M3G – Java Mobile 3D

Tomi Aarnio
Nokia Research Center
Agenda

- What is M3G
- What’s new in 2.0
M3G – Mobile 3D Graphics API for Java

- Enables real-time 3D on mass-market phones
  - Came out in 2004, now almost universally adopted
  - Installed base somewhere between 500M-1B

- Retained mode API
  - OpenGL ES features wrapped into Java objects
  - Animation and scene graph layered on top
Mobile 3D Graphics APIs

Native C/C++ Applications

Java applications

M3G

OpenGL ES

CPU / GPU
Mobile Java

Pros
+ More widely available than any other platform
+ The only platform on many/most phones
+ Easy to write code that works

Cons
– Different devices have different APIs (and bugs)
– Latest hardware features not always available
– Performance not as good as in C/C++
M3G Design Principles

#1 Minimize the amount of Java code

#2 Do not require graphics hardware

#3 Enable easy content creation
Programming Model

- M3G is not an “extensible” scene graph
  - No interfaces, events, or render callbacks
  - No threads; all methods are synchronous

- Scene update is decoupled from rendering
  - `render` ➔ Draw the scene, no side-effects
  - `animate` ➔ Update the scene to the given time
  - `align` ➔ Re-orient target cameras, billboards
Main classes

- **Graphics3D**: 3D graphics context. Performs all rendering.
- **Loader**: Loads individual objects and entire scene graphs.
- **Mesh**: Encapsulates triangles, vertices and appearance.
- **World**: Scene graph root node.
Example scene graph

World

SkinnedMesh

Group

Camera

Group

Light

Group

Mesh

Group

Mesh

Group

Group

Group
Components can be shared

- World
  - Mesh
  - Camera
  - SkinnedMesh
    - Appearance
      - Texture2D
  - Mesh
    - Appearance
Mesh

- One VertexBuffer, containing VertexArrays
- 1..N submeshes (IndexBuffer + Appearance)
Simple animation player

```java
world = (World) Loader.load("/scene.m3g")[0];

void paint(Graphics g) {
    world.animate(currentTime);
    graphics3d.bindTarget(g);
    graphics3d.render(world);
    graphics3d.releaseTarget();
}
```
犬友 (Dear Dog) Demo
Creating art assets

- DCC Tool
  - Exporter

- Intermediate Format (M3X, COLLADA)

- Optimize, Compress, Preview

- Delivery Formats (m3g, jpg, png)

- Runtime Scene Graph

- M3G Loader
Mascot Capsule M3G Exporter
Mascot Capsule M3G Viewer
Wizzer Works M3G Viewer
Selected open source projects

- [www.wizzerworks.com](http://www.wizzerworks.com)
  - M3G Toolkit & Viewer for manipulating .m3g files
- [m3x.dev.java.net](http://m3x.dev.java.net)
  - XML encoding of the .m3g file format + tools
- [www.microemu.org](http://www.microemu.org)
  - Java ME stack implemented on Java SE / Android
- [lwuit.dev.java.net](http://lwuit.dev.java.net)
  - Lightweight UI Toolkit, uses M3G for transition effects
Start developing!

• Choose IDE
  • [link](www.eclipse.org)
  • [link](www.netbeans.org)

• Choose SDK
  • [link](forum.nokia.com/java)
  • [link](developer.sonyericsson.com/java)
  • [link](mpowerplayer.com/sdk)

• Choose Exporter
  • [link](www.m3gexport.com) – Maya
  • [link](www.mascotcapsule.com/M3G) – Max, Maya, Lightwave, XSI
  • [link](www.nelson-games.de/bl2m3g) – Blender (open source)
Example Games
Playman Beach Volley – RealNetworks

- 2D backdrop
- 3D background
- 2D spectators
- 3D field
- 2D players
- 2D overlays

~7 layers of 2D and 3D!
Playman Winter Games – RealNetworks

2D

Perspective and depth

3D

Side view only
Playman World Soccer – RealNetworks

- 2D/3D hybrid
- Cartoon-like 2D figures in a 3D scene
- 2D particle effects etc.
Tower Bloxx – Digital Chocolate

- Puzzle/arcade mixture
- 3D with 2D overlays and backgrounds
Mini Golf Castles – Digital Chocolate

- 3D with 2D background and overlays
- Skinned characters
Rollercoaster Rush – Digital Chocolate

- 2D backgrounds
- 3D main scene
- 2D overlays
M3G 2.0
M3G 2.0

- Supercedes M3G 1.1
  - Adds programmable shaders in the high end
  - Improved features & perf also in the low end
  - Fully backwards compatible

- Work in progress
  - Get the Proposed Final Draft at [www.jcp.org](http://www.jcp.org) → JSR 297
  - Developer feedback can still make a difference!
Design Goals

Target all devices
1. Programmable HW
2. No graphics HW
3. Fixed-function HW

Enable reuse of
1. Assets & tools (.m3g)
2. Source code (.java)
3. Binary code (.class)
M3G 2.0 is a superset of 1.1

M3G 1.1
Feature set

OpenGL ES 1.0 + scene graph
M3G 2.0 is a superset of 1.1

M3G 2.0 Core

M3G 1.1

Feature set

OpenGL ES 1.1 + scene graph
M3G 2.0 is a superset of 1.1

- M3G 2.0 Core
- M3G 1.1 Feature set
- M3G 2.0 Advanced

OpenGL ES 2.0 + OpenGL ES 1.1 + scene graph
Why Not Shaders Only?

- Shader Hardware
- Fixed Function Hardware
- No Graphics Hardware

Device sales in 2011?
New Core features due to popular demand

• Optimized mesh deformation & animation
  • Morphing and skinning on the same mesh
  • Morph targets applied on a subset of the base mesh
  • Multichannel keyframe sequences
  • Animation event tracks

• Scene graph
  • Bounding volume hierarchies (boxes and spheres)
  • Neatly encapsulated multipass render-to-texture effects
  • Transparent objects can be sorted back-to-front
  • Lots of convenience methods
New Core features due to popular demand

• Improved texturing
  • Compressed textures, JPEG
  • Non-power-of-two sizes
  • Video textures
  • Bump mapping

• New primitive types
  • Point sprites, lines
  • Float/half vertices
Level of Detail (LOD)

- A Group node can select one of its children
  - Based on their size in screen pixels
  - Similar to mipmap level selection

- Formally
  1. Compute \( s = \text{pixels per model-space unit} \)
  2. Select the node whose ideal scale \( s_i \) satisfies

\[
\max \{ s_i \mid s_i \leq s \}
\]
Collision Detection

- Each Node can have a collision volume
  - k-DOP = Discrete Oriented Polytope
  - AABB with corners & edges chopped off

- `world.collide(...)` to find all collisions
**Simple vertex shader**

```cpp
#pragma M3Gpositionattrib(myVertex)
#pragma M3Gvertexstage(clipspace)
void main() {
    m3g_ffunction();
    gl_Position = myVertex;
}
```

Declare attribute semantics via `#pragmas`

Built-in function for morphing, skinning, model-view-projection

Result passed to the fragment shader.
Summary
Summary

- M3G enables real-time 3D on mass-market phones
  - Easy to use, high performance scene graph API
  - Installed base somewhere between 500M-1B
  - Grab the tools and start developing!

- M3G 2.0 is under development
  - Adds programmable shaders in the high end
  - Improved features & perf also in the low end
  - Fully backwards compatible
M3G – Java Mobile 3D

Tomi Aarnio
Nokia Research Center