



**SIGGRAPH2005**



SIGGRAPH2005

# Developing Mobile 3D Applications with OpenGL ES and M3G

Kari Pulli	Nokia Research Center & MIT CSAIL
Jani Vaarala	Nokia
Ville Miettinen	Hybrid Graphics
Tomi Aarnio	Nokia Research Center
Mark Callow	HI Corporation

# Today's program

---



- Start at 1:45
- Intro  
10 min, Kari Pulli
- OpenGL ES overview  
25 min, Kari Pulli
- Using OpenGL ES  
40 min, Jani Vaarala
- OpenGL ES performance  
30 min, Ville Miettinen
- Break 3:30 – 3:45
- M3G Intro  
5 min, Kari Pulli
- M3G API overview  
50 min, Tomi Aarnio
- Using M3G  
45 min, Mark Callow
- Closing & Q&A  
5 min, Kari Pulli
- End at 5:30

## Challenges for mobile gfx



- Small displays
  - getting much better
- Computation
  - speed
  - power / batteries
  - thermal barrier
- Memory



Fairly recently mobile phones used to be extremely resource limited, especially when it comes to 3D graphics. But Moore's law is a wonderful thing.

The displays used to be only 1-bit black-and-white displays, that update slowly, with resolutions like 48 x 84 pixels. However, the display technology has developed by leaps and bounds, first driven by the digital cameras, now by mobile phones. Only 12-bit colors are beginning to be old-fashioned, 16 or 18 bit color depths are becoming the norm, 24 bit can't be too far ahead. The main resolution for Nokia's Series 60 is currently 176 x 208, in Japan QVGA (quarter VGA, 320 x 240) is standard, Nokia Communicator (lower left) is 640 x 200, Series 90 (lower right) is 640 x 320. Soon the Japanese will no doubt upgrade from QVGA to full VGA.

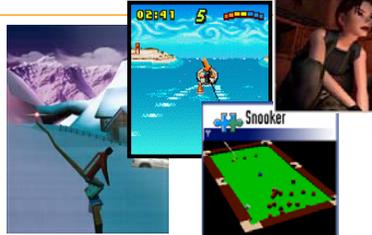
CPUs used to be tiny 10+ MHz ARM 7's, now 100-200 MHz ARM 9's are norm, pretty soon it'll be 400-600 MHz ARM 11's. It is still very rare to find hardware floating point units even in higher end PDAs, but eventually that will also be available. But the biggest problem is power. All those megahertz and increased pixel resolutions eat power, and the battery technology does not increase as fast as other components. So the amount of power in batteries compact enough to be pocketable is a limiting factor. But even if we suddenly had some superbatteries, we couldn't use all that power. More and more functionality on smaller physical size means that designing hardware so it doesn't generate hotspots that fry the electronics becomes increasingly challenging.

And memory is always a problem. Current graphics cards have 128, 256, and even more megabytes of memory, just for graphics, frame buffers, textures caches, and the like. Mobile devices have to deal with MBs that you can count with your fingers

# Mobile graphics applications



3D Menu



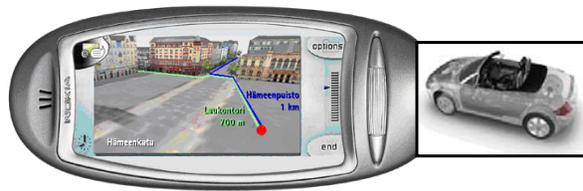
3D Games



3D Animation



3D Messaging



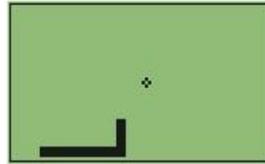
Location services

Advertising

## GSM world: State-of-the-art in 2001



- What's the world's most played electronic game?
  - The Guardian (May 2001)
- Communicator demo 2001
  - Remake of a 1994 Amiga demo
  - <10 year from PC to mobile
- Began SW 3D engine at Nokia



Around 2001, at least in Europe and Americas, the state of the art for mobile graphics was games such as Snake. Considering that in 2001 alone Nokia shipped over 100 million phones, most with Snake, with very few other games available, Snake is at least one of the most played electronic games ever.

In 2001 an old Amiga demo was ported to Nokia communicator, causing a sensation at the Assembly event in Finland.

2001 was the year Nokia started to work on their first SW 3D engine, based on a subset of OpenGL.

# State-of-the-art in 2001: Japan (from April / May)



- High-level API with skinning, flat shading / texturing, orthographic view

## GSM world: State-of-the-art in 2002

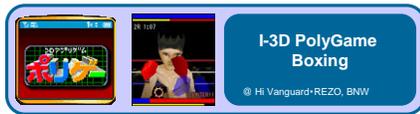


- 3410 shipped in May 2002
  - A SW engine: a subset of OpenGL including full perspective (even textures)
  - 3D screensavers (artist created content)
  - FlyText screensaver (end-user content)
  - a 3D game



# State-of-the-art in 2002: Japan

- Gouraud shading, semi-transparency, environment maps



3d menu



# 3D on GSM in 2003



- N-Gage ships
- Lots of proprietary 3D engines on various Series 60 phones
  - Starting already in late 2002



Fathammer's Geopod on XForge



# State-of-the-art in 2003: Japan

- Perspective view,  
low-level API



## Mobile 3D in 2004

---

- 6630 shipped late 2004
  - OpenGL ES 1.0 (for C++)
  - M3G (a.k.a JSR-184, for Java)
- Sharp V602SH in May 2004
  - OpenGL ES 1.0 capable HW but API not exposed
  - Java / MascotCapsule API

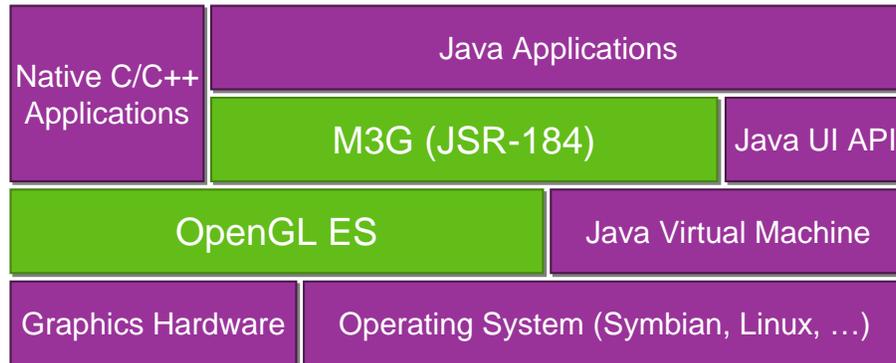


## Mobile 3D in 2005

- PSP
- Gaming phones with 3D gfx HW



# Mobile 3D APIs



The green parts show the content of today's course. We will cover two mobile 3D APIs, used by applications, either the so-called native C/C++ applications, or Java midlets (the mobile versions of applets). The APIs use system resources such as memory, display, and graphics hardware if available. OpenGL ES is a low-level API, that can be used as a building block for higher level APIs such as M3G, or Mobile 3D Graphics API for J2ME, also known as JSR-184 (JSR = Java Standardization Request).

## Overview: OpenGL ES

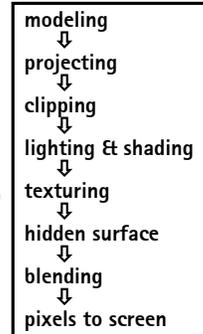
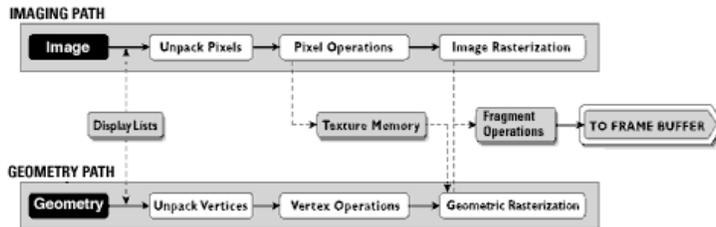
---

- Background: OpenGL & OpenGL ES
- OpenGL ES 1.0 functionality
- OpenGL ES beyond 1.0
- EGL: the glue between OS and OpenGL ES
- How can I get it and learn more?



# What is OpenGL?

- The most widely adopted graphics standard
  - most OS's, thousands of applications
- Map the graphics process into a pipeline
  - matches HW well



- A foundation for higher level APIs
  - Open Inventor; VRML / X3D; Java3D; game engines 

## What is OpenGL ES?

---



- OpenGL is just too big for Embedded Systems with limited resources
  - memory footprint, floating point HW
- Create a new, compact API
  - mostly a subset of OpenGL
  - that can still do almost all OpenGL can



## OpenGL ES 1.0 design targets

---



- Preserve OpenGL structure
- Eliminate un-needed functionality
  - redundant / expensive / unused
- Keep it compact and efficient
  - $\leq 50$ KB footprint possible, without HW FPU
- Enable innovation
  - allow extensions, harmonize them
- Align with other mobile 3D APIs (M3G / JSR-184)





SIGGRAPH2005

## Adoption

---

- Symbian OS, Series 60
- Brew
- PS3 / Cell architecture

### **Sony's arguments at GDC: Why ES over OpenGL**

- OpenGL drivers contain many features not needed by game developers
- ES designed primarily for interactive 3D app devs
- Smaller memory footprint



## Outline

---



- Background: OpenGL & OpenGL ES
- OpenGL ES 1.0 functionality
- OpenGL ES beyond 1.0
- EGL: the glue between OS and OpenGL ES
- How can I get it and learn more?

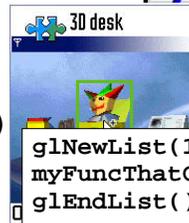
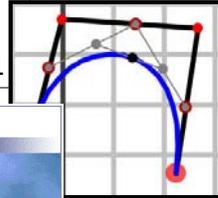


# Functionality: in / out? (1/7)

- Convenience functionality is OUT

- GLU  
(utility library)
- evaluators  
(for splines)
- feedback mode  
(tell what would draw without drawing)
- selection mode  
(for picking, easily emulated)
- display lists  
(collecting and preprocessing commands)

```
gluOrtho2D(0,1,0,1)  
vs.  
glOrtho(0,1,0,1,-1,1)
```



```
glNewList(1, GL_COMPILE)  
myFuncThatCallsOpenGL()  
glEndList()  
...  
glCallList(1)
```

## Functionality: in / out? (2/7)



SIGGRAPH2005

- Remove old complex functionality
  - glBegin – glEnd (**OUT**); vertex arrays (**IN**)
  - new: coordinates can be given as bytes

```
glBegin(GL_POLYGON);  
glColor3f(1, 0, 0);  
glVertex3f(-.5, .5, .5);  
glVertex3f(.5, .5, .5);  
glColor3f(0, 1, 0);  
glVertex3f(-.5, -.5, .5);  
glVertex3f(-.5, .5, .5);  
glEnd();
```

```
static const GLbyte verts[4 * 3] =  
{ -1, 1, 1, 1, 1, 1,  
  1, -1, 1, -1, -1, 1 };  
static const GLubyte colors[4 * 3] =  
{ 255, 0, 0, 255, 0, 0,  
  0, 255, 0, 0, 255, 0 };  
glVertexPointer( 3, GL_BYTE, 0, verts );  
glColorPointerf( 3, GL_UNSIGNED_BYTE,  
                0, colors );  
glDrawArrays( GL_TRIANGLES, 0, 4 );
```



## Functionality: in / out? (3/7)

---



SIGGRAPH2005

- Simplify rendering modes
  - double buffering, RGBA, no front buffer access
- Emulating back-end missing functionality is expensive or impossible
  - full fragment processing is **IN**  
alpha / depth / scissor / stencil tests,  
multisampling,  
dithering, blending, logic ops)



## Functionality: in / out? (4/7)



SIGGRAPH2005

- Raster processing
  - ReadPixels **IN**, DrawPixels and Bitmap **OUT**
- Rasterization
  - **OUT**: PolygonMode, PolygonSmooth, Stipple



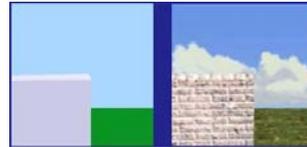
## Functionality: in / out? (5/7)



SIGGRAPH2005

- 2D texture maps **IN**

- 1D, 3D, cube maps **OUT**



- borders, proxies, priorities, LOD clamps **OUT**

- multitexturing, texture compression **IN** (optional)

- texture filtering (incl. mipmaps) **IN**

- new: paletted textures **IN**



# Functionality: in / out? (6/7)

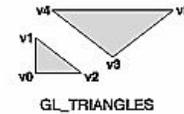
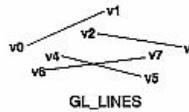
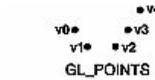
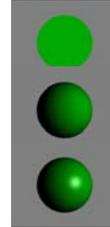
- Almost full OpenGL light model **IN**

- back materials, local viewer, separate specular **OUT**

- Primitives

- **IN**: points, lines, triangles

- **OUT**: polygons and quads



## Functionality: in / out? (7/7)

---



SIGGRAPH2005

- Vertex processing
  - **IN:** transformations
  - **OUT:** user clip planes, texcoord generation
- Support only static queries
  - **OUT:** dynamic queries, attribute stacks
    - application can usually keep track of its own state





SIGGRAPH2005

## Floats vs. fixed-point

---

- OpenGL is strongly based on floats
  - recently even frame buffers
- No HW floating-point in target devices
  - enable also low-end SW implementations
  - didn't want to wait for floating-point...





## Floats vs. fixed-point

---

- Accommodate both
  - integers / fixed-point numbers for efficiency
  - floats for ease-of-use and being future-proof
- Details
  - 16.16 fixed-point: add a decimal point inside an int

```
glRotatef(0.5f, 0.f, 1.f, 0.f );  
vs.  
glRotatex(1 << 15, 0, 1 << 16, 0 );
```

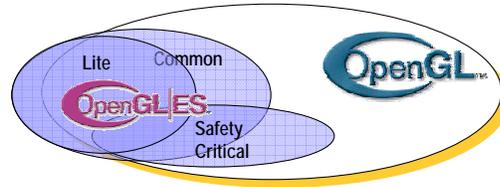
- get rid of doubles



## Profiles: Common Profile

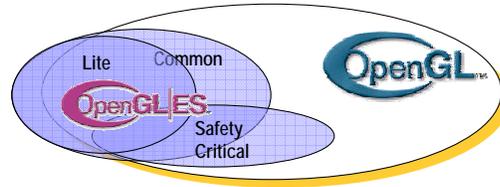


- The full OpenGL ES profile
  - both float and fixed-point function entry points
  - requires desktop OpenGL range and accuracy
- Good platform for gaming and other 3D apps
- Implementable on many platforms, including mobiles



## Profiles: Lite & Safety Critical

- Common Lite: “SW implementation-friendly”
  - for extremely limited systems
  - only fixed-point, reduced range requirements
- Safety Critical
  - key criterion: ease safety certifications
  - targeted for avionics and automotive displays





SIGGRAPH2005

## Outline

---

- Background: OpenGL & OpenGL ES
- OpenGL ES 1.0 functionality
- OpenGL ES beyond 1.0
- EGL: the glue between OS and OpenGL ES
- How can I get it and learn more?



# OpenGL ES 1.1: core

---



- **Buffer Objects**  
allow caching vertex data
- **Better Textures**  
>= 2 tex units, combine (+,-,interp), dot3 bumps, auto mipmap gen.
- **User Clip Planes**  
portal culling ( $\geq 1$ )
- **Point Sprites**  
particles as points not quads, attenuate size with distance
- **State Queries**  
enables state save / restore, good for middleware



## OpenGL ES 1.1: optional

---



- **Draw Texture**  
fast drawing of pixel rectangles using texturing units  
(data can be cached), constant Z, scaling
- **Matrix Palette**  
vertex skinning ( $\geq 3$  matrices / vertex, palette  $\geq 9$ )



## OpenGL ES 2.0

---



- Address programmability
  - Vertex and pixel shaders, GL Shading Language
  - No fixed functionality
    - no backwards compatibility
- Mobile 3D features catching up desktop fast!
  - mobile programmable API only a couple of years behind desktop



## Towards OpenGL ES 1.2

---



- Reduce variability of implementations
  - Require stencil bits, make some optional extensions mandatory, ...
  - Also some new functionality
- Announce functionality likely to be later in 1.2
  - Allows HW vendors to get ready, and gives time to get market feedback from 1.1





SIGGRAPH2005

## Outline

---

- Background: OpenGL & OpenGL ES
- OpenGL ES 1.0 functionality
- OpenGL ES beyond 1.0
- EGL: the glue between OS and OpenGL ES
- How can I get it and learn more?



## EGL glues OpenGL ES to OS



SIGGRAPH2005

- EGL is the interface between OpenGL ES and the native platform window system
  - similar to GLX on X-windows, WGL on Windows
  - facilitates portability across OS's (Symbian, Linux, ...)
- Division of labor
  - EGL gets the resources (windows, etc.) and displays the images created by OpenGL ES
  - OpenGL ES uses resources for 3D graphics





SIGGRAPH2005

## EGL surfaces

---

- Various drawing surfaces, targets for rendering
  - *windows* – on-screen rendering (“graphics” memory)
  - *pbuffers* – off-screen rendering (user memory)
  - *pixmap*s – off-screen rendering (OS native images)



## EGL context

---



- A rendering context is an abstract OpenGL ES state machine
  - stores the state of the graphics engine
  - can be (re)bound to any matching surface
  - different contexts can share data
    - texture objects
    - vertex buffer objects





SIGGRAPH2005

## Main EGL 1.0 functions

---

- Getting started
  - eglInitialize() / eglTerminate(), eglGetDisplay(), eglGetConfigs() / eglChooseConfig(), eglCreateXSurface() (X = Window | Pbuffer | Pixmap), eglCreateContext()
- eglMakeCurrent( display, drawsurf, readsurf, context )
  - binds context to current thread, surfaces, display





SIGGRAPH2005

## Main EGL 1.0 functions

---

- `eglSwapBuffer( display, surface )`
  - posts the color buffer to a window
- `eglWaitGL( )`, `eglWaitNative( engine )`
  - provides synchronization between OpenGL ES and native (2D) graphics libraries
- `eglCopyBuffer( display, surface, target )`
  - copy color buffer to a native color pixmap



## EGL 1.1 enhancements

---



- Swap interval control
  - specify # of video frames between buffer swaps
  - default 1; 0 = unlocked swaps, >1 save power
- Power management events
  - PM event => all Context lost
  - Disp & Surf remain, Surf contents unspecified
- Render-to-texture [optional]
  - flexible use of texture memory



## Outline

---

- Background: OpenGL & OpenGL ES
- OpenGL ES 1.0 functionality
- OpenGL ES beyond 1.0
- EGL: the glue between OS and OpenGL ES
- How can I get it and learn more?

# SW Implementations

---



- Gerbera from Hybrid
  - Free for non-commercial use
  - <http://www.hybrid.fi>



- Vincent
  - Open-source OpenGL ES library
  - <http://sourceforge.net/projects/ogl-es>



- Reference implementation
  - Wraps on top of OpenGL
  - <http://www.khronos.org/opengles/documentation/gles-1.0c.tgz>



## On-Device Implementations



SIGGRAPH2005

- NokiaGL (SW)
- Imagination MBX
- NVidia GoForce 3D
- ATI Imageon
- Toshiba T4G
- ...



The models shown

Nokia 6630

Dell Axim 50v

Gizmondo

LG 3600

Sharp V602SH

## SDKs

---



- Nokia Series 60 FP2 SDK (Symbian OS)
  - <http://www.forum.nokia.com>
- Imagination SDK
  - <http://www.pvrdev.com/Pub/MBX>
- NVIDIA handheld SDK
  - [http://www.nvidia.com/object/hhsdk\\_home.html](http://www.nvidia.com/object/hhsdk_home.html)
- Brew SDK & documentation
  - <http://brew.qualcomm.com>

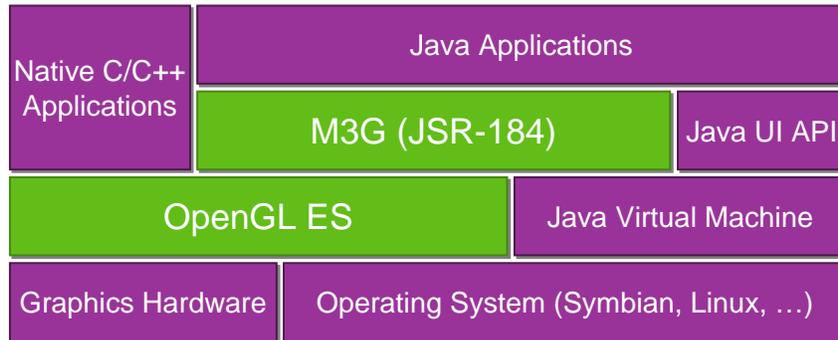


# Questions?

---



# Mobile 3D Graphics APIs



## Why a new standard for J2ME?

---



- OpenGL (ES) (and D3D) are too low-level
  - Lots of Java code needed for simple things
- Java 3D is too bloated
  - A hundred times larger than M3G
  - Does not fit together with MIDP
- Idea of subsetting Java 3D (but a new API)
  - Basic Java 3D ideas: nodes, scene graph
  - Add file format, keyframe animation
  - Remain compatible with OpenGL ES

# Status July 05 ([www.jbenchmark.com](http://www.jbenchmark.com))



SIGGRAPH2005

- **CECT**
  - GS900
- **LG**
  - MM-535
- **Motorola**
  - A780, C975, E680, E680i, E1000, i605, V980
- **Nokia**
  - 6230i, 6255, 6255i, 6630, 6680, 6681, 6682
- **Panasonic**
  - VS3
- **Samsung**
  - SGH-Z130, SGH-Z300, SGH-Z500, SPH-A880
- **Sanyo**
  - MM-7400, MM-8300, S103
- **Sharp**
  - V902sh, SX813
- **Siemens**
  - CX65, CX70, CX75, M65, M65i, S65
- **SonyEricsson**
  - F500i, K300c, K300i, K500c, K500i, K508c, K508i, K700i, K750c, K750i, S700i, V800, Z500a, Z800
- **Toshiba**
  - TS921

## Today's program

---



- **M3G Intro**  
5 min, Kari Pulli
- **M3G API overview**  
50 min, Tomi Aarnio
- **Using M3G**  
45 min, Mark Callow
- **Closing & Q&A**  
5 min, Kari Pulli
- **End at 5:30**
- **Tomi: M3G API Overview**
  - design principles
  - basic structure
  - scene graphs & animation
  - M3G file format
- **Mark: Using M3G**
  - development process, tools
  - midlets, simple -> complex
  - performance tips
  - publishing

## Closing & Summary

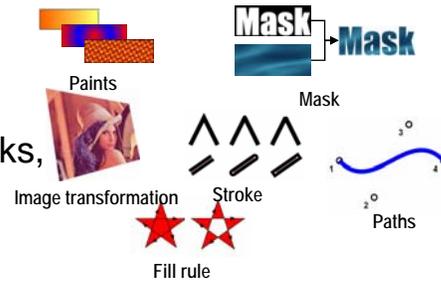
---

- We have covered
  - OpenGL ES
  - M3G

# Khronos embedded API palette



- OpenGL ES family
  - fixed functionality (1.x)
  - programmable (2.x)
- OpenVG
  - 2D vector graphics
  - SVG players, UI frameworks, low-level OS graphics



## Khronos embedded API palette

---



- OpenMAX
  - building-blocks for multimedia codecs
- Audio API
  - working group just approved
- Collada
  - interchangeable interactive 3D content
  - working group just approved

## Mobile Java

---



- M3G (JSR 184)
  - first maintenance release out
  - second generation API work can start next as OpenGL ES 2.0 is completed
- JSR 239: Java Bindings for OpenGL ES
- JSR 226: 2D vector graphics for Java
  - SVG-Tiny compatible features

## Summary

---

- Fixed functionality mobile 3D is reality NOW
  - these APIs and devices are out there
  - go get them, start developing!
- Solid roadmap to programmable 3D
  - OpenGL ES 2.0
  - M3G 2.0 work to start next winter