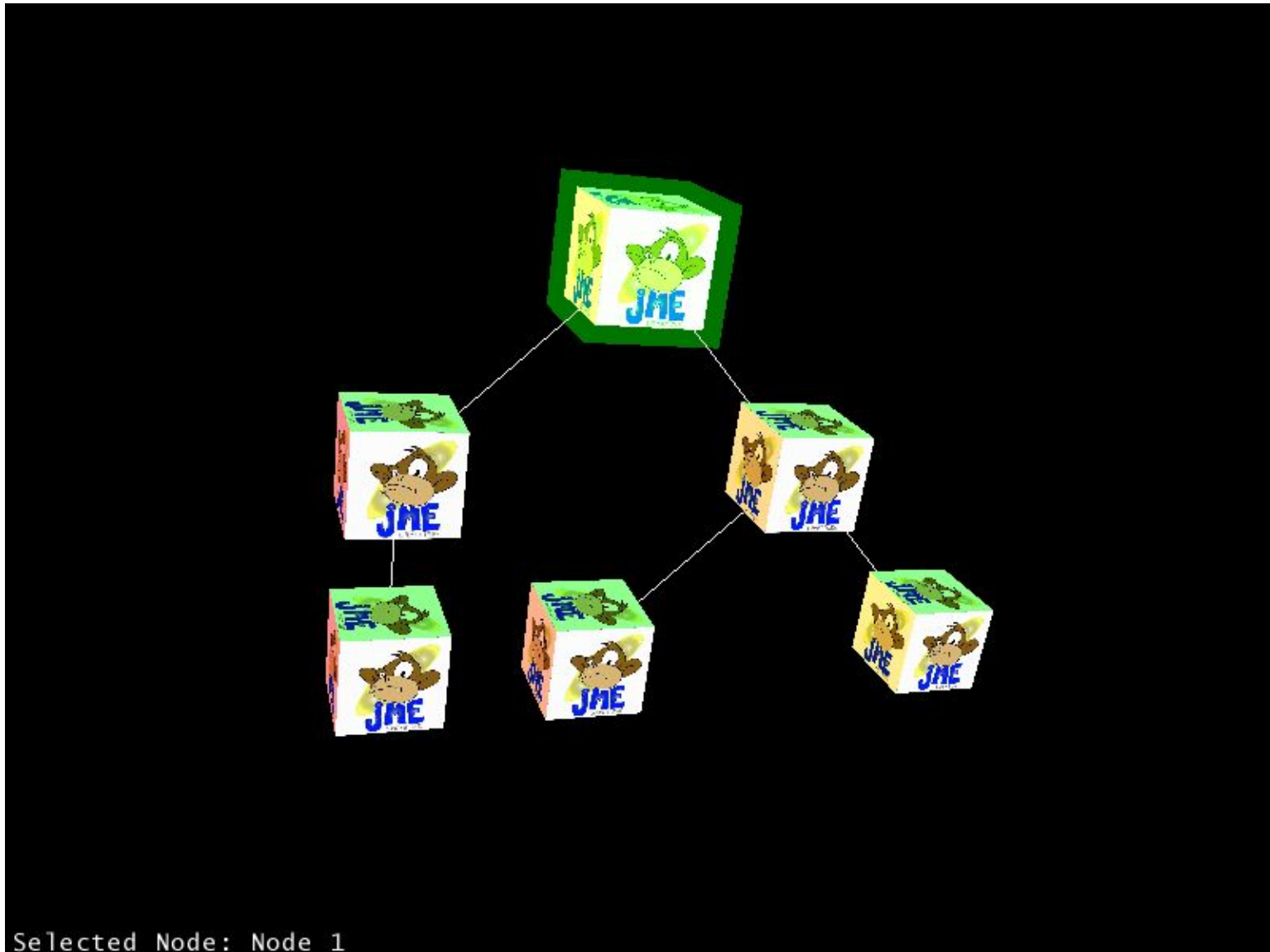
The jME Scene Graph Implementation (cont.)



The jME Scene Graph Implementation (cont.)



jME Demonstration



Test Scene Graph

jMonkey Engine Architecture

Application: Game, Modeler, Level Builder, etc.

Graphics

Effects

Physics

AI

Input

Camera

Controllers

Animation

Geometry

GUI

Sound

Scene Graph

Renderer

Java Native Interface (JNI):

LWJGL

JOGL

???

OpenGL / OpenGL ES

Operating System:

Windows

Linux

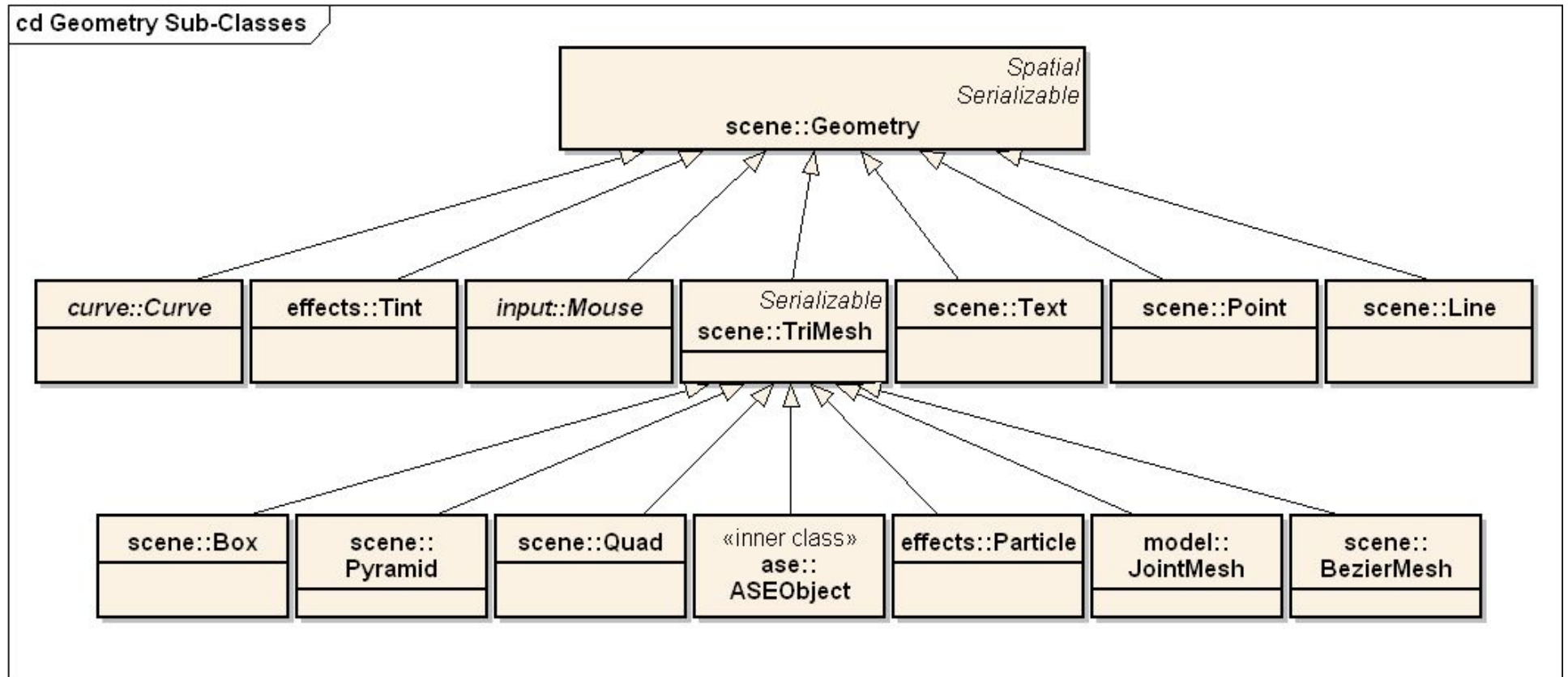
OSX



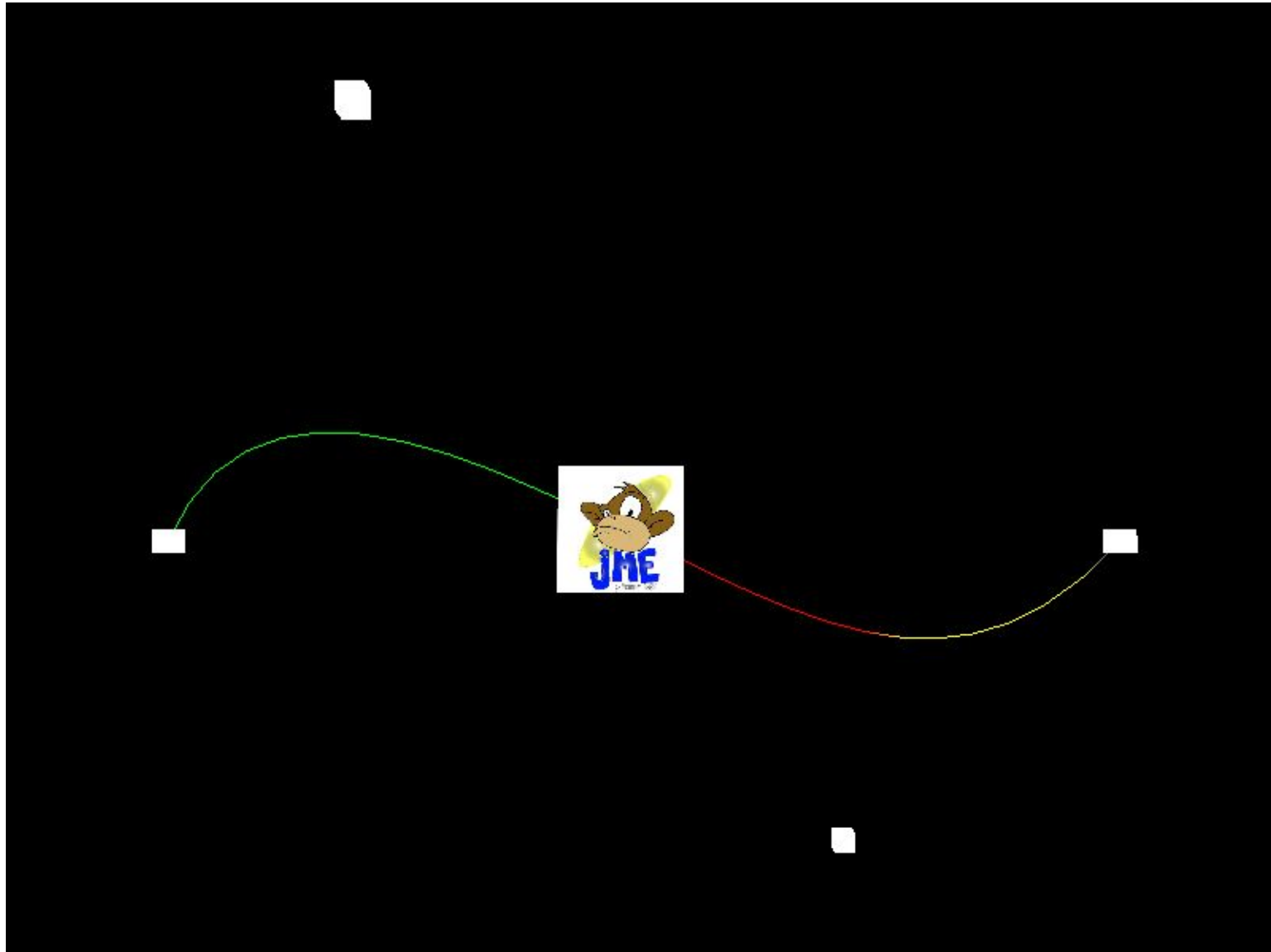
jME Geometry

- Geometry defines a leaf node of the scene graph.
- Contains geometric data for rendering objects.
- Manages all rendering information, including:
 - Different rendering states.
 - Data for a model.

jME Geometry Implementation (cont.)



jME Demonstration



Bezier Curve Test

The jME GUI

- Layout Managers
 - Modeled after `java.awt.LayoutManager`.
 - Consists of two Java interfaces.
 - `WidgetLayoutManager`
 - `WidgetLayoutManager2`
 - Implementations:
 - `WidgetAbsoluteLayout`
 - `WidgetBorderLayout`
 - `WidgetFlowLayout`
 - `WidgetGridLayout`

The jME GUI (cont.)

- Widgets
 - All Widgets implement the Widget interface.
 - Widget communication is handled by implementing the `java.util Observer/Observable` interfaces.
 - An object implements the Observer interface and then gets added to a Widget's Observer list for a particular event. i.e. mouse activity, value changes, etc. When a Widget event occurs all of the Observers in the list for that event get notified.
 - `WidgetAbstractImpl`
 - The default Widget implementation.
 - Also extends `Spatial` so Widgets can be in the Scene Graph.

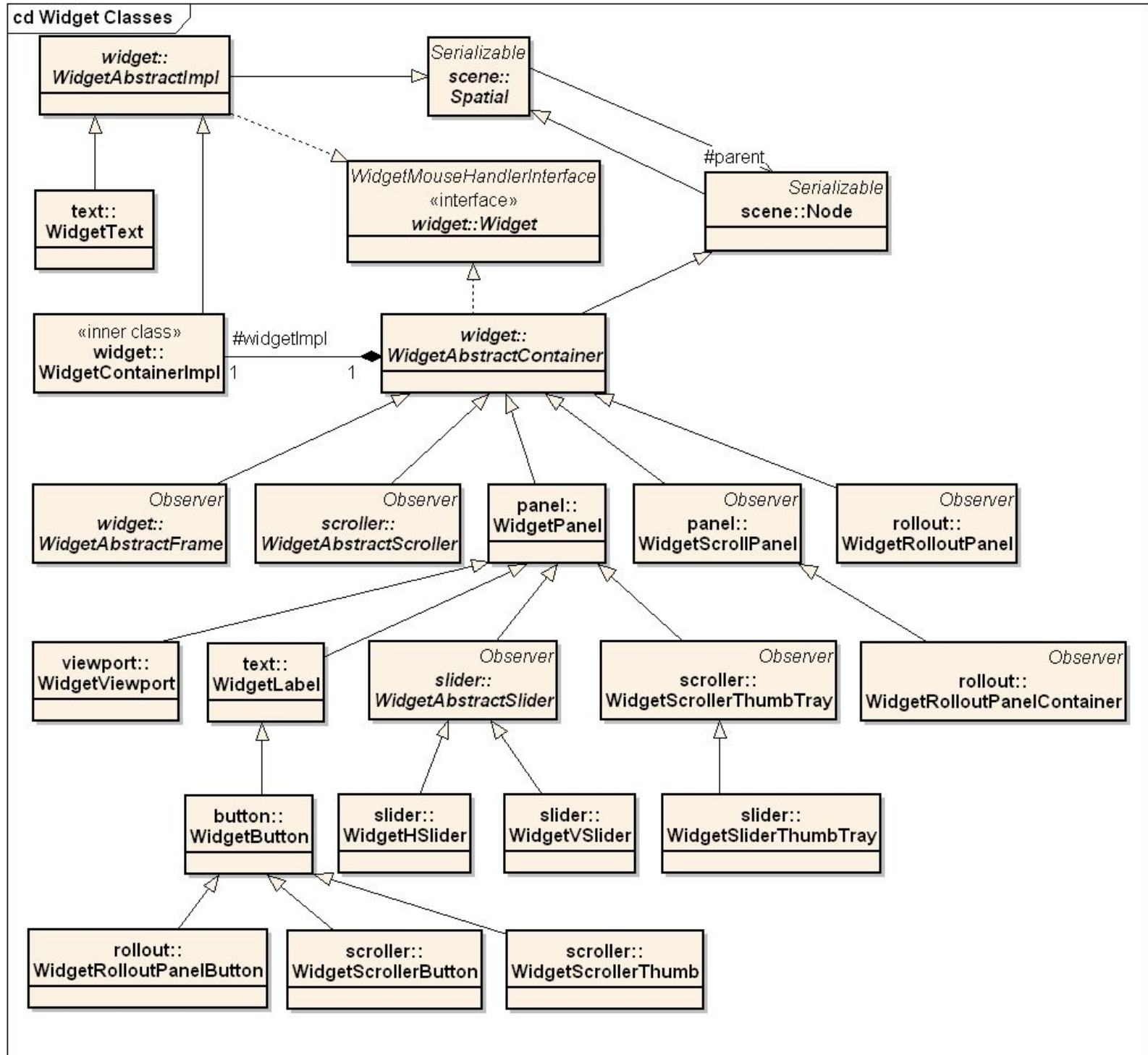
The jME GUI (cont.)

- Widgets (cont.)
 - WidgetAbstractContainer
 - Combines a Node with a Widget by extending Node and implementing the Widget interface.
 - Contains a concrete implementation of WidgetAbstractImpl to handle default Widget behavior.
 - Most Widgets extend WidgetAbstractContainer.

The jME GUI (cont.)

- Widgets (cont.)
 - WidgetAbstractFrame
 - Provides a top level framework for Widget management.
 - Extends WidgetAbstractContainer.
 - It is a Node in the Scene Graph.
 - Handles directing input events to the child Widgets in its sub-tree.
 - Maintains which Widget currently owns the input.

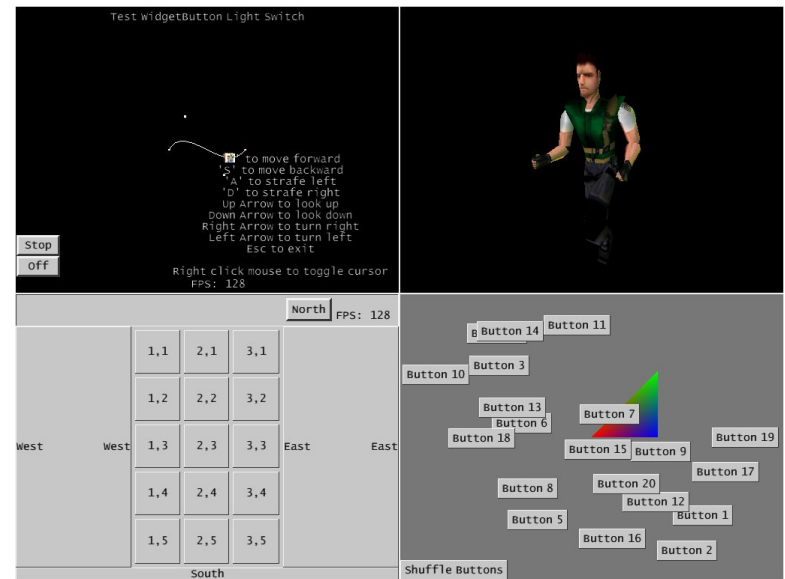
The jME GUI Implementation



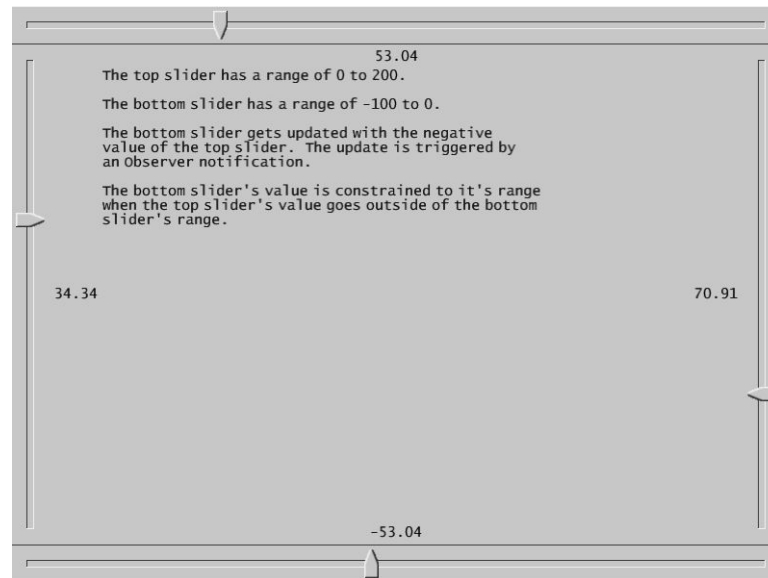
jME Demonstration



TestWidgetApp1



Test Multiple Viewports

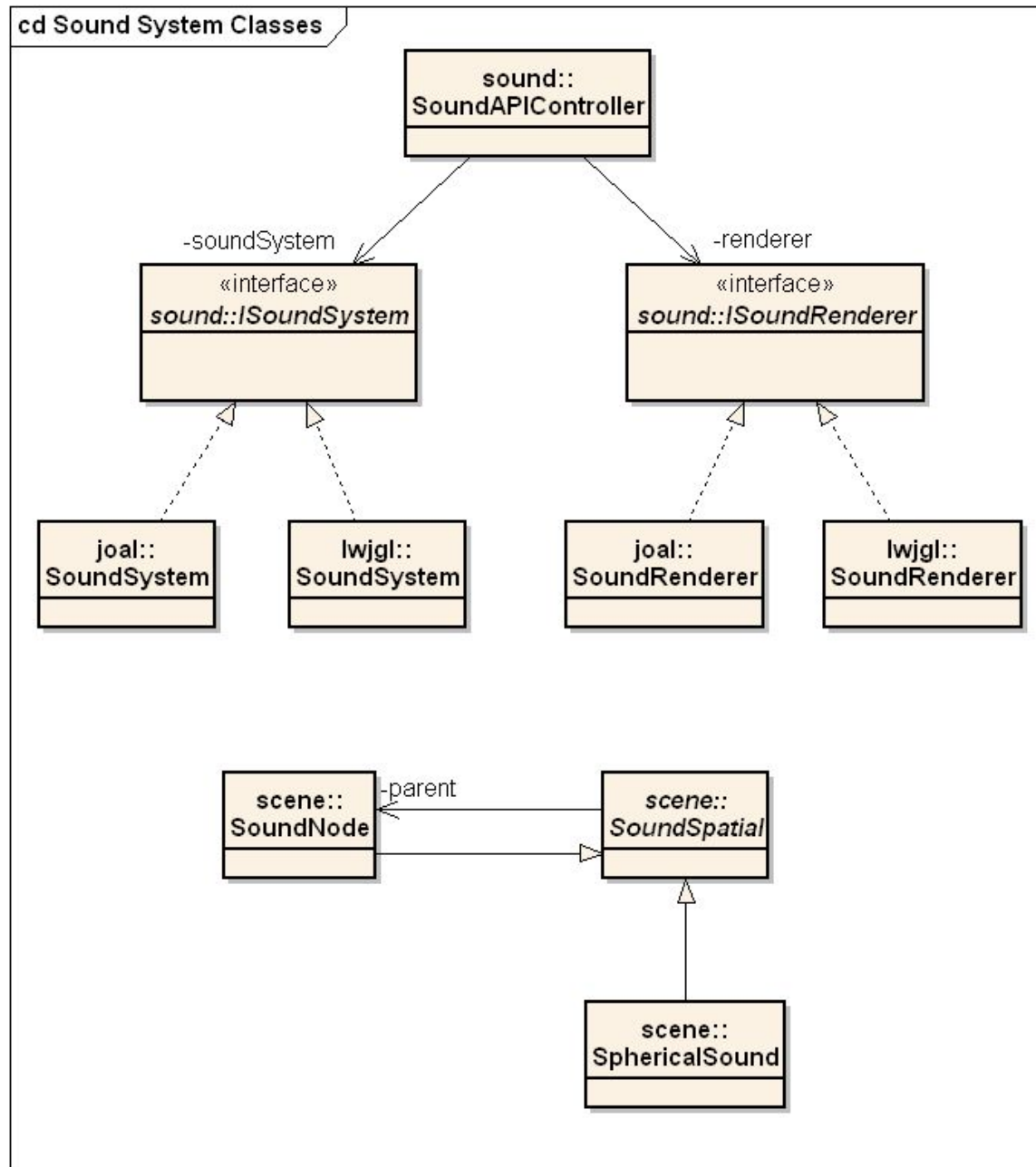


GUI Slider Interaction Test

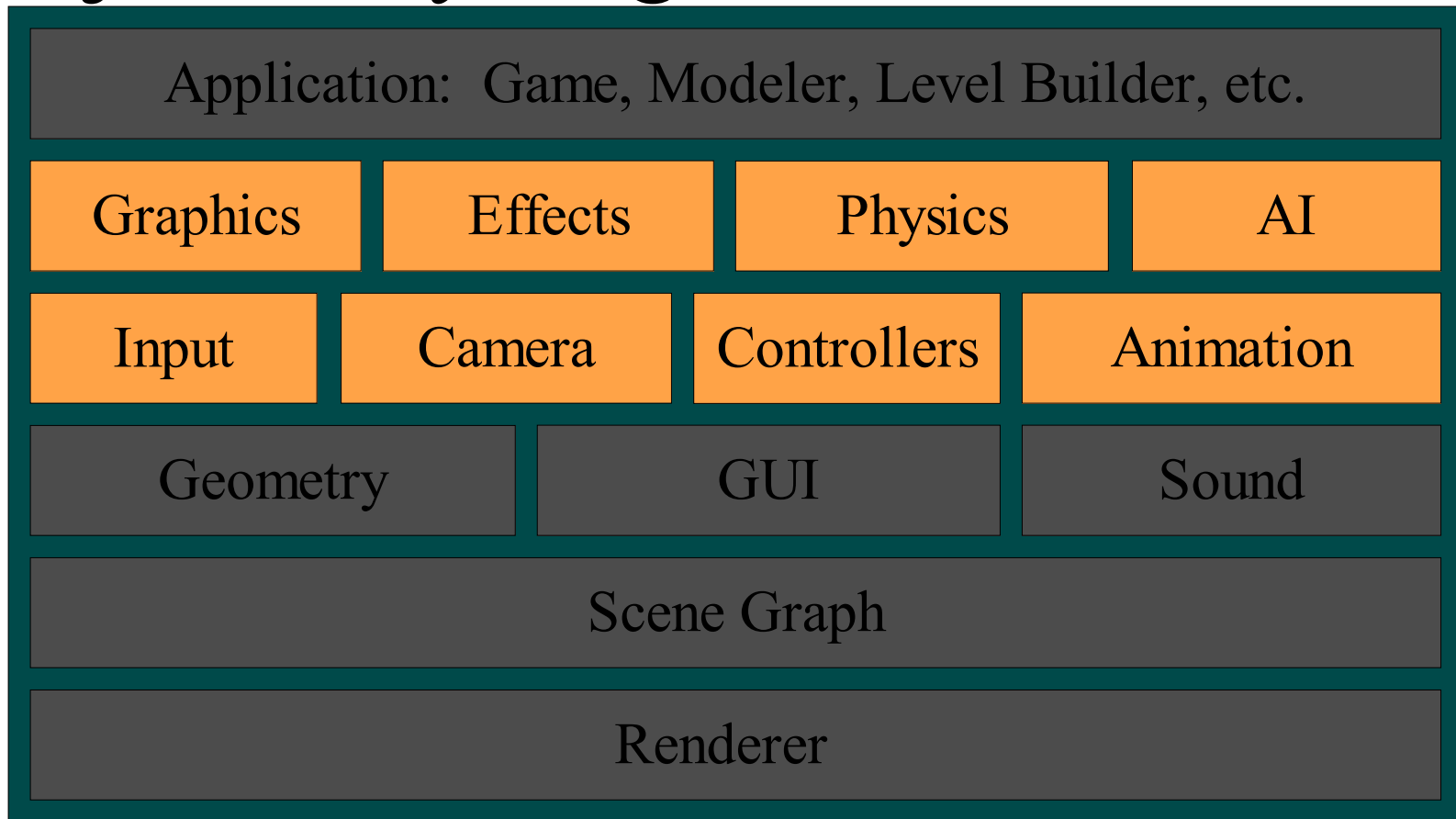
The jME Sound System

- Similar to the Renderer in architecture.
 - Platform specific sound systems are pluggable.
 - Sounds are rendered in 3D space.
- Specified by two interfaces:
 - ISoundSystem
 - ISoundSystem handles playing the sound.
 - ISoundRenderer
 - ISoundRenderer handles connecting the sound to the Scene Graph for handling 3D sound effects.
- See TestSoundGraph for an example.

The jME Sound System Implementation



jMonkey Engine Architecture



Java Native Interface (JNI):

LWJGL

JOGL

???

OpenGL / OpenGL ES

Operating System:

Windows

Linux

OSX?

jME Graphics

- Capabilities
 - Render to a texture.
 - Multi-texturing support.
 - Various model loaders.
 - MilkShape modeler ASCII format.
 - ASE - 3D Studio Max ASCII export format.
 - MD2 – Quake 2 model format.
 - Various built-in primitives.
 - Point, Line, Quad
 - Tri-mesh, Box, Pyramid
 - etc.

jME Graphics (cont.)

- Capabilities (cont.)
 - Various managed states.
 - Lights
 - Textures
 - Culling
 - Fog
 - Wireframe
 - ZBuffer
 - Materials
 - Alpha
 - Shading

jME Demonstration



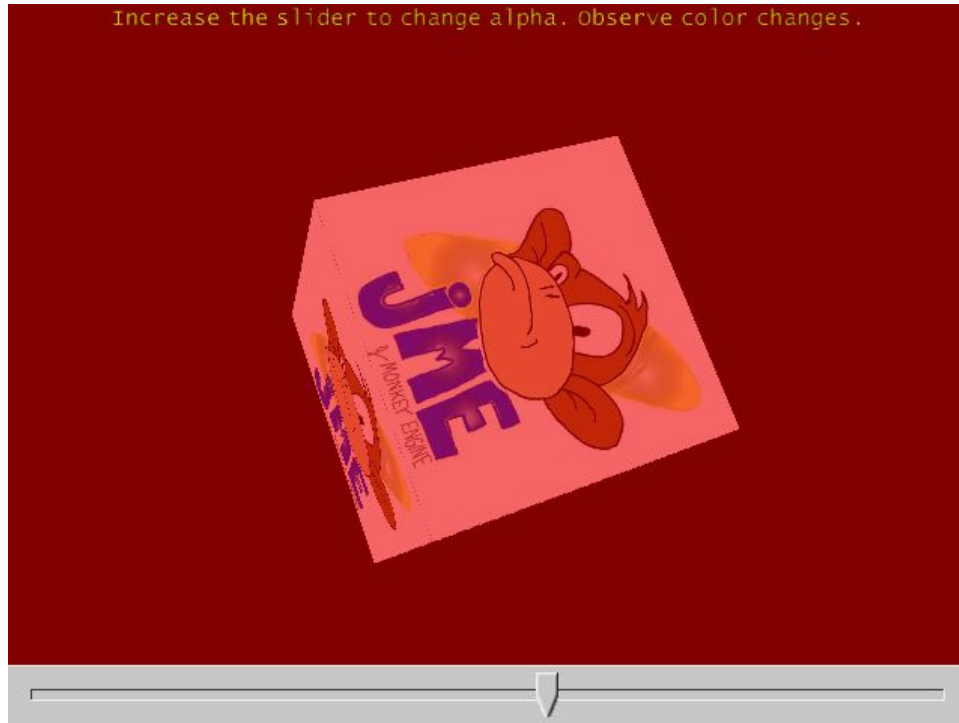
Bezier Mesh Test

jME Effects

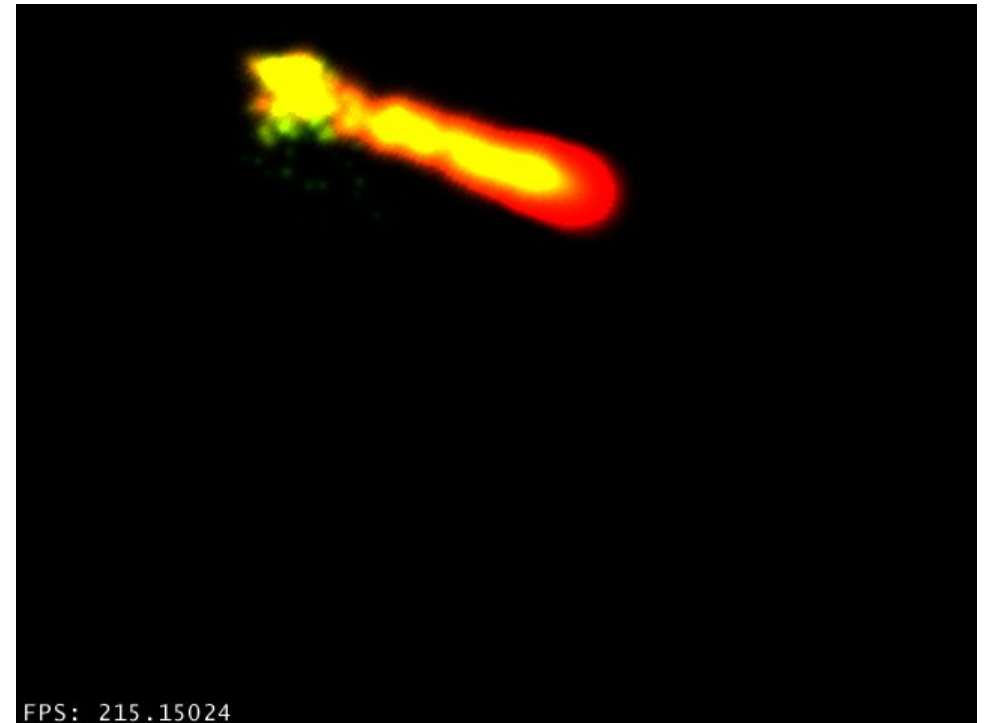


- Currently available:
 - Particle System
 - Tinting

jME Demonstration



Tinting Demo



Particle System Demo

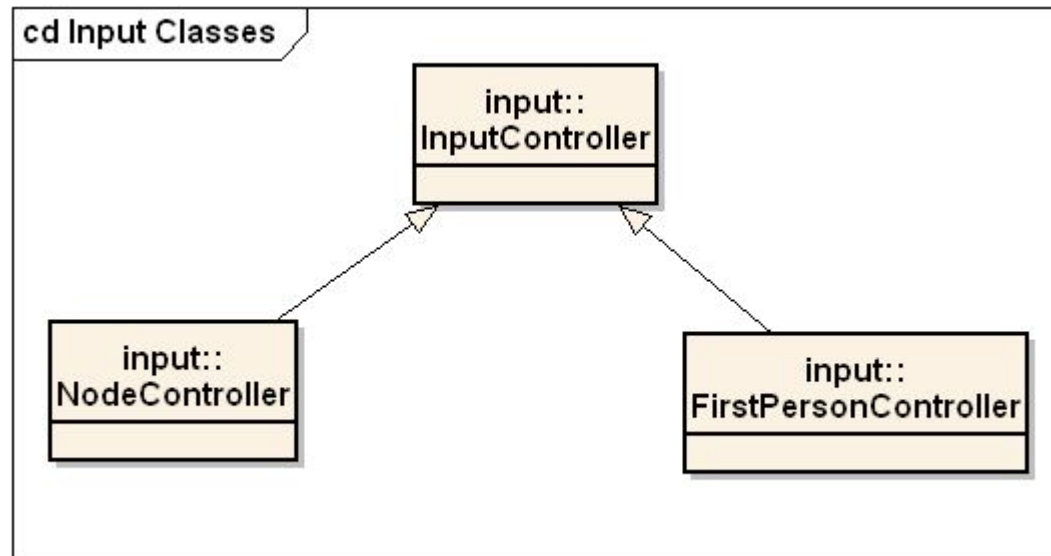


jME Input



- Currently available:
 - Mouse
 - Keyboard

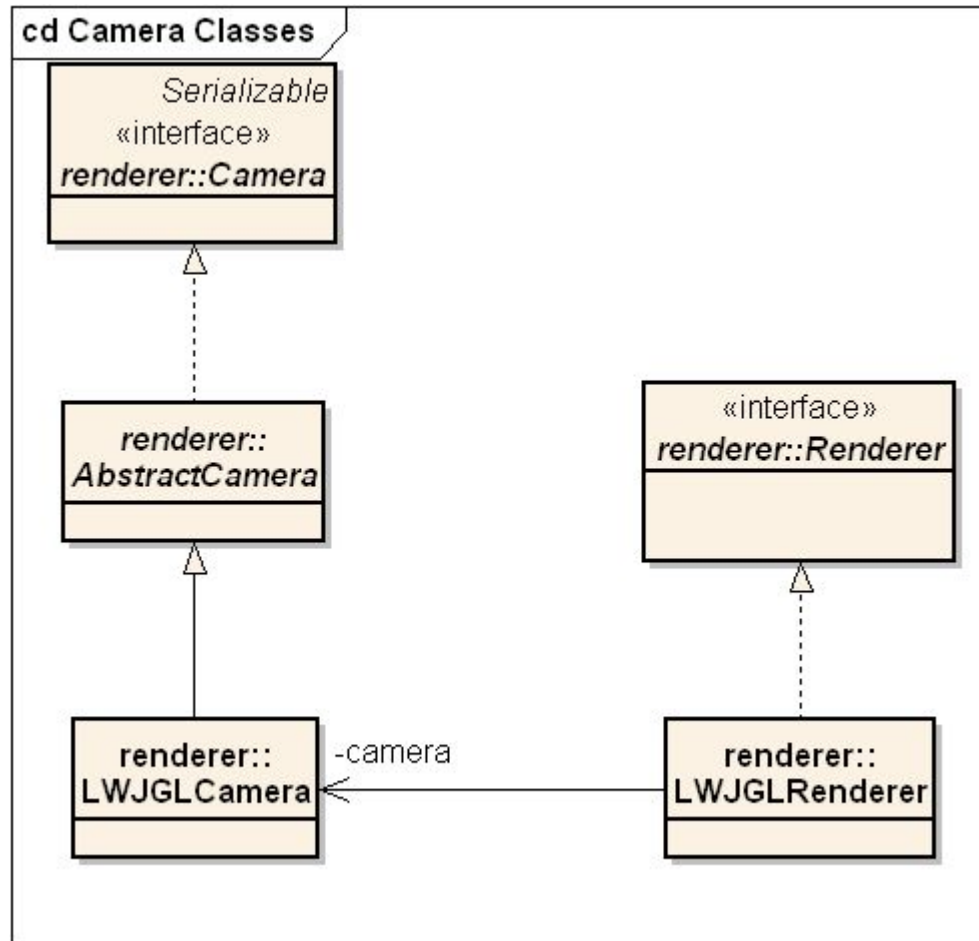
jME Input Implementation



jME Camera

- A camera model is used to project objects to the computer screen.
 - It is a combination of:
 - An eye point.
 - The location of the camera.
 - A view plane.
 - The 2D plane objects will be projected on.
 - A viewport.
 - A rectangular region of interest. A portion of the view plane.
 - A view frustum.
 - Aka viewing volume. Used for culling and clipping.

jME Camera Implementation





jME Demonstration



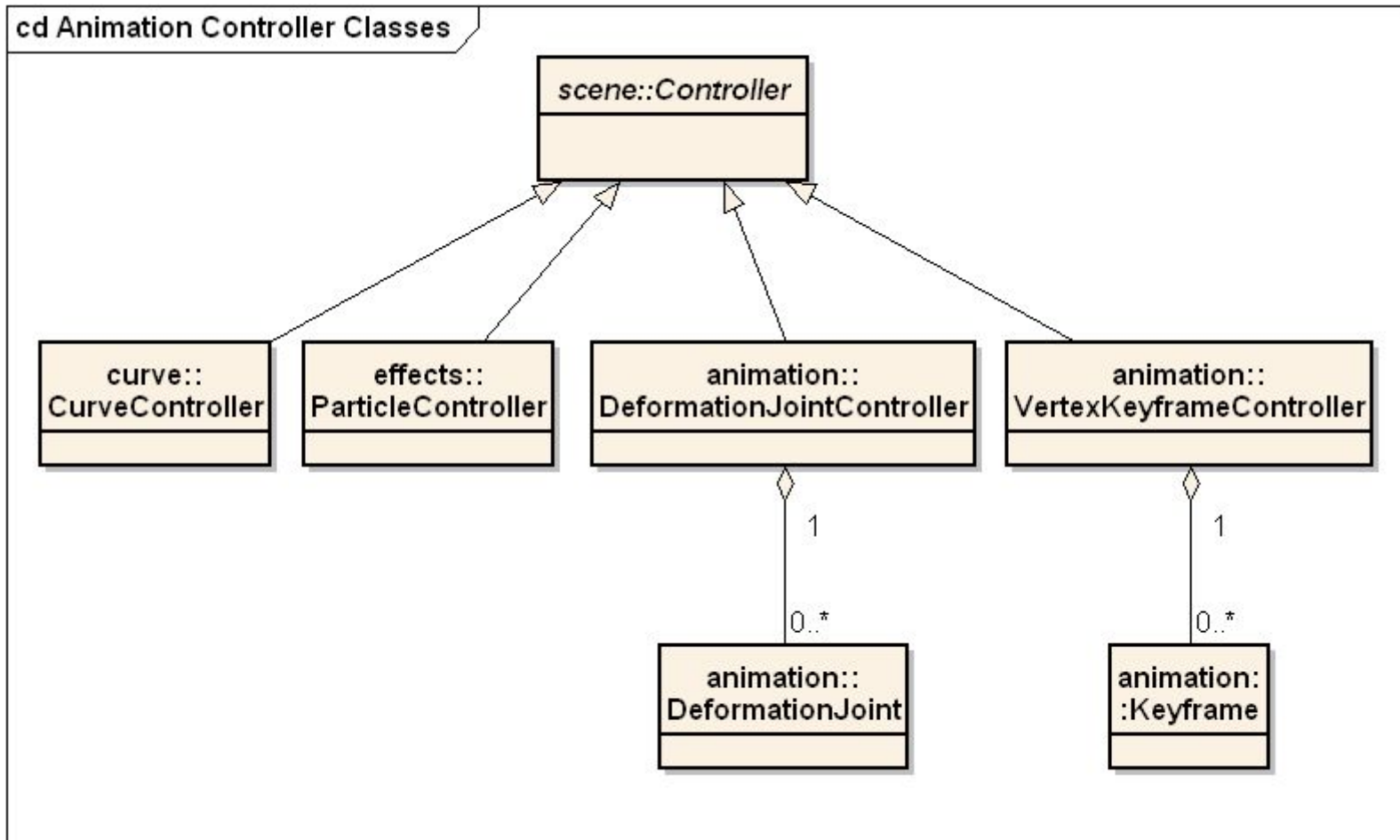
Camera / Render to Texture Demo

jME Controllers and Animation

- Controllers
 - Modify nodes and render states over time.
 - Used for animation of the scene graph.

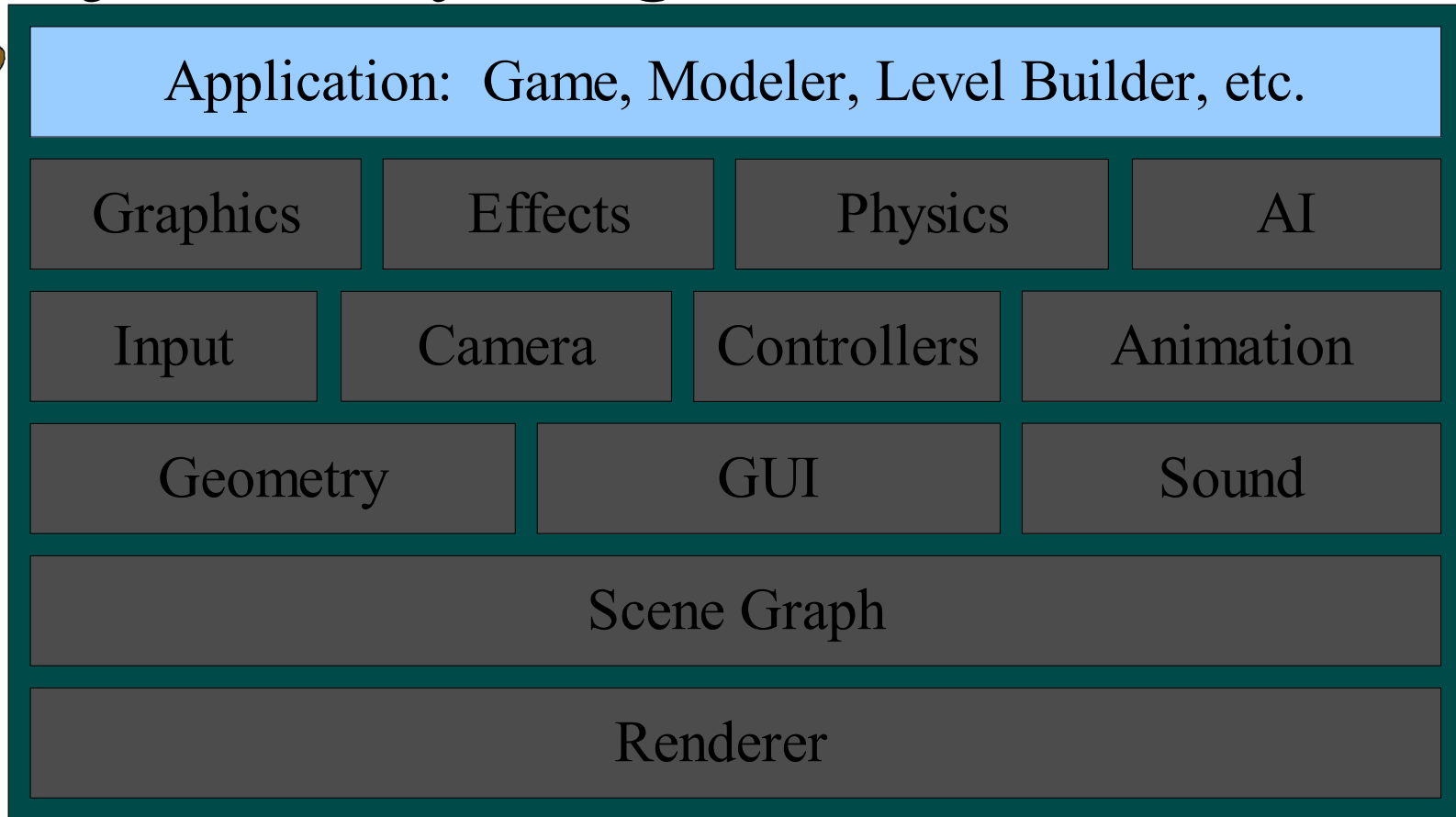


jME Controllers and Animation Implementation





jMonkey Engine Architecture



Java Native Interface (JNI):

LWJGL

JOGL

???

OpenGL / OpenGL ES

Operating System:

Windows

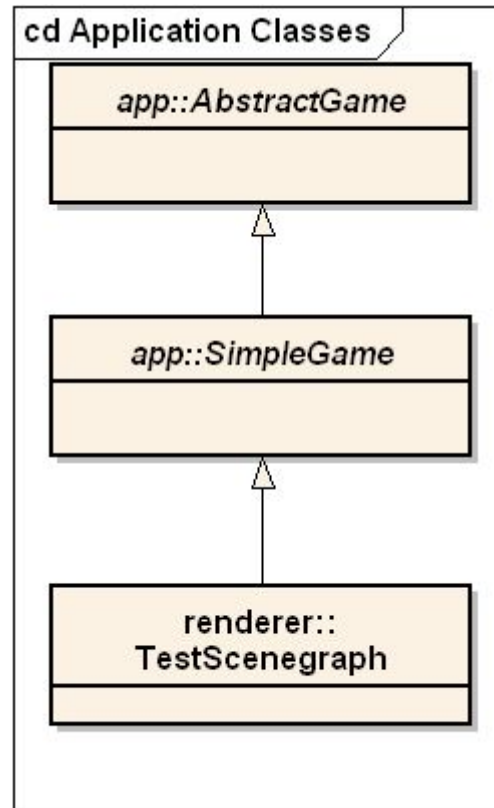
Linux

OSX

jME Applications

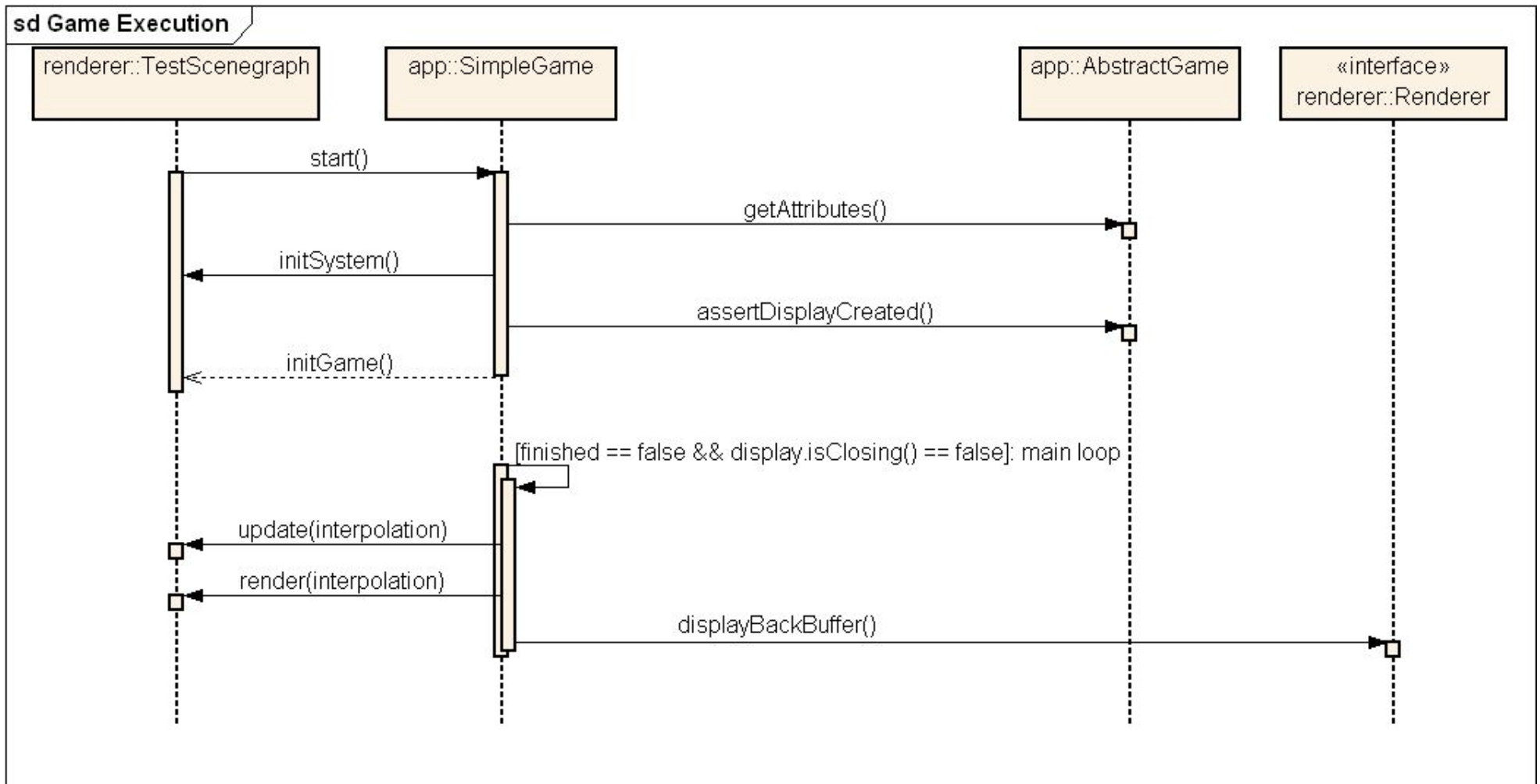
- Current
 - Test applications and Demos.
 - Available via WebStart on 'Tests and Demos' page.
 - <http://www.mojomonkeycoding.com/jmedemo.html>
 - The first jME game.
 - VolatileZapper
- Planned
 - Modeler / 3D Painter / Level builder
 - Aka, jME Studio.

jME Application Implementation

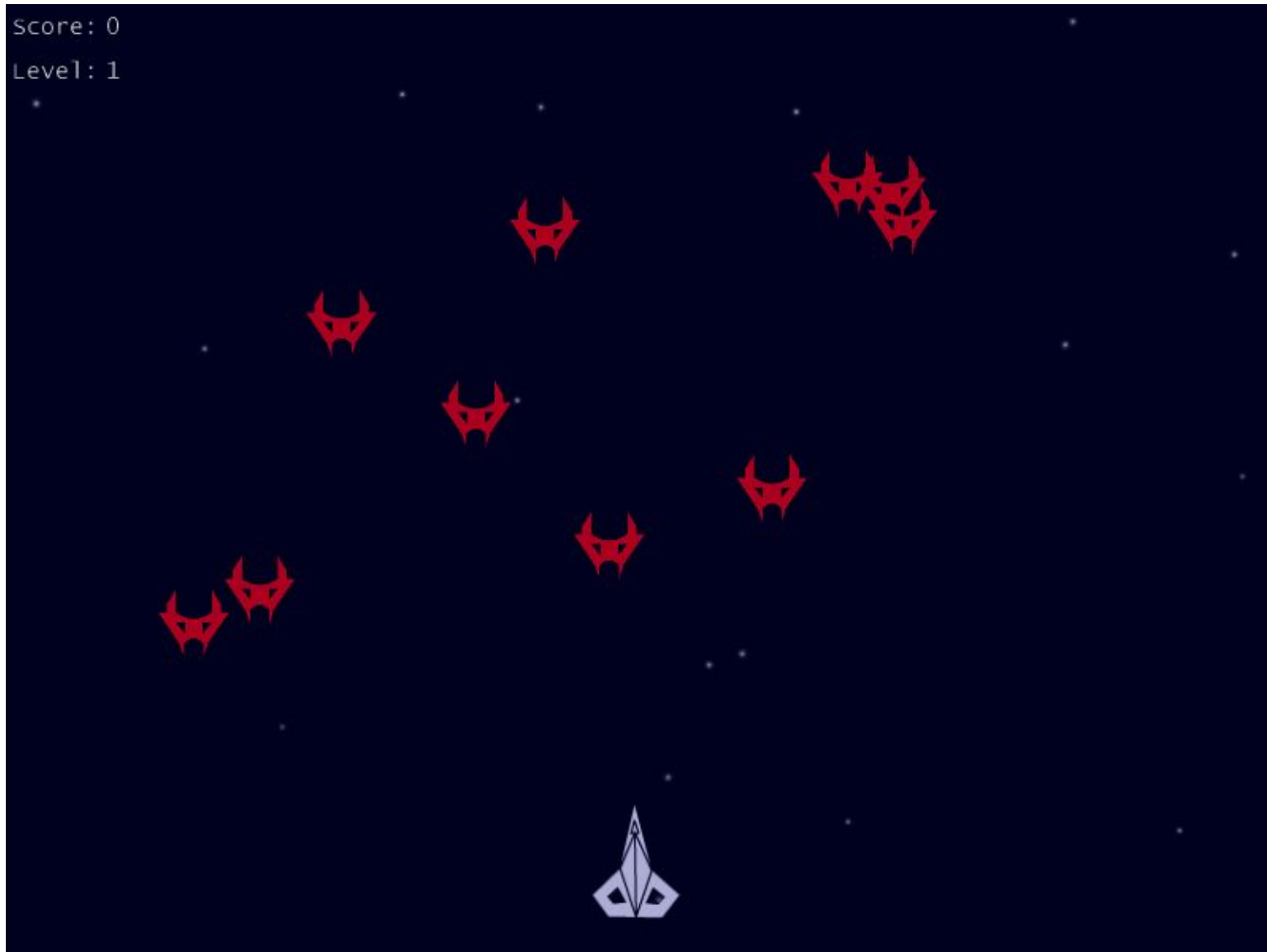




jME Application Implementation (cont.)



jME Demonstration



Volatile 7 – The First jME Game

Opportunities to Participate

- Physics
- Game AI
- Performance improvements (always)
- Testing
- Games, Games, and more Games
- Documentation
- JNI Renderer implementations

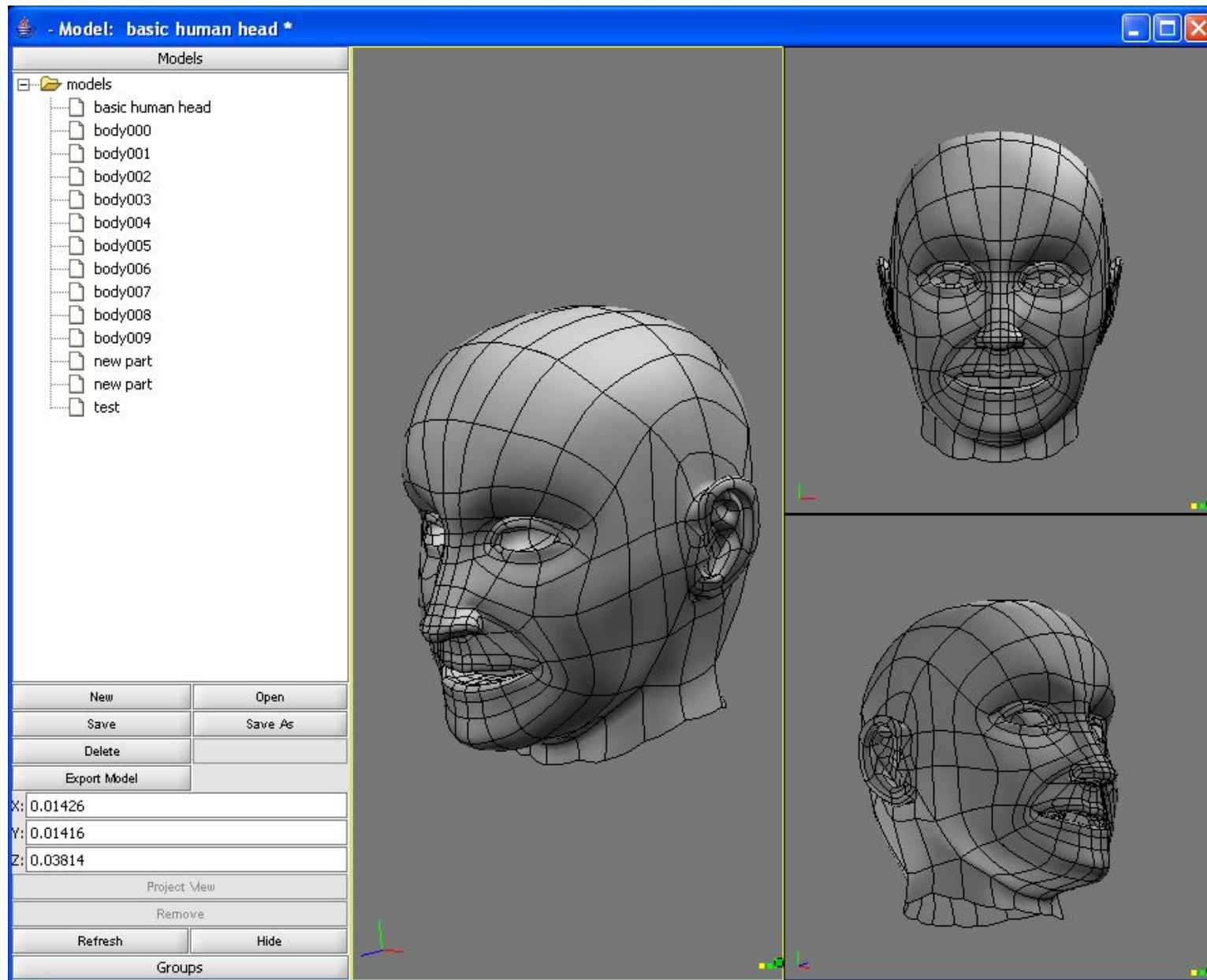
Gregg's Motivation

- I've been hooked on computer graphics ever since I typed in my first 'plot' and 'drawTo' commands on that Atari computer in the computer store way back in the fall of nineteen hundred and eight one.
- Java is cool!
- I really, really, did I say really, like computer graphics and Java.

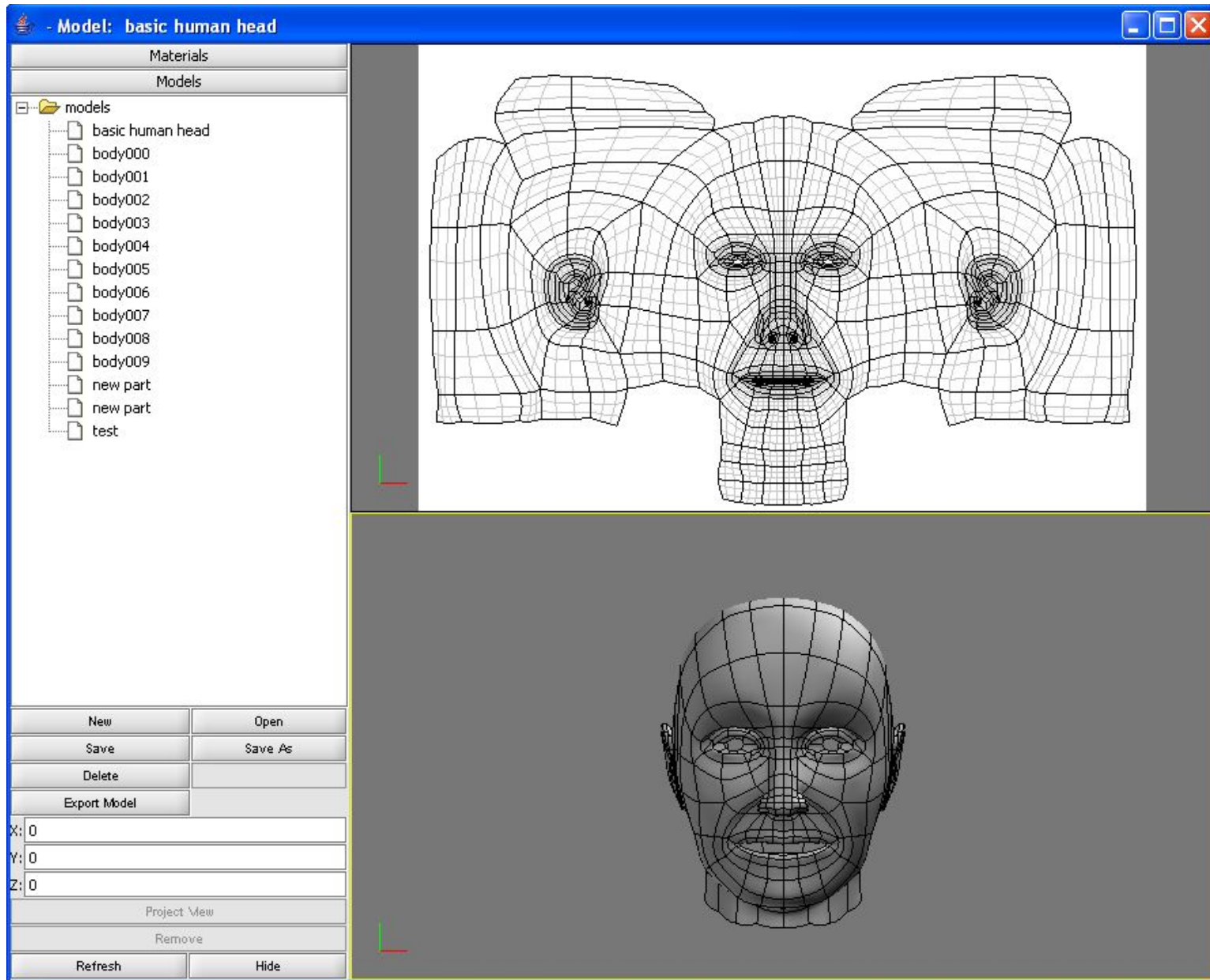
Gregg's Motivation (cont.)

- I want to write a Machinima system.
 - Machinima
 - Shooting film in virtual reality.
 - Creating films that use computer game technology.
 - More info: <http://www.machinima.com/>
 - jME provides the perfect platform for a Machinima system.
- The Modeler Prototype.
 - Last year (2003) I wrote a prototype modeler in Java.
 - It needed a better internal architecture. Enter jME.
- jME Studio
 - New modeler with 3D texture painting in 2004.
 - A full blown game creation / Machinima system in ????

Gregg's Prototype Modeler



Gregg's Prototype Modeler (cont.)



The Future of jME

- From Mark Powel:
 - Everyone is free to participate, join, contribute, etc. If you can code, and you want to help, I'm not going to turn you away.
 - Future plans are not set in stone.
 - Finish the core scene graph (terrain pages, spatial partitioning (BSP, portal, etc), some more basic effects).
 - This is fun for me, so I'm not going to treat it like a job. If there is something cool to add, I'll add it.

Acknowledgments

- Mark Powell
 - Thanks for writing TestScenegraph at a moments notice.
- All the jME developers.
 - Thanks for all those demos and applications and making them accessible with WebStart.
 - Thanks for the feedback.
- Doug “the inquisitor” Park
 - Thanks for taking time to review this presentation.

References

- 3D Game Engine Design
 - by David H. Eberly
 - <http://www.magic-software.com/Books.html>
 - ISBN: 1558605932
- The Elements of UML Style
 - Scott W. Ambler
 - ISBN: 0521525470

Links to Web Sites

- Main jME web site
 - <http://www.mojomonkeycoding.com/>
- jME discussion forums
 - <http://www.mojomonkeycoding.com/jmeforum/>
- jME test applications and demos.
 - <http://www.mojomonkeycoding.com/jmedemo.html>
- jME on java.net
 - <https://jme.dev.java.net/>

Links to Web Sites (cont.)

- OpenGL
 - <http://www.opengl.org/>
- OpenGL ES
 - <http://www.khronos.org/index.html>
- LWJGL project
 - <http://www.lwjgl.org/>
- JOGL project
 - <https://jogl.dev.java.net/>
- Machinima
 - <http://www.machinima.com/>

Software

- Software used to make this presentation.
 - OpenOffice.org 1.1.0
 - <http://www.openoffice.org/>
 - Enterprise Architect
 - <http://www.sparxsystems.com.au/>

JME
MONKEY ENGINE