Ubiquitous 3D: Graphics Everywhere

Kari Pulli
Research Fellow
Nokia Research Center
MIT CSAIL

Outline

- History of mobile 3D
- Key enablers and challenges
- Mobile graphics APIs
- Current mobile HW offering
- Mobile gaming
- The way forward

State-of-the-art in 2001: GSM

- What’s the world’s most played electronic game?
  - According to The Guardian (May 2001)

    “it took Nintendo 10 years to sell 100m Game Boys whereas Nokia sold 128m handsets last year alone”

- Communicator demo
  - Assembly 2001
  - Remake of 1994 winning Amiga demo
  - ~10 year from PC to mobile

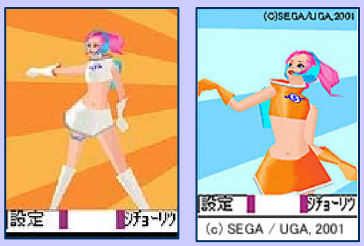
Splatting in 2001

- M-splat
  - Experiment on points as rendering primitives
  - Port of the Q-splat viewer to 9210 Communicator

- Key lesson
  - Floating point operations killed the performance
  - On integer hardware, design the whole system with fixed point maths

State-of-the-art in 2001: Japan

GENKI 3D Characters
(C) 2001 GENKI



ULALA
(c)SEGA/UGA.2001

J-SH07
by SHARP

Space Channel 5
©SEGA/UGA,2001 ©SEGA/UGA,2002

Snowboard Rider
©WOW ENTERTAINMENT INC., 2000-2002all rights reserved.

J-SH51
by SHARP

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

GSM world in 2002

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

Japan in 2002

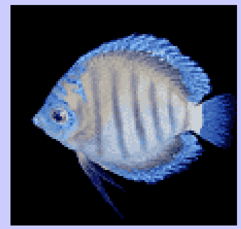
- Gouraud shading, semi-transparency, environment maps


I-3D PolyGame Boxing

@ Hi Vanguard· REZO, BNW

Ulala Channel J

©SEGA/UGA,2001 ©SEGA/UGA,2002



KDDI Au 3D Launcher

©SAN-X+GREEN CAMEL

3d menu

N-GAGE NOKIA

GSM world 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

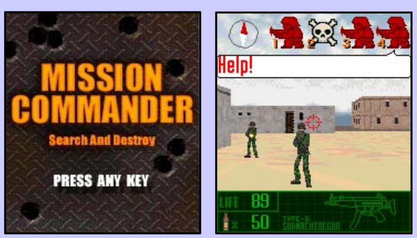
N-GAGE NOKIA

Japan in 2003

- Perspective view, low-level API

Ridge Racer

@ Namco



Mission Commander
Multi-player FPS Game

© IT Telecom

N-GAGE
NOKIA

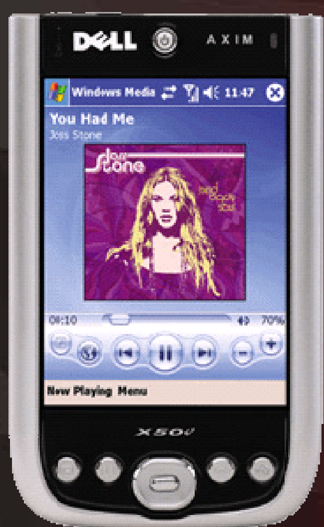
Mobile 3D in 2004

- N-Gage QD
  - better input, hot-swap MMC cards
  - several 3D titles
- 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

N·GAGE
NOKIA

Mobile 3D in 2005: With HW

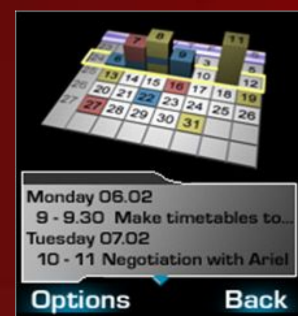
- PSP
- Gaming phones with 3D gfx HW

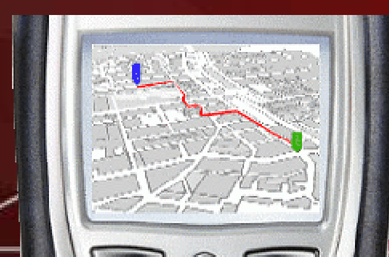


Mobile graphics use cases

- Mostly gaming
  - and other entertainment
    - screensavers, etc.
- The user interface
  - individual applications
  - application launcher
  - …
- Mapping applications



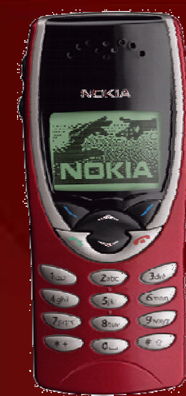





Key enablers and challenges

Changed? Displays!

- Recent improvements
  - Resolution
    - S60: 176 x 208
    - S80: 640 x 200
    - S90: 640 x 320
  - Color depth
    - Not many new B/W phones
    - 12 / 16 / 18 / … bit RGB
- Physical size remains limited
  - Near-eye micro displays in the future?



Changed? Computation!

- 3410
  - ARM 7 @ 26MHz
  - Not much caching, narrow bus
- 7650
  - ARM 9 @ ~100MHz
  - Decent caching, better bus
- 6630
  - ARM 9 @ ~200MHz
  - Faster memories
- Still no FPUs though
  - To high-end soon (at least for graphics), mid-tier later, low-end remains integer longer

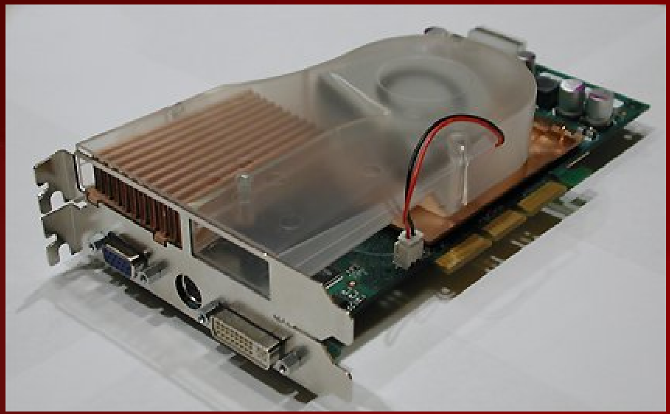
Challenge? Power!

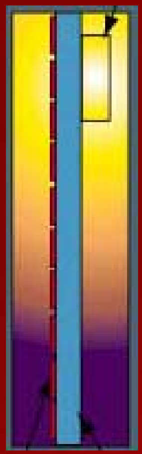
- Battery improvement doesn’t follow Moore’s law
  - Only 5-10% per year
- Gene’s law
  - "power consumption of integrated circuits decreases exponentially" over time and because of that the whole system built around chips will get smaller, and batteries will last longer
  - Since 1994, the power required to run an IC has declined 10x every two years
  - But the performance of two years ago is not enough
    - Pump up the speed
    - Use up the power savings

N-GAGE NOKIA

Challenge? Thermal mgt!

- But ridiculously good batteries still won't be the miracle cure
  - The devices are small
  - Generated power must get out
  - No room for fans
- Thermal management must be considered early in the design
  - Hot spot would fry electronics
    - Or at least inconvenience the user…
  - The heat must be conducted through the case walls, and finally removed to the ambient





www.coolingzone.com

N-GAGE NOKIA

What are phones made of?

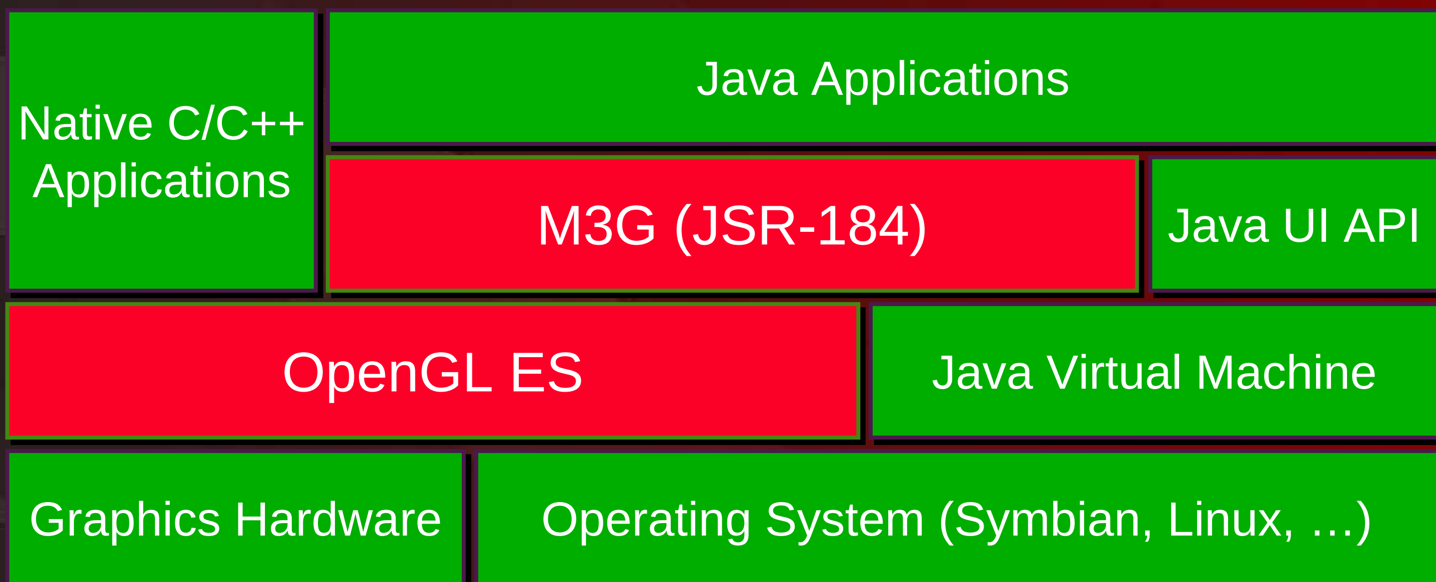
- There's an amazing amount of technology in a phone
  - CPU, sometimes more than one
  - DSP, usually several
  - Radio units, many!
  - Memory: ROM, RAM, Flash, MMC, DMA, eDRAM, …
- So how do I access the iron to get most out of it?
  - Thru APIs
    - you really don't want go hacking too close to HW
    - you might get some speed advantage, but that is fleeting
    - HW and products change so much faster than in the old hacking days of 70's and 80's
    - common APIs get you across products of a single manufacturer and also across manufacturers

N-GAGE NOKIA

Standard APIs for mobile graphics

Layering for resource reuse

- OpenGL ES and M3G
  - Designed concurrently (and partly by the same people)
  - Influenced each other
  - Layered implementation model (immediate benefit of same HW)



OpenGL ES

- Forum: KHRONOS GROUP
- OpenGL ES 1.0 design targets
  - Compactness: Eliminate un-needed functionality
    - redundant, expensive, unused
    - footprint target 50KB
  - Target both SW and HW implementations
    - don’t mandate FPU support
  - Retain good things of OpenGL
    - basic architecture
    - extensibility
    - conformance tests

OpenGL ES 1.0 in a nutshell

- Retain functionality that apps can’t emulate
  - Full fragment processing of OpenGL 1.3
- Corollary: drop convenience functionality
  - GLU, evaluators, feedback mode, selection, display lists
  - Stippling, polygons / quads, drawpixels, bitmap: all can be emulated
  - Texcoords, user clipping, can be calculated in the application
- Simplify state machine
  - Drop glBegin – glEnd, use arrays instead
  - Only RGBA (no indices), only double-buffering, draw only to back buffer
- Queries
  - Only static ones – Apps can keep track of their own state

Profiles

- OpenGL ES Common
  - Doubles => floats
  - Also support fixed point input (signed 16.16)
  - Require OpenGL (~ float) accuracy with matrix calculations
- OpenGL ES Common Lite
  - Drop floats altogether
    - Only fixed (and integers, of course)
  - Only require 16.16 fixed point accuracy with matrices, allow overflows
- OpenGL ES Safety Critical
  - For aviation, cars, etc. --- anywhere where certification required
  - Still under work

Matrices without FPUs OK!

- What makes the emulation of IEEE floats slow?
  - Special cases (such as NaN, Inf, …) take a lot of processing
  - After every operation you have to do book-keeping
    - Renormalize, i.e., shift the mantissa and update exponent accordingly
- If you know the context, you can make shortcuts
  - Matrix times vector: lots of structure ((4x4, 4x1) => 64 muls, 48 adds)
    - Assume int / fixed-point vertices, matrix elements floats
    - First do the maths on mantissas (effectively integer arithmetics)
    - Book-keeping of exponents at the end
  - Easily 10x speed win compared to emulated IEEE floats
    - Only a modest speed loss w.r.t fixed point-only implementation
    - Floats multiplied with other floats more difficult and slower…

OpenGL ES 1.1 in 2004

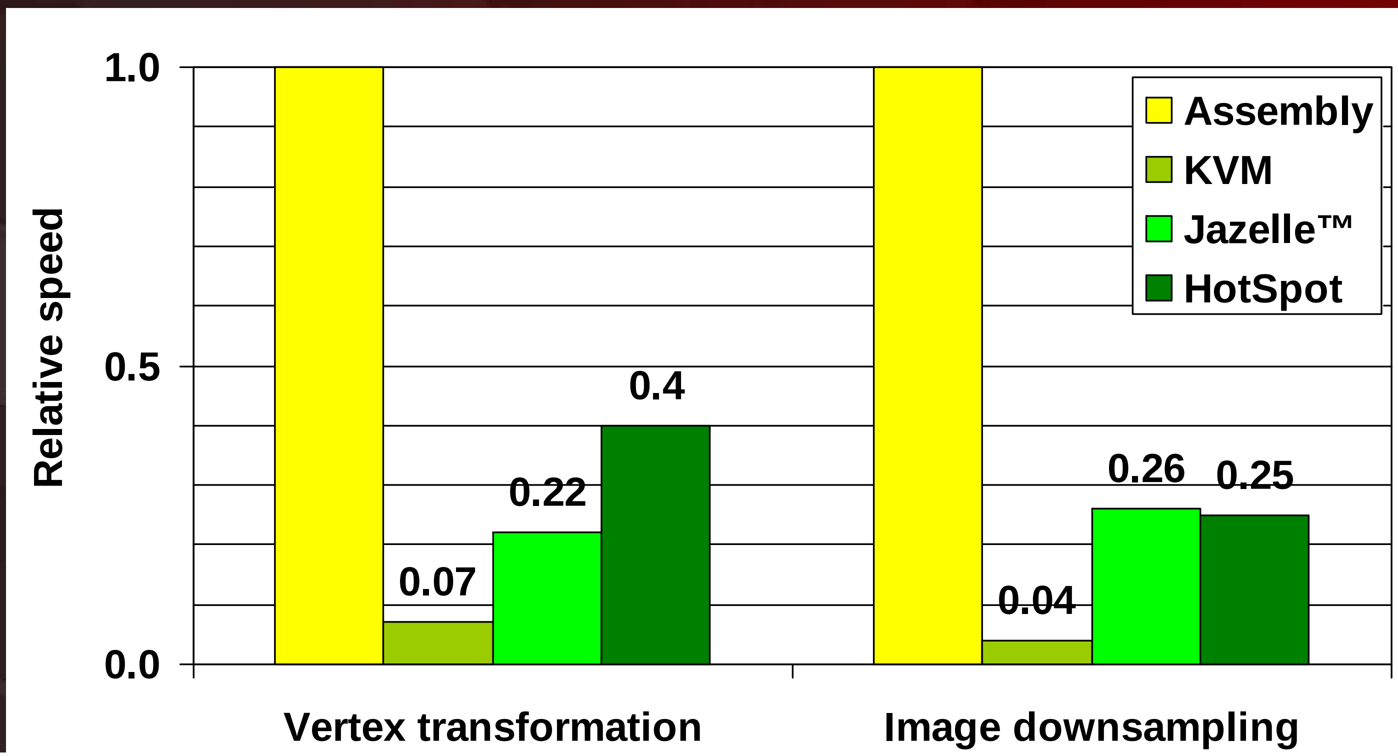
- More HW oriented than 1.0
- Buffer Objects --- allow caching vertex data
- Draw Texture --- pixel rectangles using tex units (data can be cached)
- Better Textures --- >= 2 tex units, combine (+,-,interp), dot3 bumps
- Matrix Palette --- vertex skinning (>= 3 M / vtx, palette >= 9)
- User Clip Planes --- portal culling (>= 1)
- Point Sprites --- particles as points not quads, attenuate size w/ distance
- State Queries --- enables state save / restore, good for middleware

OpenGL ES 2.0 in 2005

- Plan to release at SIGGRAPH
- Features
  - Vertex and fragment shaders
    - GL Shading language
  - Keep it compact
    - no fixed functionality
    - no backwards compatibility
  - Allow compilation both offline and on-device
- Mobile 3D feature set is catching up desktop fast!
  - Sony PS3 chose OpenGL ES 2.0 as the API
    - along with CG, Collada

What about Java?

- On desktop new hotspot compilers are pretty good
  - but on mobiles there's a clear difference between C and Java performance



Benchmarked on an ARM926EJ-S processor with hand-optimized Java and assembly code

M3G (JSR-184)

Need a higher level API

- A game is much more than just 3D rendering
  - Objects, properties, relations (scene graph)
  - Keyframe and other animations
  - Etc. (game logic, sounds, …)
- If everything else but rendering is in Java
  - A very large percentage of the processing is in slow Java
  - Even if rendering was 100% in HW, total acceleration remains limited
- A higher level API could help
  - More of the functionality could be implemented in native (=faster) code
  - Only the game logic must remain in Java

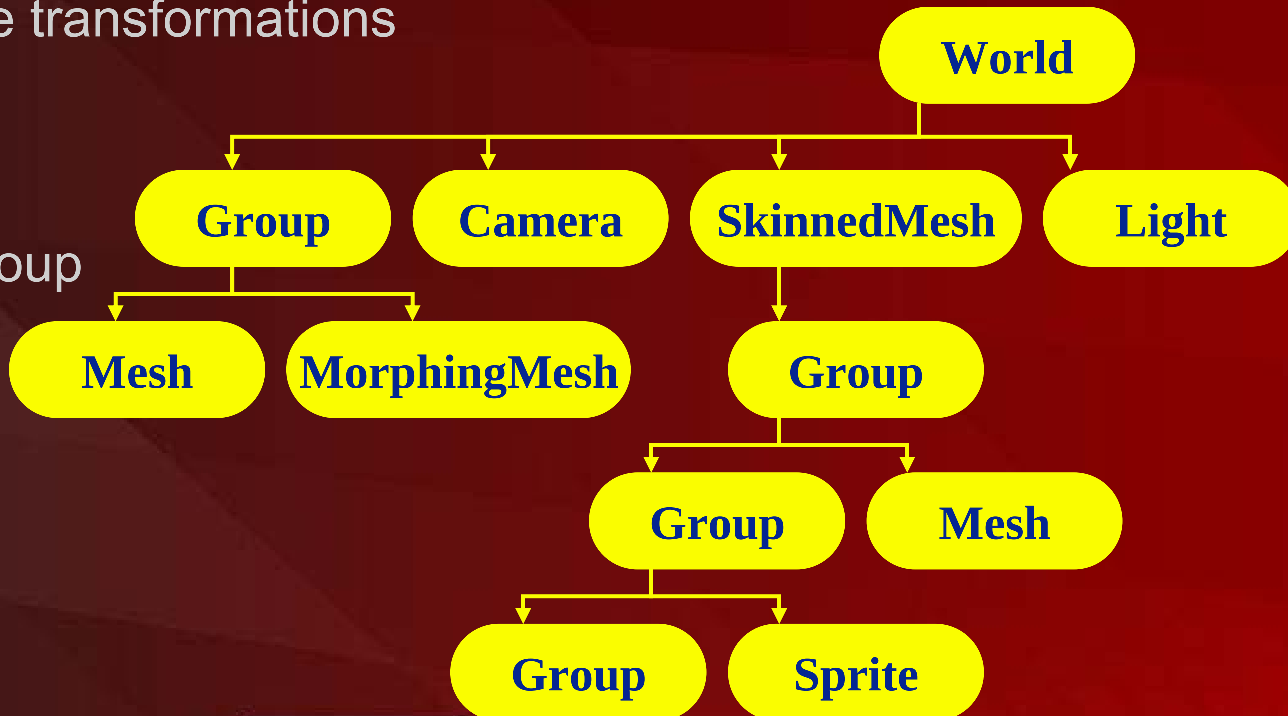
Java3D ES? No, M3G!

- Java3D seemed a good starting point
  - But "Java3D ES" didn't work out
  - Java3D was designed for large-resource systems
    - Java3D distribution is ~40MB (~300x too big for us)
  - Didn't really fit together with MIDP
    - a large redesign necessary
- M3G (JSR-184), a new API
  - Nodes and scene graph
  - Extensive animation support
  - Binary file format and loader

M3G
!=
Java3D

Scene graphs from nodes

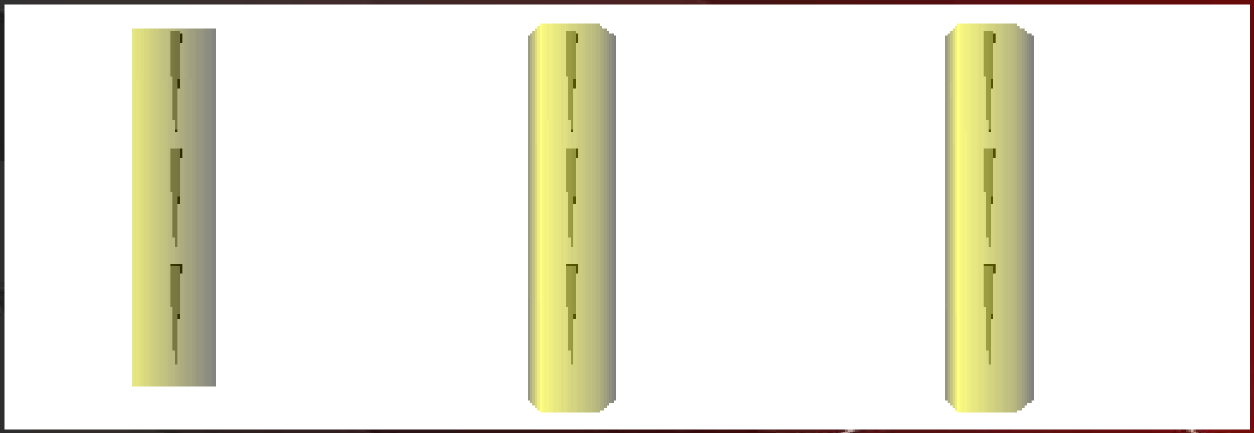
- The tree encodes structure
  - Data (vertices, textures, animation data, …) can be shared
  - Nodes encode relative transformations and inherited alpha
- World is the root
  - A special case of a group
- Other nodes
  - Camera
  - Light
  - Mesh
  - Sprite



M3G (JSR-184)

Special meshes

- SkinnedMesh
  - For articulated motions
  - Bones have weighted associations to vertices
- MorphingMesh
  - For unarticulated animations
  - Base mesh, targets, weighted interpolations towards / away from targets



M3G (JSR-184)

Everything can be keyframed

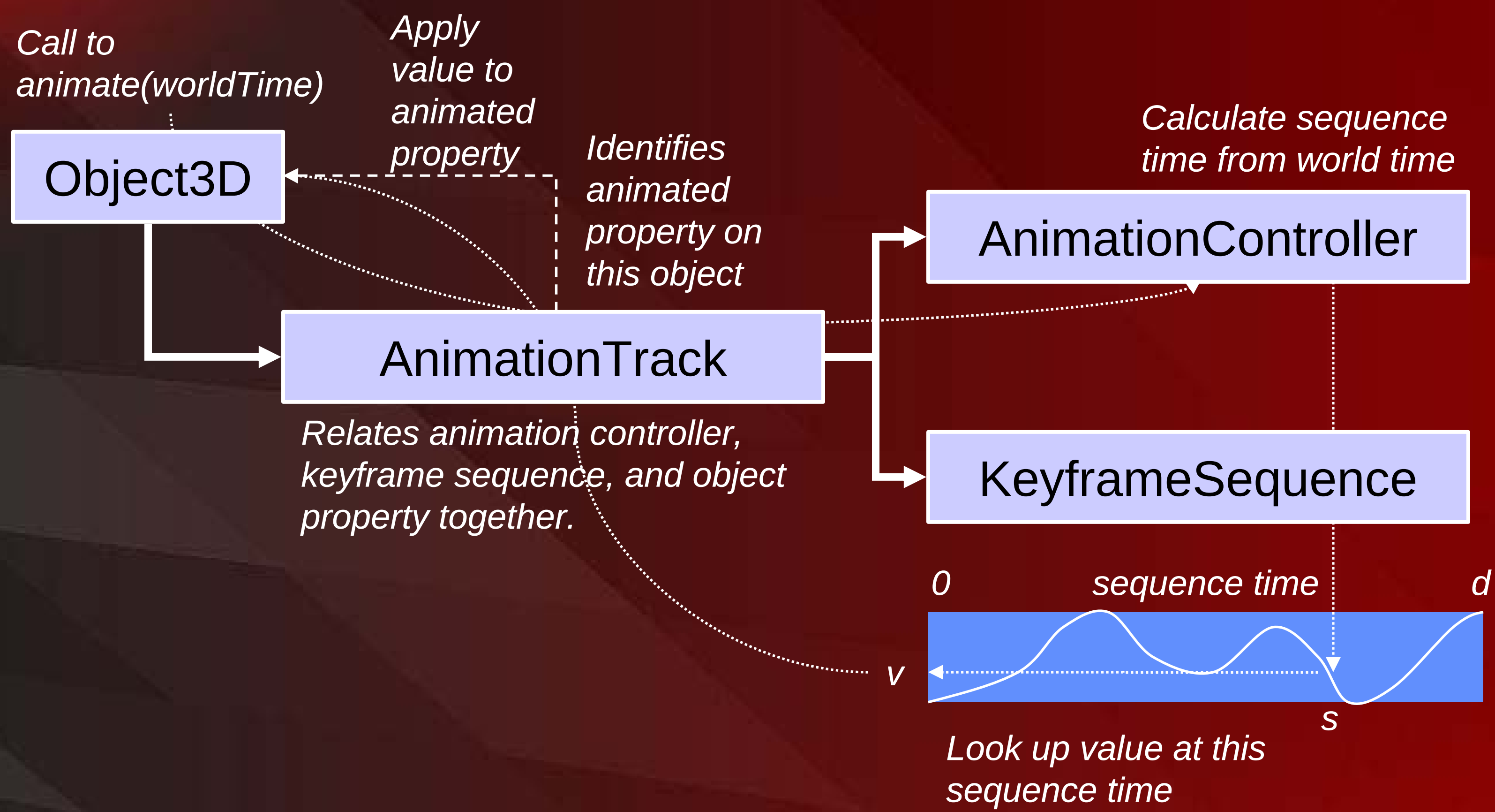
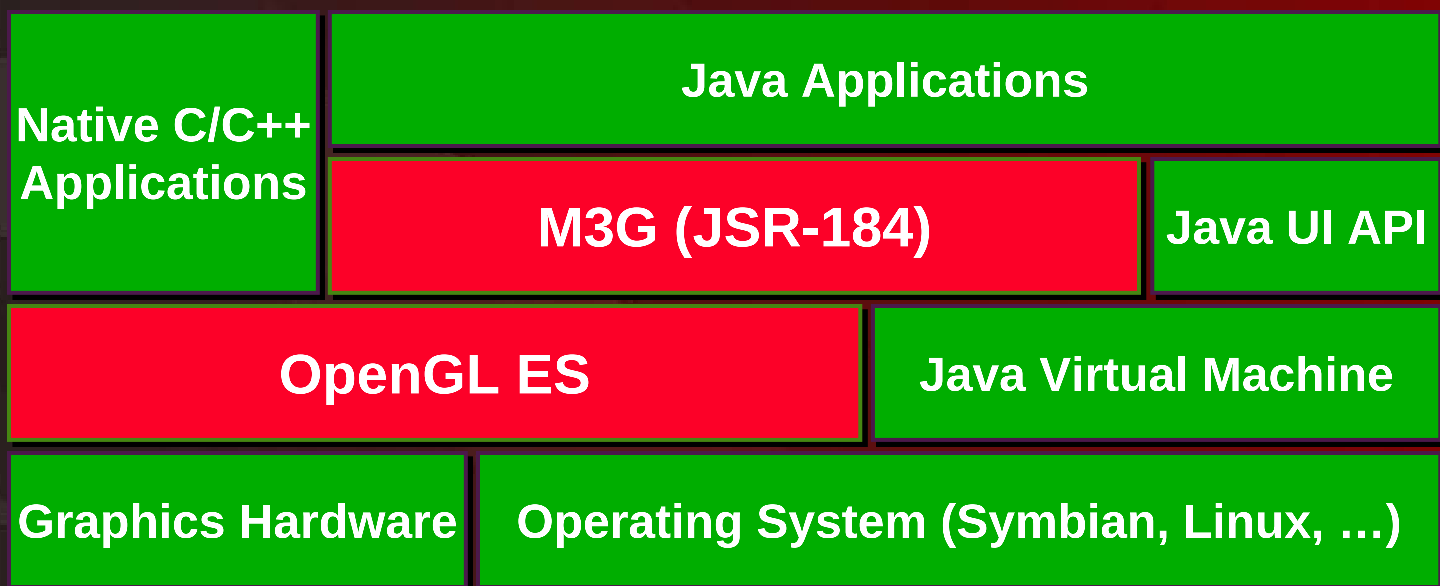


Diagram courtesy of Sean Ellis, Superscape

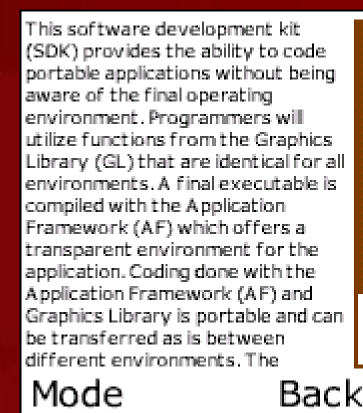
Layering for resource reuse

- OpenGL ES and M3G
  - Designed concurrently (and partly by the same people)
  - Influenced each other
  - Layered implementation model (immediate benefit of same HW)



What about 2D vector gfx?

- Need for low-level 2D vector graphics primitives
  - Portable mapping and GPS, E-book readers and text packages
  - Games
  - Advanced user interfaces and screen savers, even GDIs
- Many vector graphics formats in use
  - Flash, SVG, PDF, Postscript, vector fonts, PPT, etc.
- Scalable graphics = Easier to port content



OpenVG fills the need

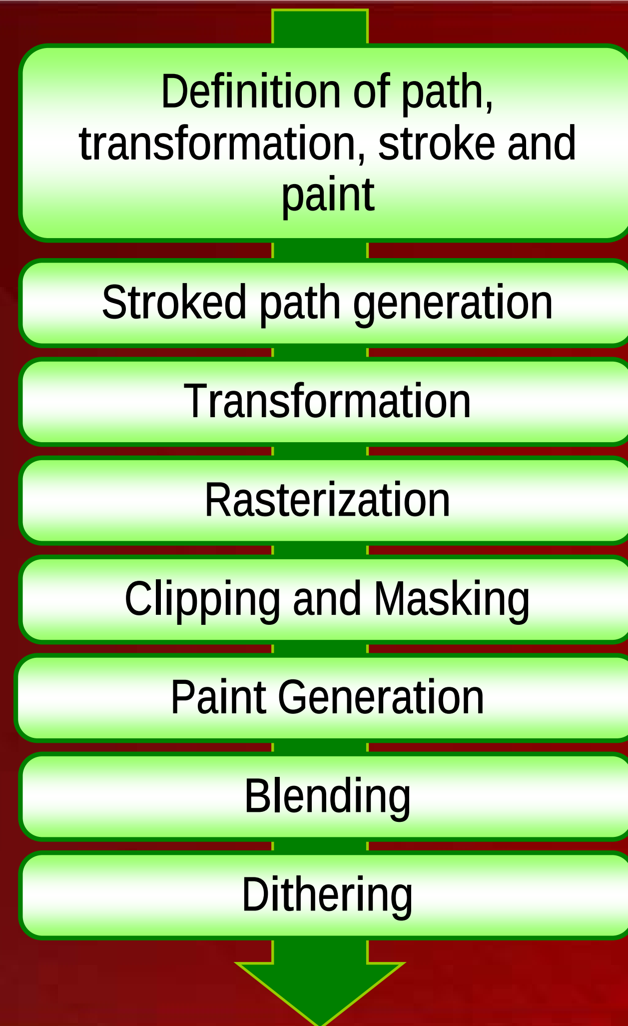
- Need to offload 2D vector graphics operation from the CPU
- Need to save power when running 2D vector graphics
- Need of open standard API

OpenVG™

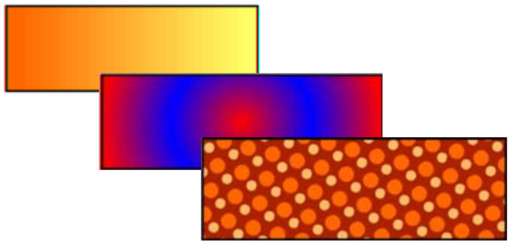
Open Standard for Vector Graphics Hardware Acceleration

OpenVG: Features

- Core API
  - Coordinate systems and transformations
    - image drawing uses a 3x3 perspective transformation matrix)
  - Paths
  - Images
  - Image filters
  - Paint (gradient and pattern)
  - Blending and masking
- The VGU Utility Library
  - Higher-level geometric primitives
  - Image warping



OpenVG: Features



Paints

Mask

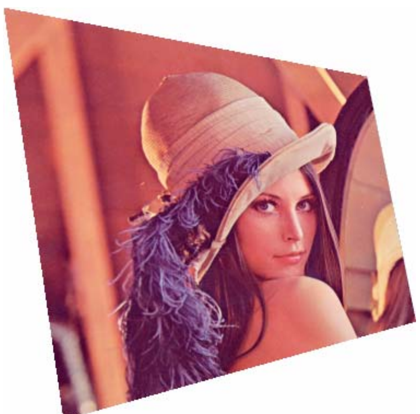
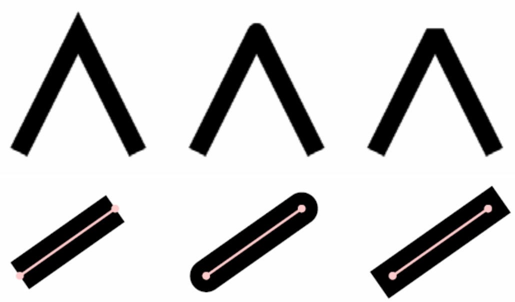
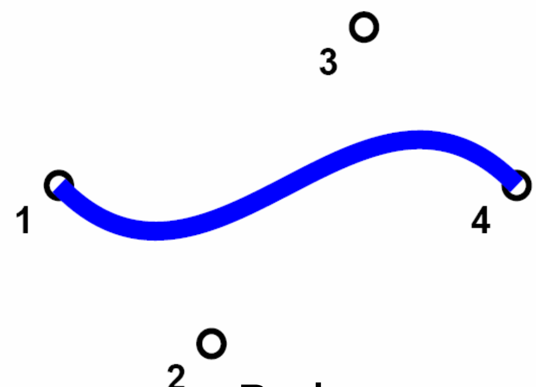


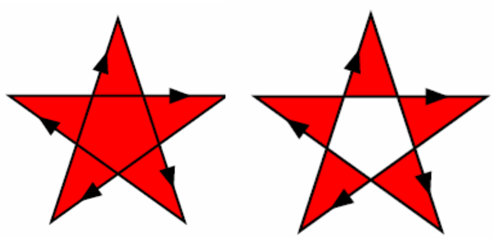
Image transformation



Stroke

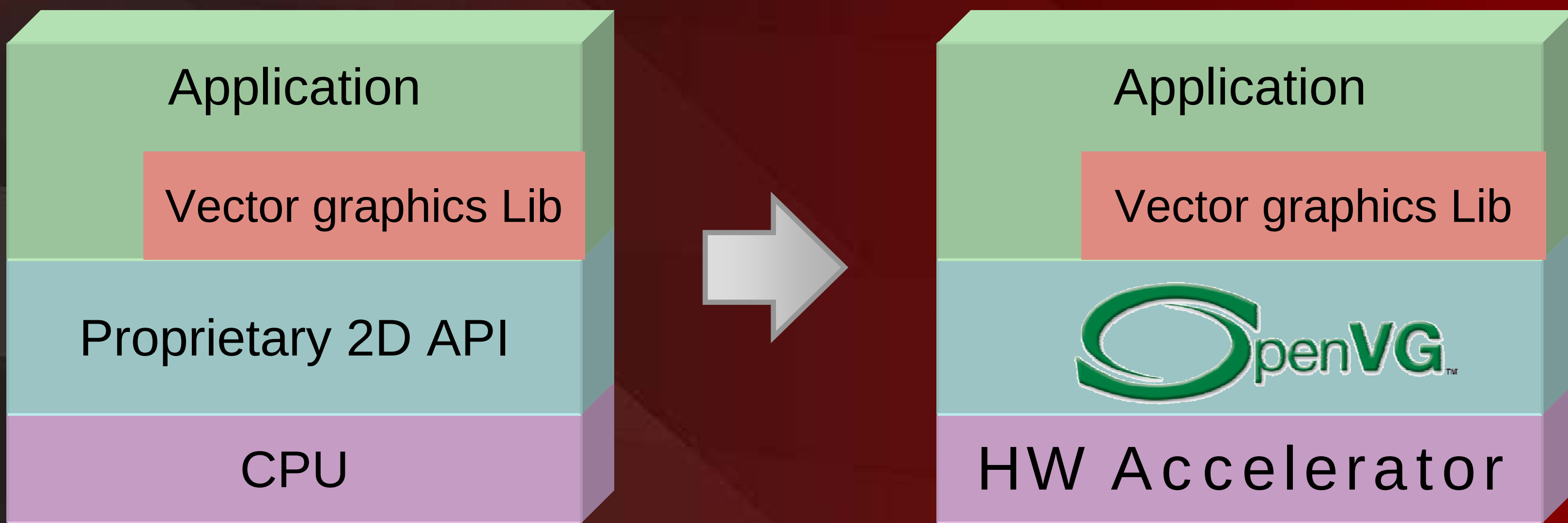


Paths



Fill rule

Where it fits in



2D Vector Graphics Today

2D Vector Graphics with OpenVG

Mobile 3D HW now!

Current HW offering

- There's quite a bit out there available for licensing
  - e.g., ATI, BitBoys, Falanx, Imagination Technologies, Mitsubishi, Nvidia, Toshiba, …
  - … and many others
- Marketing performance figures in the following pages
  - Scaled to 100MHz
  - Usually tri/s means vtx/s, actual number of triangle setups is sometimes taken into account, sometimes not, some numbers estimated some measured, MHz vary, …
  - So don't take the numbers too seriously

ATI

- Imageon 2300
  - OpenGL ES 1.0
  - Vertex and raster HW
    - 32 bit internal pixel pipe
    - 16 bit color and Z buffers
  - 100M pix/s, 1M tri/s @ 100 MHz
  - Integrated QVGA frame buffer
  - Imaging / video codecs
- Qualcomm / Imageon 3D
  - OpenGL ES 1.1
  - Vertex shader, 2x multitex
  - 100M pix/s, 3M tri/s @ 100 MHz
- Imageon 2^{nd} gen. 3D
  - OpenGL ES 1.2
  - Vertex shader (T&L, skinning)
  - Cube map, projective textures, stencils, advanced texture combiners
  - HyperZ
  - 100M pix/s, 4M tri/s @ 100 MHz
  - 3D audio
- Partners / Customers
  - Qualcomm, LG SV360, KV 3600

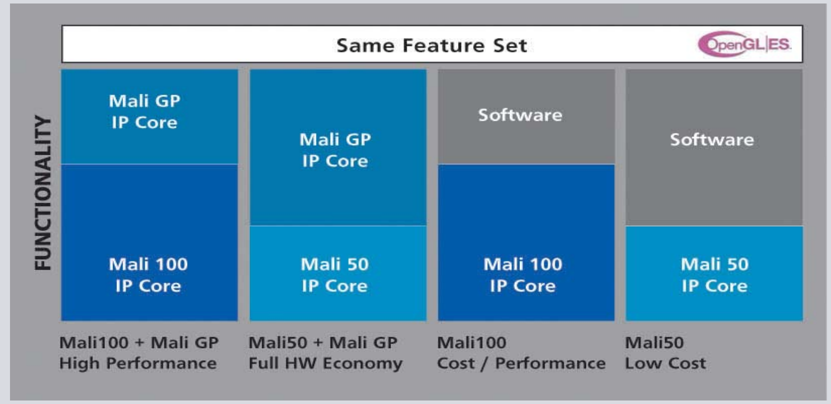
IMAGEON

Bitboys

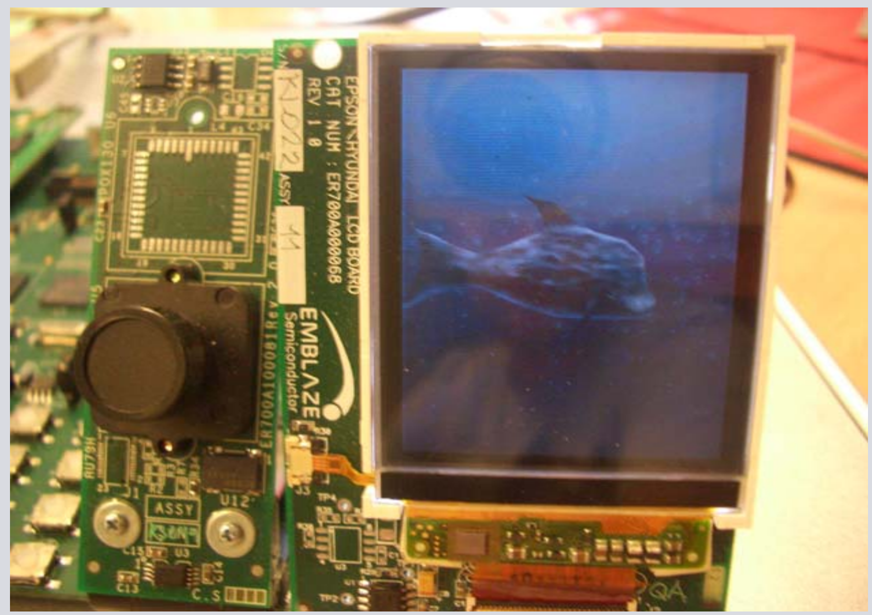
- Graphics processors
  - G12: OpenVG 1.0
  - G34: OpenGL ES 1.1
    programmable geometry processor
  - G40: OpenGL ES 2.0, GLSL
    OpenVG 1.0
    programmable geometry
    and pixel processors
  - Flipquad antialiasing
  - Max clock 200MHz
- Partners / Customers
  - NEC Electronics
  - Hybrid Graphics (drivers)

Falanx

- The Mali Family of IP Cores
  - OpenGL ES 1.1 + Extensions
  - 4X / 16X Full Scene Anti-Aliasing
  - Video Encoding / Decoding (e.g MPEG4)
  - 170 – 400k Logic Gates + SRAM
  - Performance 100MHz Mali100 + Mali Geometry
    - 2.8M tri / s
    - 100M pix / s with 4X FSAA
- Partners / Customers
  - Zoran



Mali Family of IP Cores



Falanx ARM9 / Mali100 rendering
Dot3 bump mapped fish

Imagination Technologies

- MBX
  - OpenGL ES 1.1, raster HW
  - 400M pix / s, 120 mW (@ 100 MHz)
- VGP [Vertex Geometry Processor]
  - Vertex HW, programmable
  - 2.1M tri / s (@ 100 MHz)
- Tile-based architecture
  - Buffer triangles
  - Rasterize a block at a time
    - Deferred everything, hires color, FSAA
- Partners / Customers
  - ARM, Samsung, TI (OMAP 2),
    Renesas (SH-Mobile3), Intel 2700G, …

PowerVR Technologies is a division of Imagination Technologies Ltd.

Mitsubishi

- Z3D family
  - Z3D and Z3D2 out in 2002, 2003
    - Pre-OpenGL ES 1.0
    - Embedded SRAM architecture
  - Current offering Z3D3
    - OpenGL ES 1.0, raster and vertex HW
    - Cache architecture
    - @ 100 MHz: 1.5M vtx / s, 50-60 mW, ~250 kGates
  - Z3D4 in 2005
    - OpenGL ES 1.1
- Partners / Customers
  - Several Japanese manufacturers

Z3D

First mobile 3D HW?

Z3D Graphics Engine

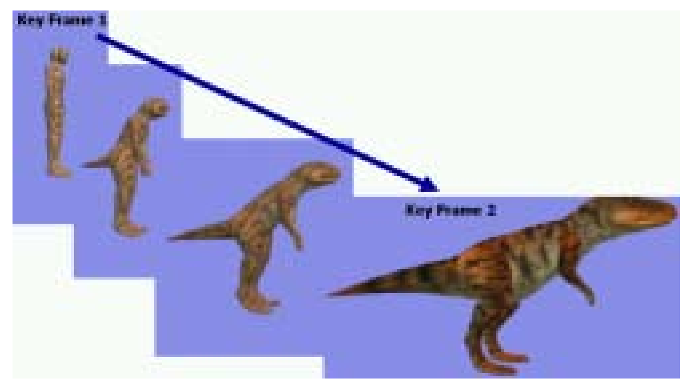
NVidia

- GoForce 3D 4800 multimedia companion chip
  - AR 15 3D IP Core
    - OpenGL ES 1.1 / MD3D, OpenVG 1.0
    - Geometry engine and rasterizer in HW
    - Programmable dot3 pixel shader
    - 40 bit signed non-int (overbright) color pipeline
    - Dedicated 2D engine (bitblt, lines, alpha blend)
    - Supersampled AA, up to 6 textures (8 surfaces)
    - 1.4M vtx|tri / s, 100M pix / s (@ 100 MHz)
    - <50mW avrg. dynamic power cons. for graphics
  - 3MPxl camera support, video
- Partners / Customers
  - ARM

GoForce 4800 Dawn

Sony PSP

- Game processing unit
  - Surface engine
    - tessellation of Beziers and splines
    - skinning (<= 8 matrices), morphing (<= 8 vtx)
    - HW T&L
    - 21 MTri / s (@ 100 MHz)
  - Rendering engine
    - basic OpenGL-style fixed pipeline
    - 400M pix / s (@ 100 MHz)
  - 2MB eDRAM
- Media processing engine
  - 2MB eDRAM
  - H.264 (AVC) video up to 720x480 @ 30fps
  - VME reconfigurable audio/video decoder



Toshiba

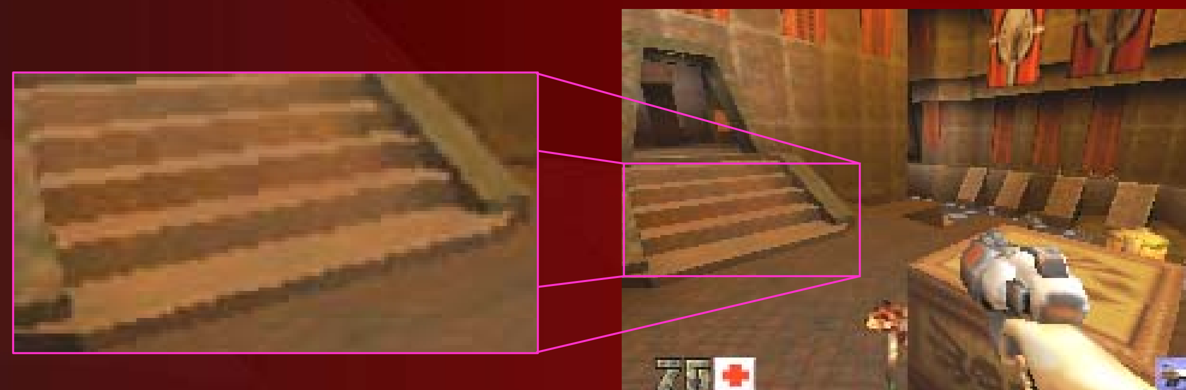
- T5G
  - OpenGL ES 1.1
  - Raster and vertex HW, fixed functionality
  - Large embedded memory for
    - Color and Z buffer
    - Caches for vertex arrays, textures
    - Display lists (command buffer)
  - Cube maps, aniso textures, 2-stage multi tex, …
  - 1.2M vtx / sec (@ 100 MHz)
  - 100M pix / sec (@ 100 MHz)
  - 87 mW @ 100 MHz (incl. eDRAM)
- Partners / Customers
  - Sharp, Toshiba’s own phones, Vodafone

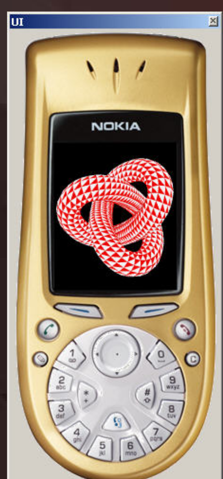
What's different in mobile HW?

- Fewer pixels on the screen
  - Fill rates don't have to be as high
  - Quality matters more
- Power usage
  - Internal bandwidth is the key
- Cost
  - High parallelism => large silicon size => high cost
    - won't work on mobiles
    - at least if we really want to make graphics ubiquitous

Anti-aliasing

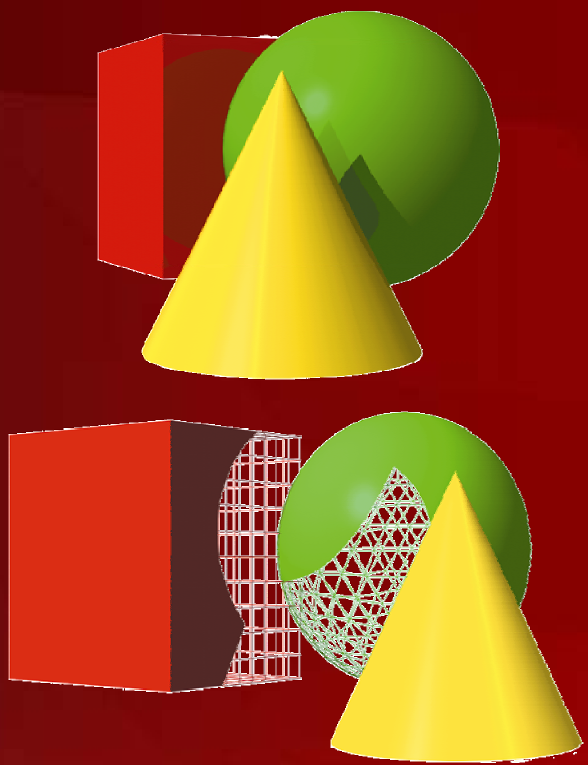
- Even (especially?) on a small screen
  - Few pixels, make the most out of them
  - Even when pixels get small, high frequency noise is annoying
    - especially in animation





Output sensitivity

- Do high-quality and low power mix?
  - Work smart rather than hard
- Work only on visible pixels
  - Avoid read-modify-write
    - Those memory accesses use a lot of power!
    - Especially important with complicated pixel shaders
  - Visibility culling
  - Deferred shading, texturing



N·GAGE
NOKIA

Local read-modify-write

- Tiled rendering
  - Store the triangles into a buffer
  - Process triangles one screen tile at a time in local memory
  - Need auxiliary buffers, extra storage for AA, etc., for only one tile
  - Then block-copy out to display buffer

- Put the whole frame into eDRAM
  - It's also possible to have a tile for the whole frame
    - though screen resolutions are growing…
  - No need to buffer triangles
  - But large embedded memories are expensive

Avoid also other traffic

- Send less data to the engine
  - Caching
    - textures, vertex arrays
  - Texture compression
  - Better pixel processing gives same or better visual quality cheaper
    - Example: bump maps (supported already in OpenGL ES 1.1)

N-GAGE NOKIA

Avoid traffic, period.

- Procedural everything
  - Ultimate compression!
    - Both for transmission *and* storage in handset *and* btw CPU and GPU
  - Send only seed data, generate rest in the shaders
  - Procedural *textures* in the pixel shader instead of texture lookups
    - Possible with current shaders
  - Procedural *geometry* in the vertex shader from control points
    - Regular tessellation becoming possible, adaptive later
  - Procedural *animation*
    - Natural phenomena (water, smoke, fire, …)
    - Constraints, IK, solved at vertex shader
      - instead of keyframing everything

Example: KKrieger

- A fully procedural FPS (first person shooter)
  - Dynamic lighting, nice textures, monsters, music & effect sounds, …
- This image (800 x 400 gif) takes 160 KB
  - The whole kkrieger exe file is ~96KB!

Avoid any other waste

- Unused HW blocks cost
  - Clock still eats power
    - Even with clock gating there's leakage current from the silicon
    - Voltage islands avoid leakage current, but can't be turned on/off in the middle of pipeline at quick demand
- Efficient use of hardware
  - Example: don't use all of 3D HW for 2D UI
  - Example: dynamically scale voltage & frequency based on loads
  - Example: unified shaders
    - With separate shaders, within the same object, let alone the scene, the bottleneck varies between the vertex and pixel processing
    - Unified shaders are a very efficient use of HW

N-GAGE NOKIA

Mobile gaming

Gaming still the driving force

- Gaming is the force that moves the PC industry
  - Both the CPUs / systems and graphics cards
  - Lately multimedia has been pretty effective too…
- One of the key technology drivers for mobiles as well
  - But is mobile gaming somehow different?

Always ready

- Many people carry a phone all the time
- Implications of combined phone / gaming device
  - Always on
  - Always ready
  - Always connected
  - Can play when you have time
    - waiting, commuting, …
    - but with good games people actually take time

N·GAGE NOKIA

Social gaming

- Lots of traditional gaming is exclusive
  - Play alone at home
  - Or with people you can't see
- Sometimes people have game evenings
  - Many players close to each other connected by the net
  - An inclusive, social event
- Short range connectivity
  - Brings the social gaming out of living room

N-Gage QD – Vital Statistics

N-Gage ecosystem

- Growing game portfolio
  - over 45 titles
- Arena provides services
  - find other players
  - post hi-scores
  - etc.

N-GAGE NOKIA

A bigger picture

“The Gamers’ Phone” devices

Native gaming for Series 60

SNAP Mobile Java

A concept from E3

A concept from E3

N·GAGE NOKIA

Demo

The way forward:
Where are we going?

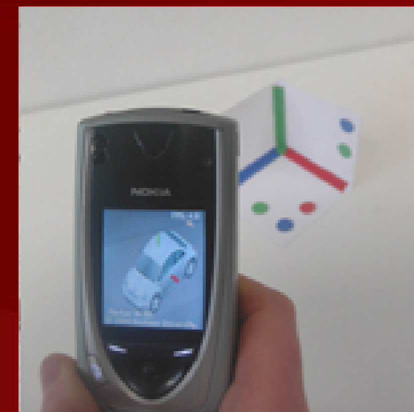
Where are we going?

- Performance vs. quality
  - First get performance that is “adequate”
  - Then work on improving quality
- Desktop has often taken brute-force approach
  - Can’t do so on mobiles, must be smarter
  - But many tricks that are must on mobiles are also useful on desktop
    - So mobile graphics can affect desktop
- New features will appear soon also on mobiles
  - See OpenGL ES 2.0
  - Later perhaps at the same time, why not even sooner
  - More active players on mobile than on desktop

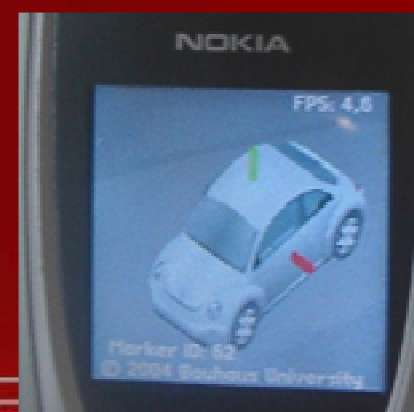
N·GAGE NOKIA

Where are we going?

- Graphics and image processing are merging
  - Life-like models are difficult to create
  - Lots of image/video/etc. papers at SIGGRAPH
- Most new mobile phones have cameras
  - With megapixel class resolutions
  - Real-time video encoding at decent resolution coming
- Add things up
  - Mobile computation
  - Location awareness
  - Connectivity
  - High-quality displays
  - Real-time graphics
  - Real-time image processing



→ AR



O.Bimber
U. Weimar

Ideas and material from

Ed Plowman, ARM
Micky Aleksic et al., ATI
Petri Nordlund et al., Bitboys
Mark Callow, HI Corp
Nicolas Thibieros et al., Imagination
Miroslaw Bober et al., Mitsubishi
Tim Leland et al., NVidia
Sean Ellis, Superscape
Masafumi Takahashi et al., Toshiba
Neil Trevett, NVidia

Nokia

- Tomi Aarnio
- Panu Brodkin
- Ilkka Harjunpää
- Tapio Hill
- Jani Karlsson
- Jarkko Kemppainen
- Tapani Leppänen
- Jouka Mattila
- Koichi Mori
- Kimmo Roimela
- Jani Vaarala

And many others: Thank You!

N-GAGE NOKIA