Creating Games on the Java™ Platform with the jMonkeyEngine

Joshua Slack, Rikard Herlitz
jMonkeyEngine
jMonkeyEngine.com

TS-5711
Our Goal:

To get you started on the path to creating professional quality 3D games and applications in Java™ technology today!
Our other jobs…

NCsoft Corp

- Makers of popular online games such as Guild Wars, Lineage, City of Heroes, and Tabula Rasa
- Started hiring jMonkeyEngine developers in 2006
- Demonstrated a strong commitment to the Java gaming community by actively contributing back to the jMonkeyEngine
Agenda:

- Myths and Realities
- Getting Your Feet Wet
- Taking it to the Next Level
- Trail Blazers
- Q&A
Myths and Realities

> #1 - Speed
  • Myth: Java technology is too slow for games
  • Reality: Since 1.4.2, Java technology has closed the speed gap. Besides, much of the heavy lifting in games can be left to the hardware.

> #2 - Visual Quality
  • Myth: Java technology-based games are ugly. Just look at [game X]
  • Reality: With jMonkeyEngine, quality is limited by the art assets you have available and your skill as a graphics programmer—not the language.
Myths and Realities

Is this what Java based games have to look like?
Myths and Realities

Here’s an example of what can be done!
Let’s get our feet wet!

What is the jMonkeyEngine?

- jMonkeyEngine is a 3D scene graph that empowers you to create high quality games and applications with engaging graphics and sound.

- The engine is written 100% in Java programming language and uses a thin JNI layer to communicate directly with your audio, video and input device hardware.
# The 10,000 Foot View

## Your Code

<table>
<thead>
<tr>
<th>jMonkeyEngine Core</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scene Management</td>
</tr>
<tr>
<td>Input Management</td>
</tr>
<tr>
<td>Resource Management</td>
</tr>
<tr>
<td>Graphics Utils</td>
</tr>
<tr>
<td>Model Importers</td>
</tr>
<tr>
<td>Collision / Boundings</td>
</tr>
<tr>
<td>Audio Management</td>
</tr>
<tr>
<td>Community Extensions</td>
</tr>
</tbody>
</table>

## LWJGL

- OpenGL
- OpenAL
- jInput

## jME-physics

- ODE, PhysX, …
Making a Simple Game

“MonkeyPong”

Why?
• We aren’t artists
• Everyone knows the mechanics of the game
• Everything we need is right there in the engine API
First Step – the framework

We can get up and running very quickly by using one of JME’s application classes:

- AbstractGame, SimpleGame, SimplePassGame, StandardGame

We’ll use SimpleGame for this game.

```java
public class MonkeyPong extends SimpleGame {
    protected void simpleInitGame() {
    }
}
```
Next – the game elements

We use jME’s primitives for our ball, walls and paddles

```java
ball = new Sphere("Ball", 8, 8, 2);
bball.setModelBound(new BoundingSphere());
bball.updateModelBound();

player1 = new Box("Player1", new Vector3f(), 2, 5, 10);
player1.setModelBound(new BoundingBox());
player1.updateModelBound();
player1.getLocalTranslation().set(-100, 0, 0);
```
Now – input control

The simplest way of getting keyboard input is through the KeyBindingManager

```java
simpleInitGame() {
    KeyBindingManager.getKeyBindingManager().set("MOVE_UP", KeyInput.KEY_W);
}

simpleUpdate() {
    if (KeyBindingManager.getKeyBindingManager()
        .isValidCommand("MOVE_UP", true)) {
        player1.getLocalTranslation().z -=
            player1Speed * timer.getTimePerFrame();
    }
}
```
Mix in some collision…

> Bounding box collision is more than enough for us

```java
public void simpleUpdate() {
    if (player1.hasCollision(ball, false)) {
        ballVelocity.x *= -1f;
    }

    if (sideWalls.hasCollision(ball, false)) {
        ballVelocity.z *= -1f;
    }

    if (player1GoalWall.hasCollision(ball, false)) {
        player2Score++;
    }
}
```
Monkey Pong Live Demo #1
That was too easy, let’s add sound!

First we setup a track in our init section:

```java
AudioTrack collideSound = 
    audio.createAudioTrack("/jmetest/data/sound/laser.ogg", false);
collideSound.setRelative(true);
```

Then we’ll simply play the track when we detect a collision:

```java
collideSound.play();
```

Finally, make sure we update the AudioSystem in our game loop:

```java
AudioSystem.getSystem().update();
```
More spice…

> Creating a particle system is easy through the factory

```java
ParticleMesh particles = ParticleFactory.buildParticles("particles", 30);
```

> Setup particle system lifetime, sizes, colors, etc.

```java
particles.setInitialVelocity(.05f);
nparticles.setStartSize(3f);
...
```

> Add an optional influence like gravity, wind or swarming

```java
SwarmInfluence swarm = new SwarmInfluence(new Vector3f(particles.getWorldTranslation()), .001f);
nparticles.addInfluence(swarm);
```
Adding water…

> Realistic water with reflections and refraction is just a few lines of code

```java
waterEffectRenderPass = new WaterRenderPass(cam, 4, false, true);
waterQuad = new Quad("waterQuad", 1, 1);
waterEffectRenderPass.setWaterEffectOnSpatial(waterQuad);
```
Terrain...

Generate a terrain from image data or through our heightmap generators

```java
RawHeightMap heightMap = new RawHeightMap(MonkeyPong.class.getClassLoader().getResource("jmetest/data/texture/terrain/heights.raw").getFile(), 129, RawHeightMap.FORMAT_16BITLE, false)
TerrainPage page = new TerrainPage("Terrain", 33, heightMap.getSize(), terrainScale, heightMap.getHeightMap(), false);
```
Monkey Pong Live Demo #2
Let’s recap…

- jMonkeyEngine provides a lot of foundational classes and examples to get you started
- You can use jME’s supplied special effects to add extra punch to your game
- Even with a programmer’s eye for art, you can build a fun game

- Get a closer look at the source for this example from the jME project svn
Taking it to the Next Level

➢ Production quality games require a whole new level of effort
➢ To make such a game we need to work together with other creative types:
  • Artists
  • Level builders
  • Game designers
➢ Collaboration is achieved through good pipeline and tools
➢ Tool installation and start-up needs to be fast and hassle-free
Pipeline

- Your game’s pipeline is the path that artist generated content takes to get from their mind into the game

- jME has support for most popular image formats and some standard audio formats:
  - tga, png, jpg, gif, bmp, dds.
  - wav, ogg

- We also have support for several standard model formats:
  - Ase, Obj, 3ds, Md2-Md5, X3d, Milkshape and Collada
But jME needs to improve in this area:

- Improved Collada support
- Community is working on better md5 support.
- Create an XML equivalent to our binary import/export process and let the community create their own exporters (or tools.)
Tools turn your pipeline assets into a game environment

Options Include:

• MonkeyWorld 3D – Built using SWT and Eclipse RCP
• Various small utilities in jME – Particle Editor, Control Editor, etc.
• Rolling your own tool
Roll Your Own - An Example: NCsoft’s World Builder

- Swing + jME Canvas
- Created by a small team in short time
- Some features include:
  - Asset integration with Perforce, terrain generation/painting, lighting, lightmap generation, LOD setup, etc.
NCsoft’s Java technology-based World Builder
Tools continued…

► Not many cons
  • Direct memory handling
  • Native buffer performance

► But lots of pros
  • Your tool runs anywhere (many artists prefer Macs)
  • Swing GUI development
  • Exception handling (not many hard crashes)
  • Logging (console, file, mail)
  • Scripting (Lua or JavaScript™ programming language, etc.)
Client

Things to consider on your game client:

- Aim for min spec, next-gen, or use fallbacks to handle both?
- Give users controls – give as many configuration options as you can to allow the user to tweak things for their platform. (But use smart default settings.)
- User Interface – several options: BUI, FengGUI, jMEDesktop or your own
- Deploying your game:
  - Format: Applet or application
  - Java technology installation and min version
  - Delivery: Webstart, GetDown, etc.
  - Crash reporting and bootstrapping
- Future options: Java Consumer JRE
To recap...

- You can make use of existing model and asset formats
- To make a professional game, you need artists and you need to provide them with tools
- There are some existing tools
- It’s easy to make your own tools with jME embedded
Evaluating Java as a Game Platform: Selling Points

> Versatile deployment options
  • Applet or application; fullscreen or windowed

> Error handling is more elegant
  • Easier than in traditional C/C++ frameworks

> Cross Platform:
  • OpenGL + Java platform means never having to say you’re sorry

> The Power of Java technology:
  • Easy to use, familiar, powerful
  • Lots of open source code out there to make use of
  • Easy integration into web-services, etc.
Evaluating Java as a Game Platform: Issues

Major Problem Area – Infrastructure
- Lack of source materials (books, articles, code samples)
- Lack of existing games
- Lack of developer support (disbelief, inexperience)
- Lack of middleware support

ALL of these points can be turned around rather quickly
- This is still a fairly new area for Java technology
- Releasing one or two high quality games would change attitudes and give inspiration (and create experienced developers)
- Use by companies or universities with money to spend will encourage existing middleware to add Java technology support
- Will this happen?
It's Already Happening – Commercial Games

**Bang! Howdy**
Three Rings
Fast-paced wild west tactical strategy

**Hockey Heroes**
Jadestone
Ice hockey with an attitude.
It's Already Happening - Commercial Games

Call of the Kings - Gamalocus
Online fantasy strategy-roleplaying game.

Nord - SLX Games
Your personal online social experience.
It's Already Happening - Commercial Games

**JCRPG – Classic RPG**

**Project X**

**NCsoft Corporation**

Unannounced game under development…
It's Already Happening – Casual Gaming

BigFun
Motorcycle Trials
OurAwesomeGames

Mad Skills Motorcross
Turborilla
Race against the neural network trained riders to prove you’re the best.
It's Already Happening – Event Based Entertainment

Polyball 2007

Sail the high seas and do ship combat in front of a ball room of nine thousand guests.
It's Already Happening – Student Projects

**Matics**  
Georgia Tech  
Puzzle based platformer with real physics.

**Lord of the Fjord**  
Georgia Tech  
Viking boat bongo battle!
It's Already Happening – Research Applications

Wubble World
USC
Interactive playground for AI research

Multitouch Environment
Durham University
Research into interactive learning interfaces
It's Already Happening – Commercial

Project Wonderland

JPericia
Team Cadanus
Scene visualizer for crime scene investigation in Brazil.
It's Already Happening - Science

**Intelligent Robotics Group - NASA Ames**
jME 2.0 – The future

▶ Planned features include:
  • Easy/safe threading
  • Separate game and render loop
  • Visibility/space partitioning handling in core
  • More Enumerations
  • Latest in OpenGL features
  • Refactoring / documentation
  • Pipeline Improvements
  • Community code process

▶ The jME 2.0 Architecture group
jMonkeyEngine in action!
For More Information

➢ Check us out on line:
   • Home: www.jMonkeyEngine.com
   • Wiki: www.jMonkeyEngine.com/wiki

➢ Talk to the community!
   • Forums: www.jMonkeyEngine.com/jmeforum

➢ Check out the “MonkeyPong” source:
   • jME’s SVN repository:
     • http://code.google.com/p/jmonkeyengine/source/checkout
THANK YOU

Rikard Herlitz, Joshua Slack
jMonkeyEngine
jMonkeyEngine.com

TS-5711