Graphics Everywhere – Pixels in your Pocket

Kari Pulli
Research Fellow
Nokia Research Center

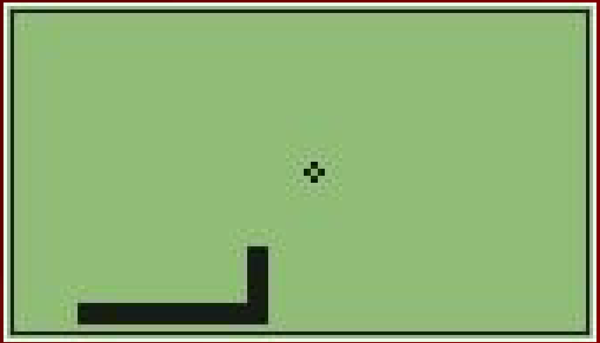
Outline

- History of 3D on Nokia / GSM
- Key enablers and challenges for mobile 3D
- Mobile 3D APIs
- Current Mobile HW offering
- Use cases for mobile 3D graphics
- Mobile gaming
- What’s important in mobile 3D graphics
- The way forward

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

- 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 difference from PC to mobile
- Began SW 3D engine at Nokia



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

- The fruits of the labor in 2002
  - A SW engine implementing a subset of OpenGL
  - 3410 shipped in May 2002
  - The applications were
    - Downloadable 3D screensavers (artist created content)
    - FlyText screensaver (end-user created content)
    - a 3D game

Top-of-the-line 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

Top-of-the-line 3D on GSM in 2004

- N-Gage QD
  - Better input keys
  - Hot-swap of MMC cards
- 6630 will ship in late 2004
  - OpenGL ES 1.0 (for C++)
  - M3G (aka JSR-184, for Java)

Key enablers and challenges

What has 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 bit RGB, 16 bit RGB, ...
- Physical size remains limited
  - Near-eye micro displays in the future?

What has changed? Computation!

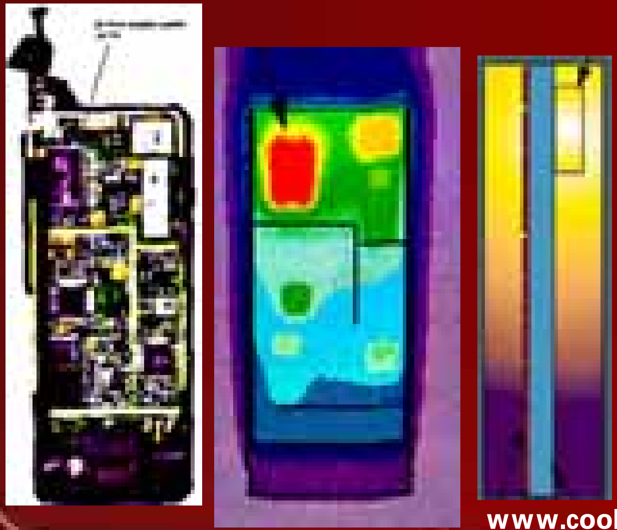
- 3410
  - ARM 7 @ 26MHz
  - Not much caching, narrow bus
- 7650
  - ARM 9 @ ~100MHz
  - Decent caching, better bus
- 6630
  - ARM 9 @ ~200MHz
  - Faster memories
- Still no FPUs though
  - To high-end soon (at least for graphics), mid-tier later, low-end remains integer longer

Remains a 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

Remains a 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

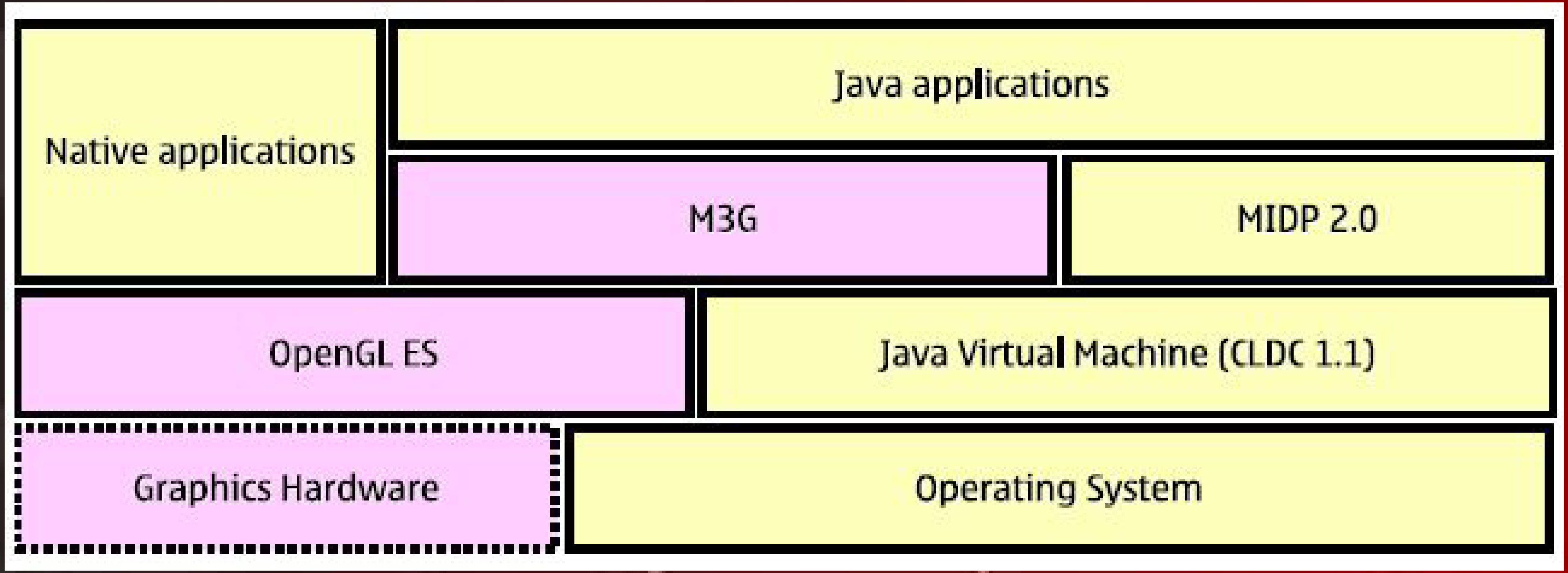


www.coolingzone.com

New standard APIs for mobile 3D

Layered model --- reuse of resources

- 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
  - Apps can keep track of their own state
  - Only static ones

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

OpenGL matrices without FPUs OK!

- What makes the emulation of IEEE fp 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 context you can make shortcuts
  - Matrix times vector: lots of structure ((4x4, 4x1) => 64 muls, 48 adds)
    - Lets also assume vertices are ints for fixed, matrix elements floats
    - First do the maths on mantissas (effectively integer arithmetics)
    - Then do the book-keeping of exponents at the end
  - Easily 10x speed win compared to emulated IEEE fp
    - 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 --- for SIGGRAPH 2004

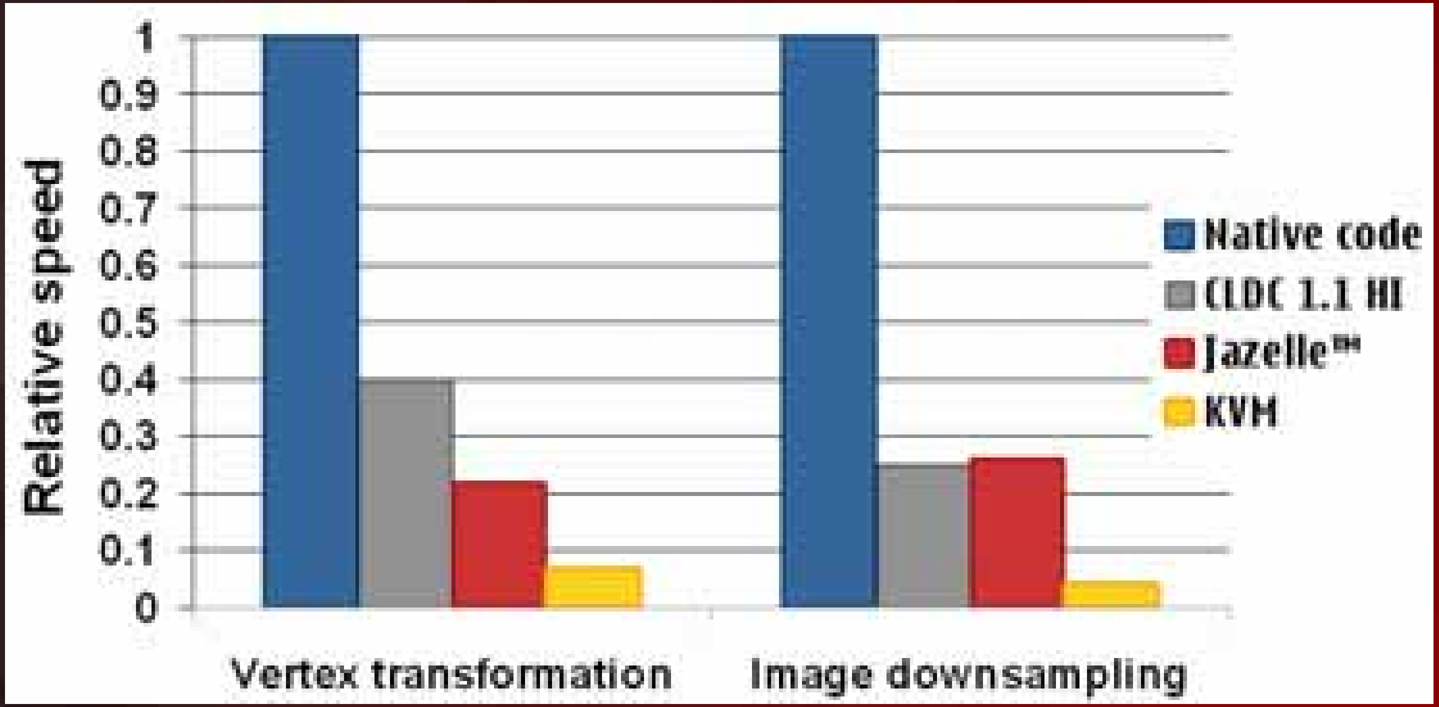
- More HW oriented than 1.0
- Buffer Objects --- allow caching vertex data
- Draw Texture --- pixel rectangles using tex units (data can be cached)
- Better Textures --- >= 2 tex units, combine (+,-,interp), dot3 bumps
- Matrix Palette --- vertex skinning (>= 3 M / vtx, palette >= 9)
- User Clip Planes --- 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 for SIGGRAPH 2005?

- Address programmability
  - Assume vertex and pixel shaders
  - GL Shading language
- Work just starting, but possibly
  - Really stripped-down version of OpenGL 2.0
    - no fixed functionality
  - No backwards compatibility
  - Should the shader compiler be on the device?
- Mobile 3D feature set is catching up desktop fast!

What about Java?

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



Need for a higher abstraction level

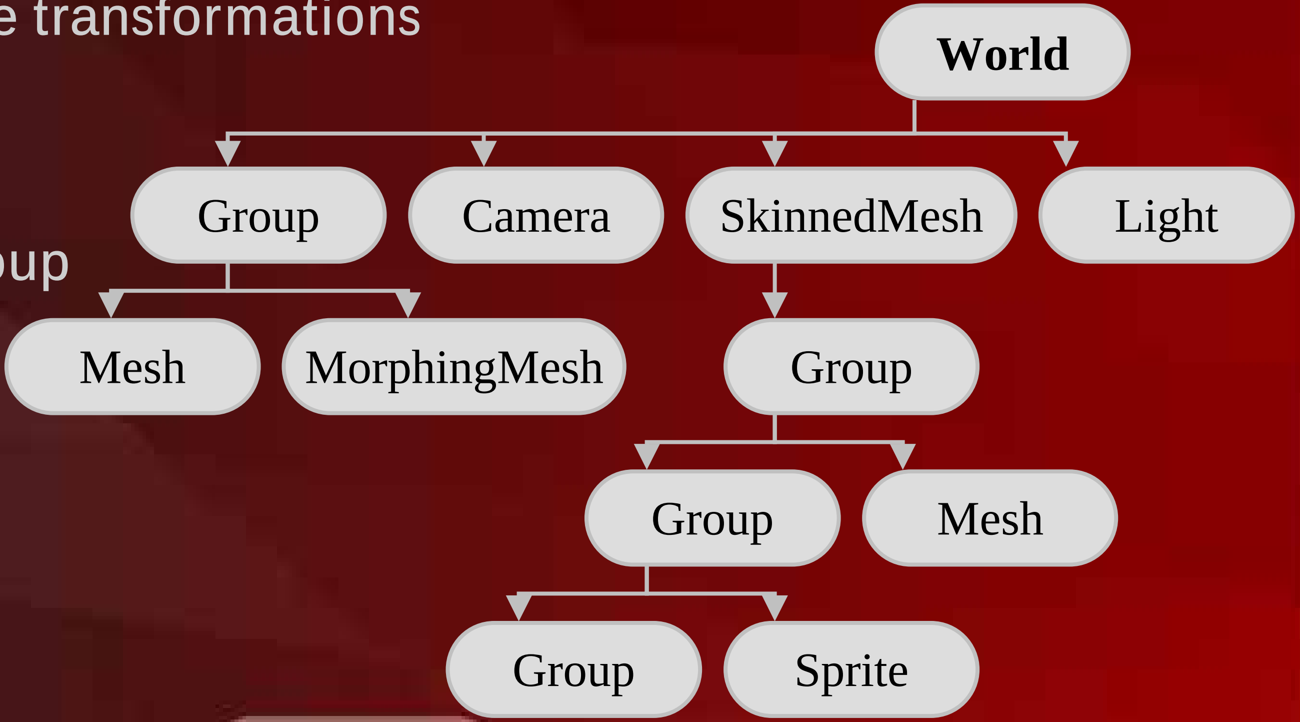
- A game is much more than just 3D rendering
  - Objects, properties, relations (scene graph)
  - Keyframe and other animations
  - Etc. (game logic, sounds, ...)
- If everything else but rendering is in Java
  - A very large percentage of the processing is in slow Java
  - Even if rendering was 100% 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?

- Java3D seemed to be a good starting point
  - But “Java3D ES” didn’t work out
  - Java3D was really 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

Scene graphs are made of nodes

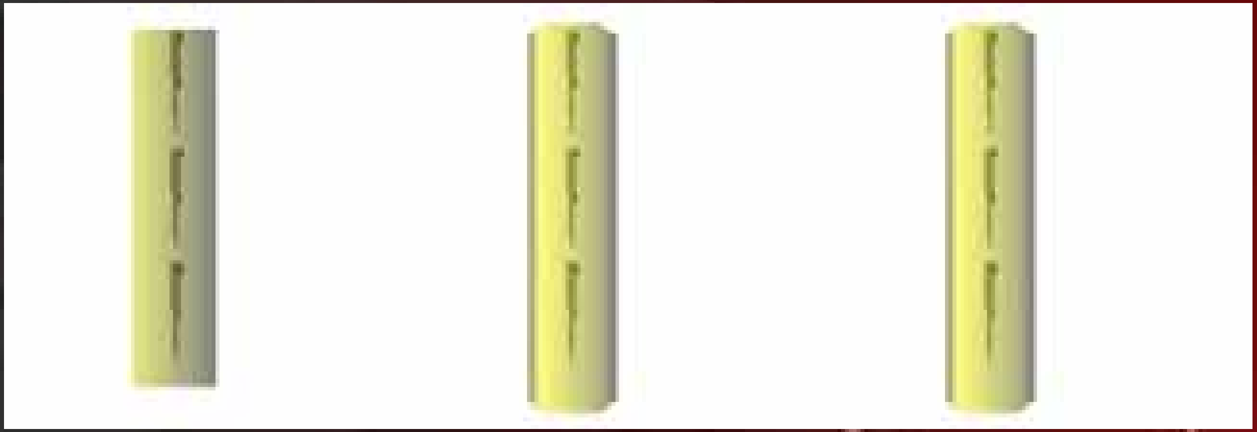
- The tree encodes structure
  - the actual data (vertices, textures, animation data, ...) can be shared
  - Nodes encode relative transformations and inherited alpha
- World is the root
  - a special case of a group
- Other nodes
  - camera
  - light
  - mesh
  - sprite



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



Everything can be keyframe-animated

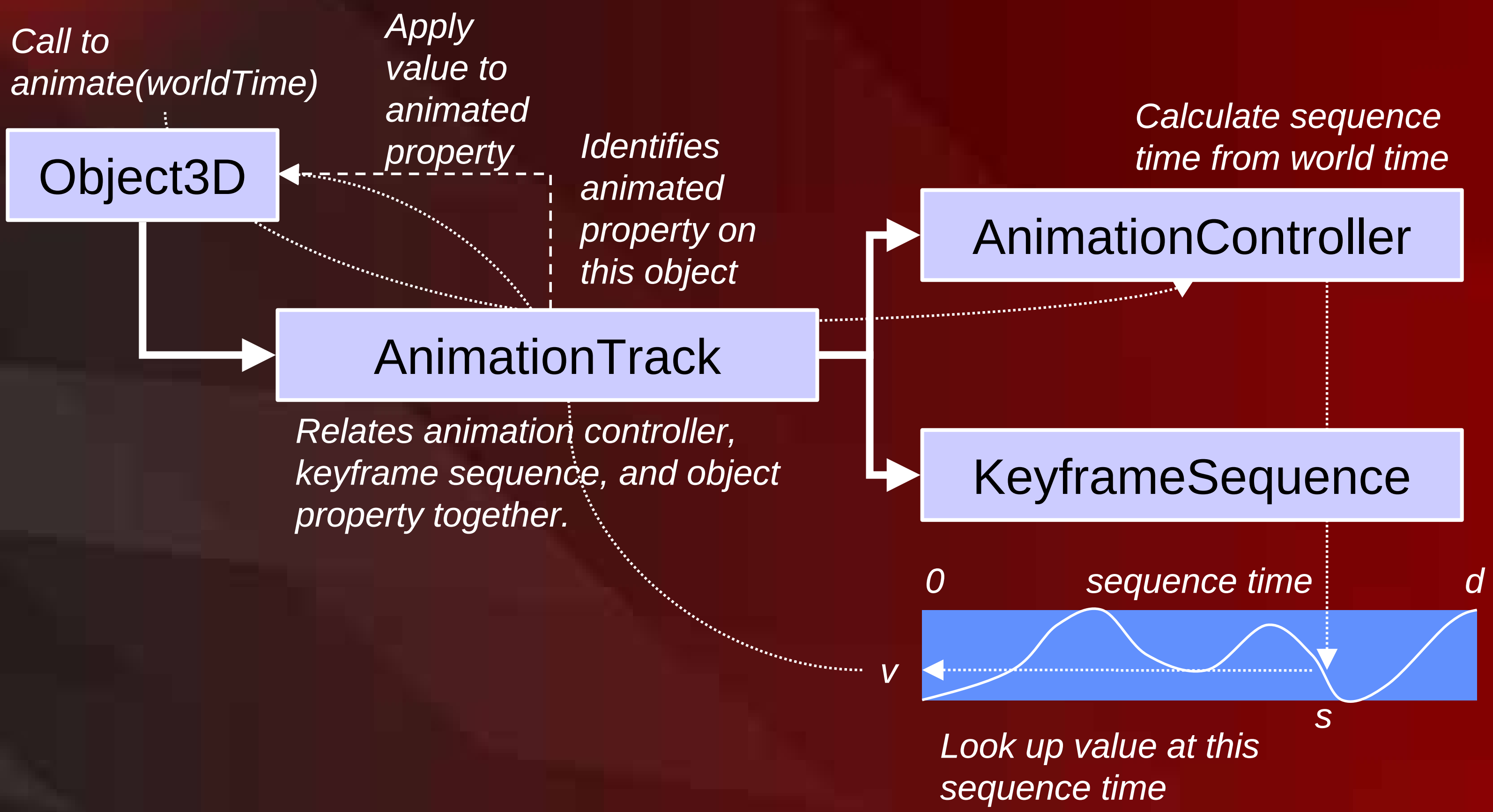
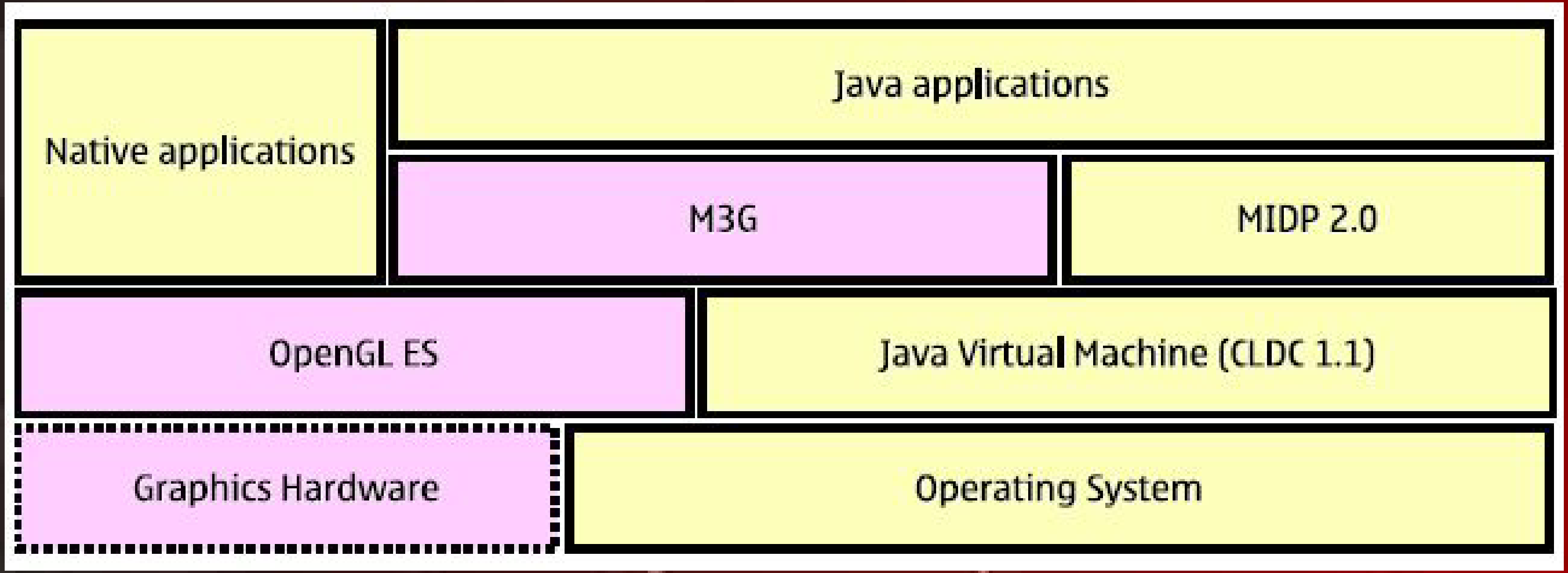

Call to animate(worldTime)
Apply value to animated property
Identifies animated property on this object
Calculate sequence time from world time
Object3D
AnimationController
AnimationTrack
KeyframeSequence
Relates animation controller, keyframe sequence, and object property together.
0
sequence time
d
v
s
Look up value at this sequence time

Diagram courtesy of Sean Ellis, Superscape

Layered model --- reuse of resources

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



Mobile 3D HW now!

Current HW offering

- There's quite a bit out there available for licensing
  - ATI, BitBoys, Falanx, Imagination Technologies, Mitsubishi, Nvidia, Toshiba
  - and probably a handful of 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

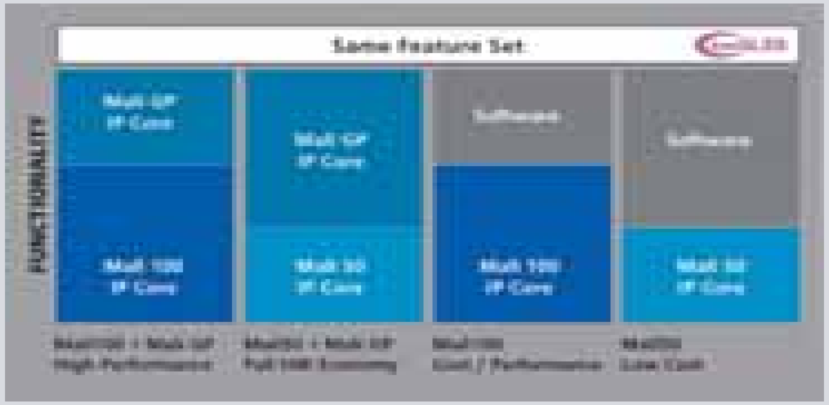
- Imageon 2300
  - OpenGL ES 1.0
  - Vertex and raster HW
    - 32 bit internal pixel pipe
    - 16 color and Z buffers
  - 100M pix/s, 1M tri/s @ 100 MHz
  - Integrated frame buffer (up to QVGA)
  - Imaging / video codecs
- Qualcomm / Imageon 3D
  - 100M pix/s, 3M tri/s @ 100 MHz
  - 2x multitex
- Partners / Customers
  - Qualcomm

Bitboys

- Graphics processors
  - G30: OpenGL ES 1.0, raster HW
  - G32: OpenGL ES 1.1 (multitex), raster HW
  - G34: OpenGL ES 1.1, vertex shaders
  - G40: ~OpenGL ES 2.0 (vertex and pixel shaders)
  - Flipquad aa
- Partners / Customers
  - NEC Electronics (G34)
  - 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



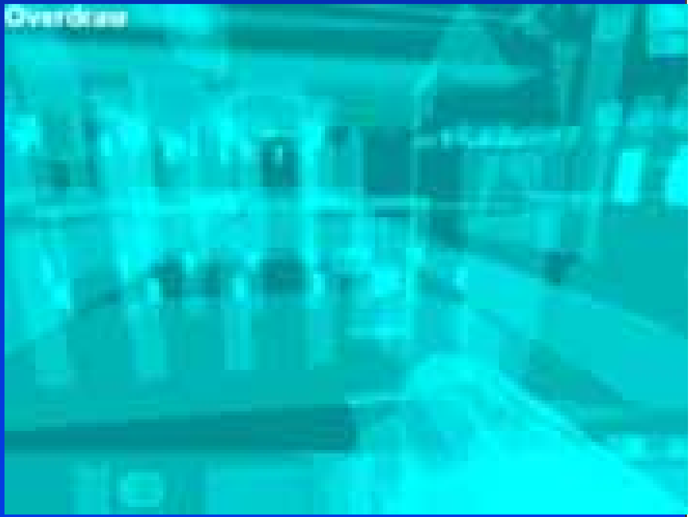
Mali Family of IP Cores

Falanx ARM9 / Mali100 rendering
Dot3 bump mapped fish

Imagination Technologies

POWERVR 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, ...



Imagination TECHNOLOGIES

PowerVR Technologies is a division of Imagination Technologies Ltd.

Mitsubishi

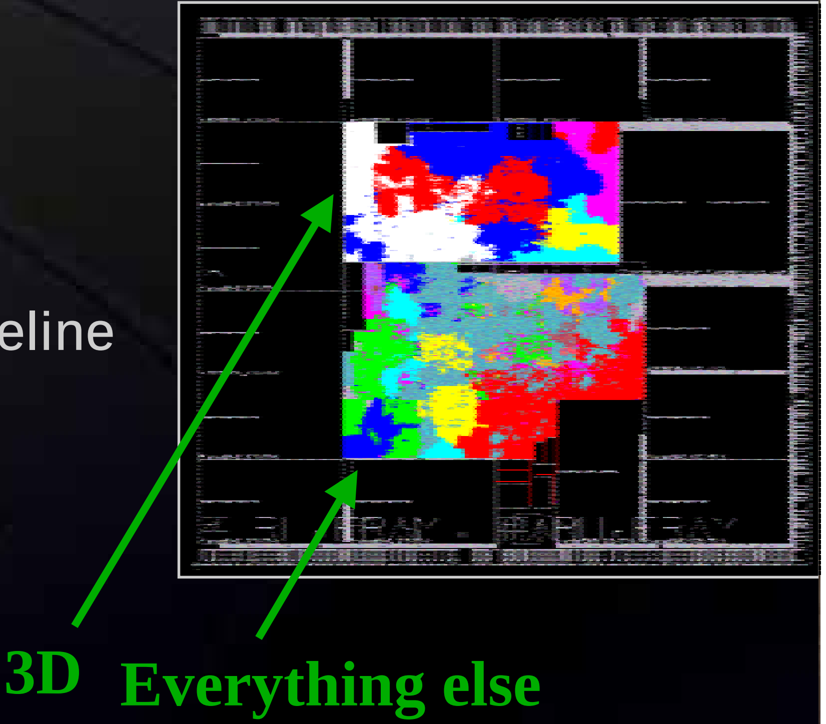
- Z3D family
  - Z3D and Z3D2 out in 2002, 2003
    - Pre-OpenGL ES 1.0
    - Embedded SRAM architecture
  - Current offering Z3D3
    - OpenGL ES 1.0, raster and vertex HW
    - Cache architecture
    - @ 100 MHz: 1.5M vtx / s, 50-60 mW, ~250 kGates
  - Z3D4 next year
    - OpenGL ES 1.1
- Partners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

Z3D

First mobile 3D HW?

NVidia

- GoForce 3D / SC10 / AR 10
  - OpenGL ES 1.1
  - Raster and vertex HW
  - Camera support, video encode/decode
  - 40 bit signed non-int (overbright) color pipeline
  - Programmable pixel shader
  - Supersampled aa, up to 6 textures
  - 1.4M vtx|tri / s, 100 Mpxl / s (@ 100 MHz)
- Partners / Customers
  - None announced yet



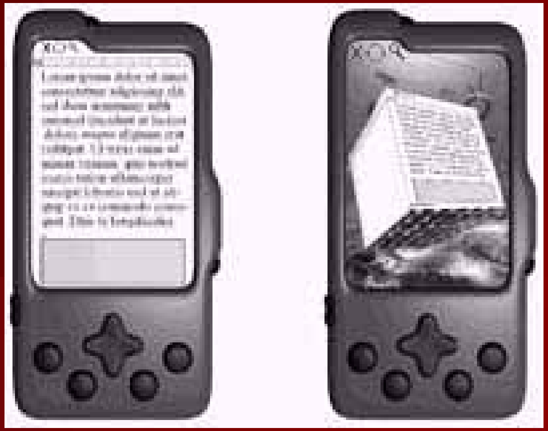
Toshiba

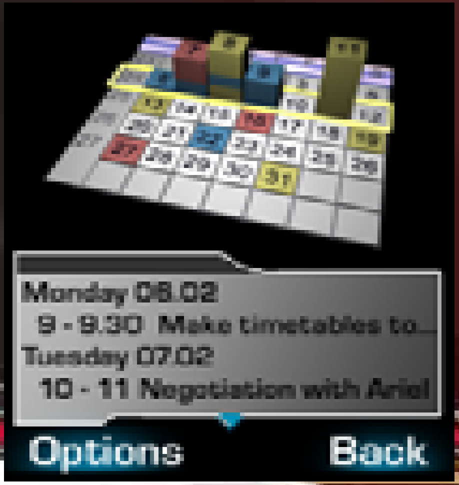
- T4G
  - OpenGL ES 1.0
  - Raster and vertex HW, fixed functionality
  - Large embedded memory for
    - Color and Z buffer
    - Caches for vertex arrays, textures
    - Display lists (command buffer)
  - Cube maps, aniso textures, 2-stage multi tex, ...
  - 1.2 Mvtx / sec (scaled to 100 MHz)
  - 100 Mpxl / sec (scaled to 100 MHz)
  - 136 mW @ 100 MHz (incl. eDRAM)
- Partners / Customers
  - Sharp, Toshiba's own phones, Vodafone

Standards and HW:
What for?

UIs

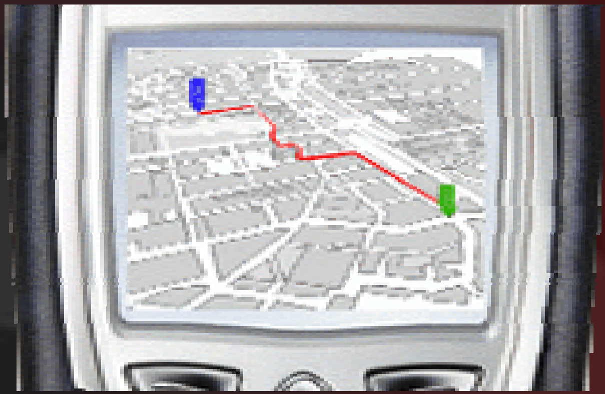
- Pack more data to small screen
- Cute animated icons
- Cute application launchers
- Create a landscape for easier data navigation

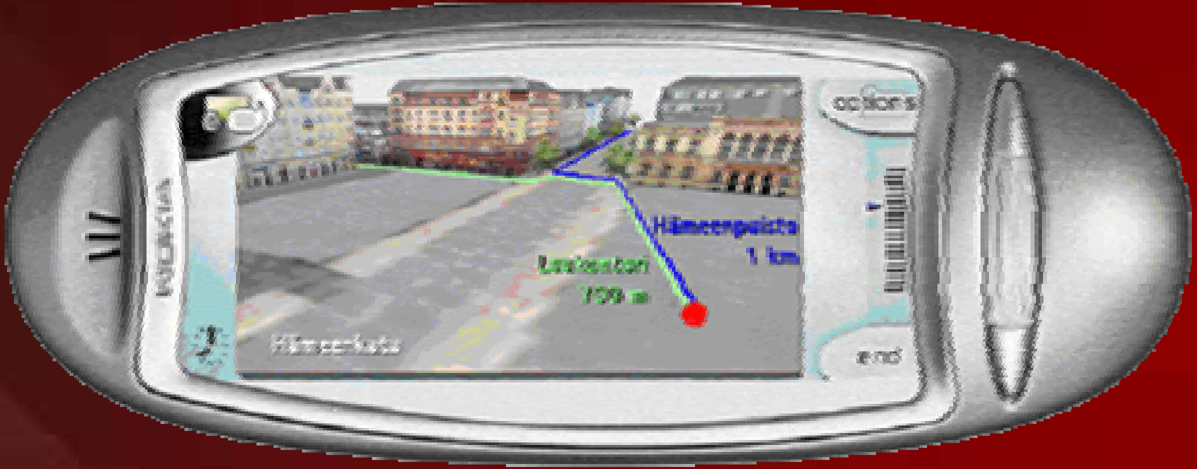



Monday 08.02
9 - 9.30 Make timetables to...
Tuesday 07.02
10 - 11 Negotiation with Ariel
Options
Back

Mapping, navigation

- These are *mobile* devices, after all

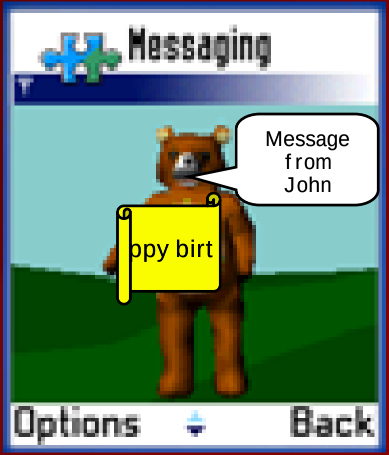




- Now www.mapquest.com, etc. begin to make sense!

Messaging, screen savers

- 3D messages?
  - If it's just pre-made Hallmark-type e-cards, how exciting is that?
  - Research problem:
    How can regular users create 3D messages on a phone, with no mouse, hardly a keyboard?
    - Wizards, really powerful sketching
- 3D screen savers?
  - Now who would want to pay for those?
  - Answer: ringing tones, still & animated logos, etc., is a multimillion dollar business



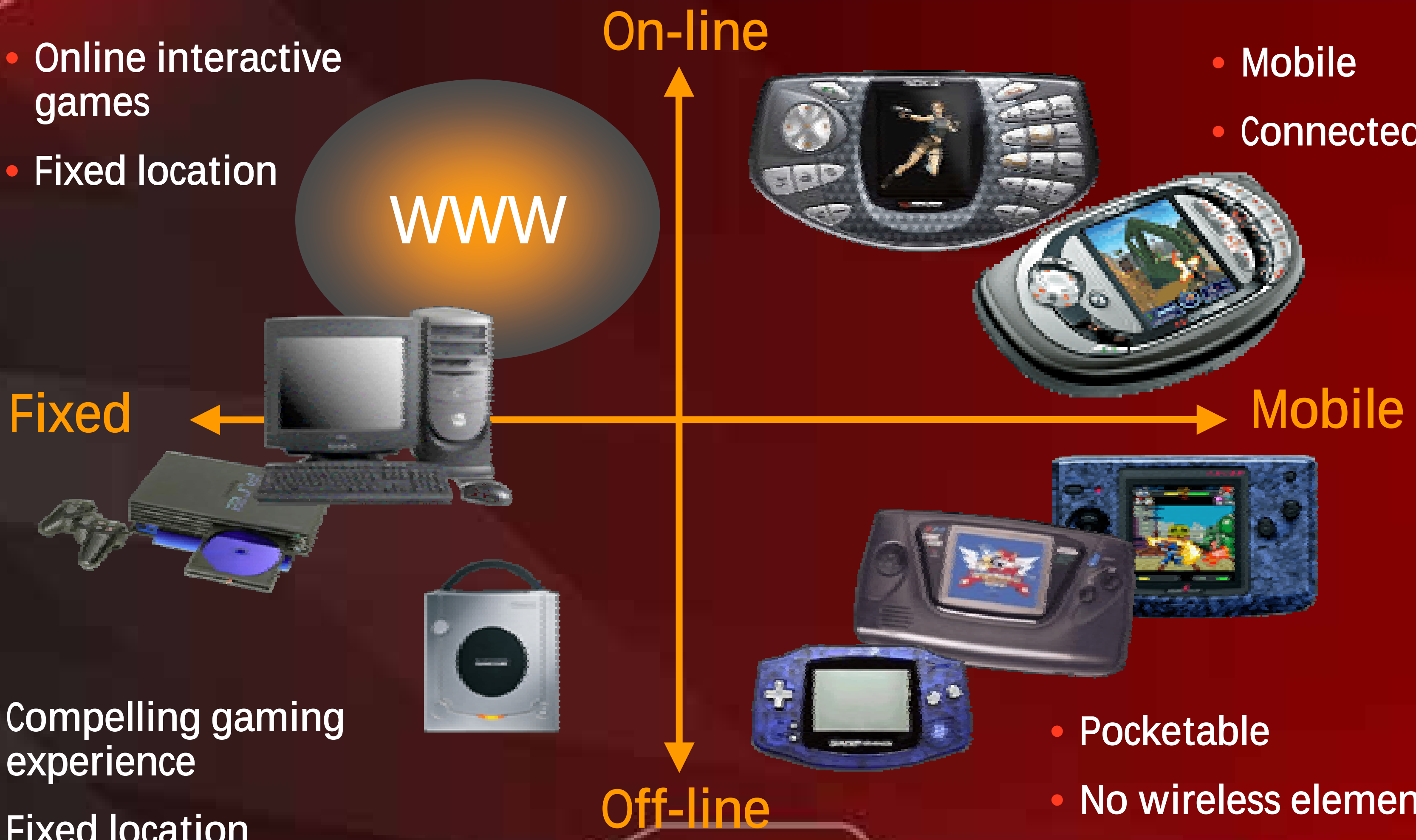
Gaming

- Gaming is the force that moves the PC industry
  - Both the CPUs / systems and graphics cards
  - Lately multimedia has been pretty effective too…
- High-end smartphones are already pretty capable
  - The further technology push comes mostly from entertainment
    - Gaming, multimedia, always-on-internet-etc-connectivity

Gaming categories

- Online interactive games
- Fixed location

- Mobile
- Connected



- Compelling gaming experience
- Fixed location

- Pocketable
- No wireless elements

Will the mobile side converge?

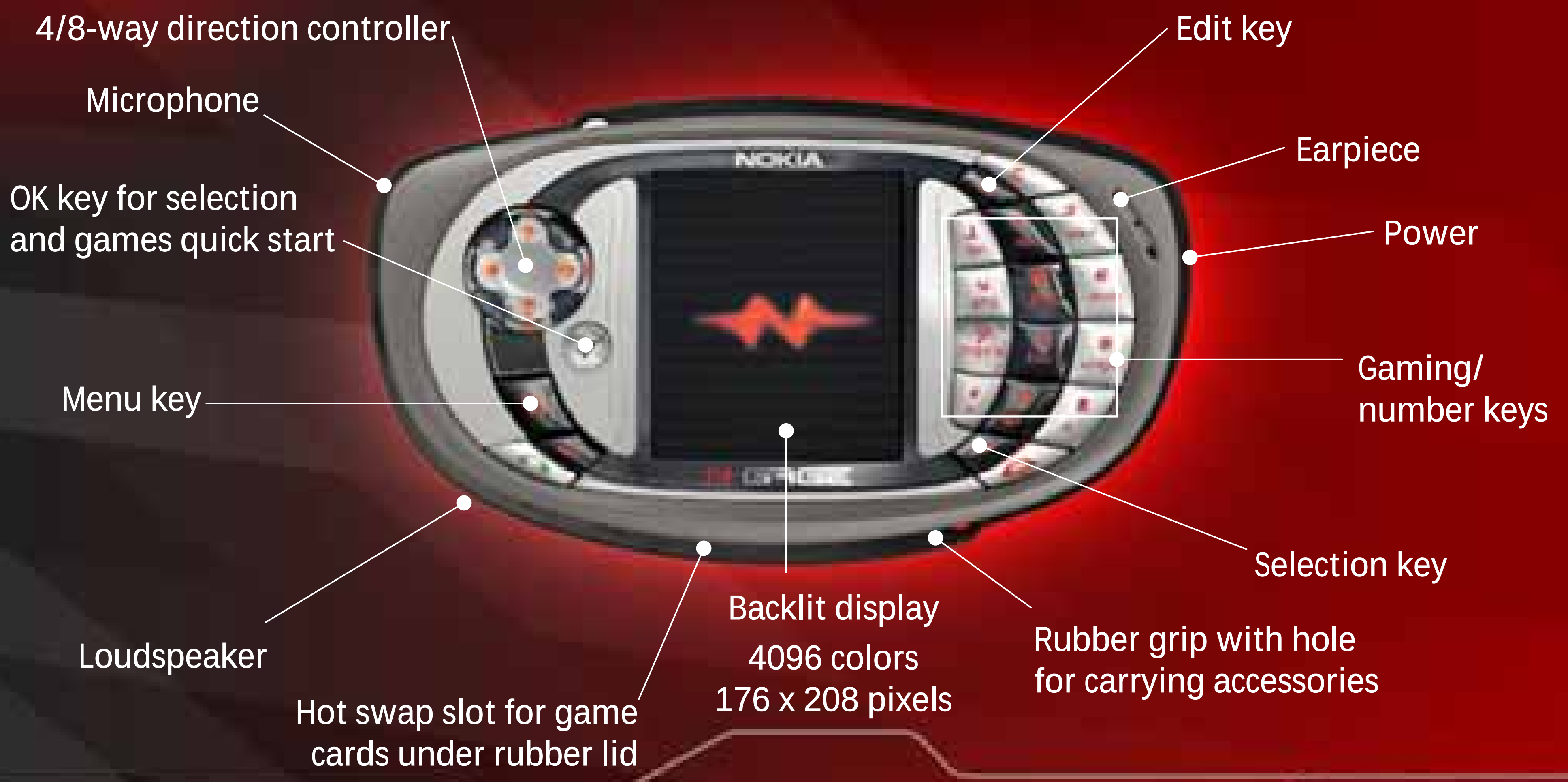
- Phone, also games
  - Volumes > 500 M / year
  - Heterogeneous SW / HW
  - Renewal period months
- Key: interoperability (Java, APIs)

- Games, maybe also a phone
  - Volumes in 10's of M / year (if lucky)
  - Fixed SW / HW environment
  - Renewal period years
- Key: performance (C, HW)

N-Gage QD – Both platforms in one

- Wireless multiplayer gaming
  - GPRS: online gaming, arenas
  - Bluetooth: local multiplayer (4-8)
- Full 2.5G phone functionality
- Optimized for mobile gaming
  - Games oriented landscape design
  - Large 4/8 way direction controller for navigation and gaming
  - TFT colour display with backlight
  - Truly pocketable size
- Open for 3RD party apps
  - Symbian / Java MIDP games

N-Gage QD – Vital Statistics


4/8-way direction controller
Microphone
OK key for selection and games quick start
Menu key
Loudspeaker
Hot swap slot for game cards under rubber lid
Backlit display
4096 colors
176 x 208 pixels
Rubber grip with hole for carrying accessories
Selection key
Gaming/ number keys
Power
Earpiece
Edit key
NOKIA

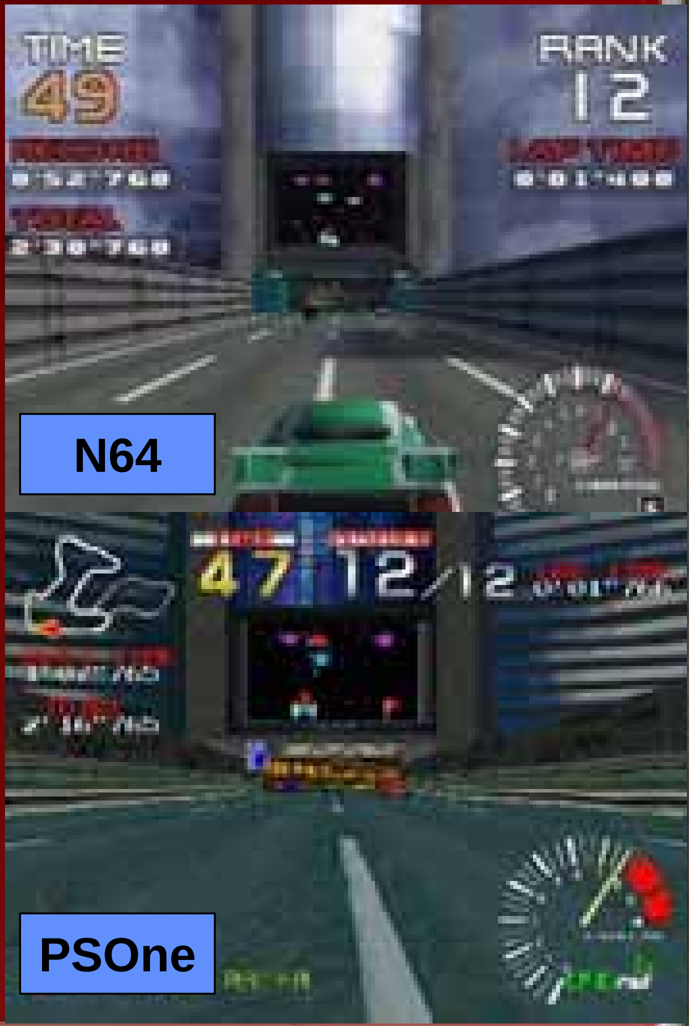
Social gaming

- 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

Back to technology: What’s important on Mobile 3D?

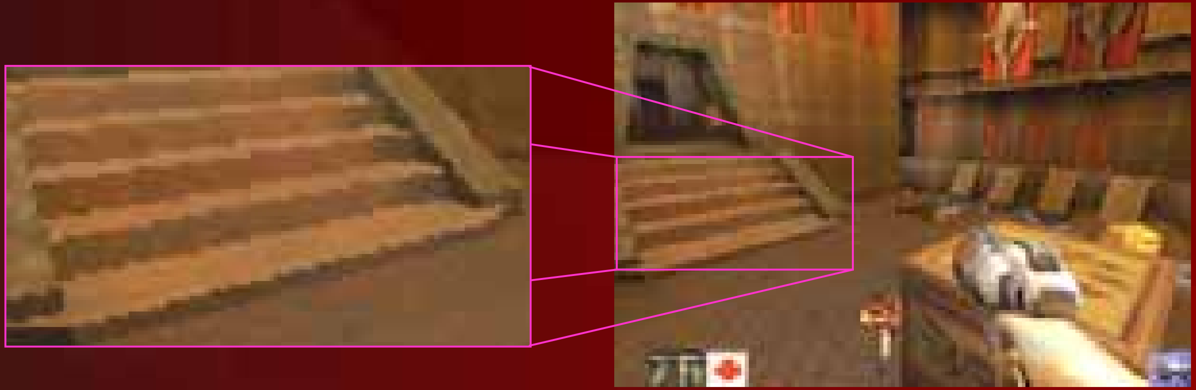
What's important in mobile graphics?

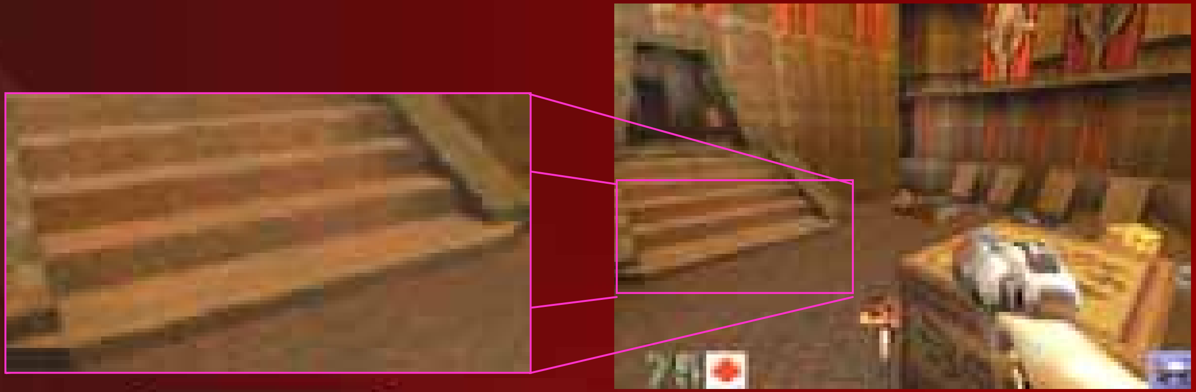
- The raw performance (TPS, PPS) not
  - Only designed as an indicator of performance
    - ...a bit like MIPS
  - Physical limitation on screen size = DPI limits
  - Little advantage in >3M tri/sec @ QVGA
  - Avg poly area in pixels into single figures
- Visual quality is important
  - N64 & PSOne had comparable TPS
  - N64's visual quality vastly superior
  - Important when running @ lower res.

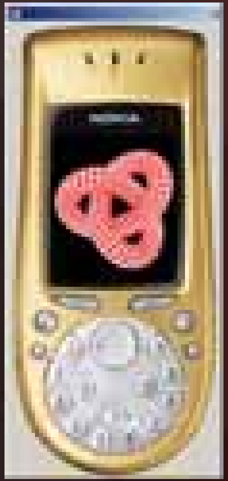


Anti-aliasing

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

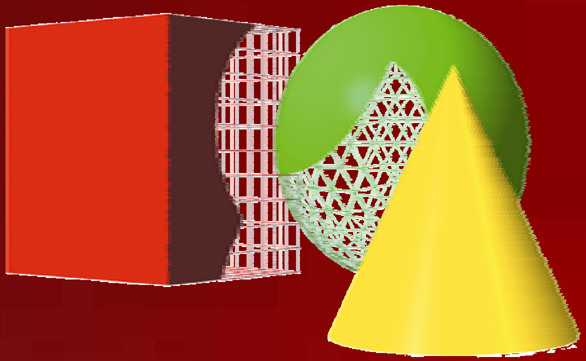






Output sensitivity

- So we want high-quality rendering
  - Need at least apparent detail
  - So most pixels need individual attention
  - Don’t process more than what you must
- Work only on visible pixels
  - Avoid read-modify-write
    - Those memory accesses use a lot of power!
    - Especially important with complicated pixel shaders
  - Deferred shading, texturing
  - Visibility culling



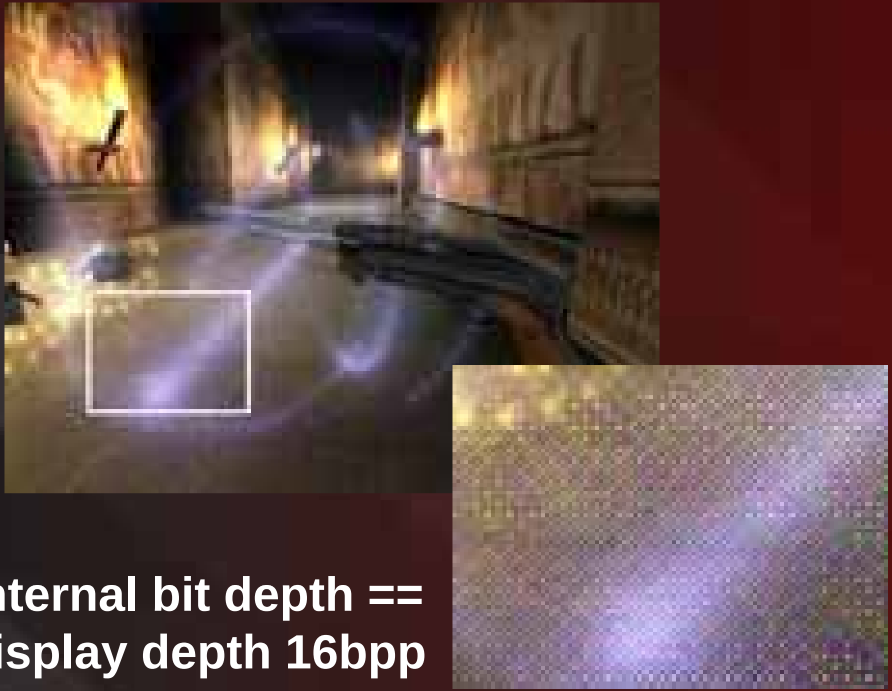
Deferred everything (1)

- Two-pass rendering
  - Render first just Z
    - Can be made much faster than generic rendering
  - Then render with only Z-test
- Advantages
  - Simple
  - Works on pretty much all architectures
- Problem
  - Extra pass
  - Transparent objects

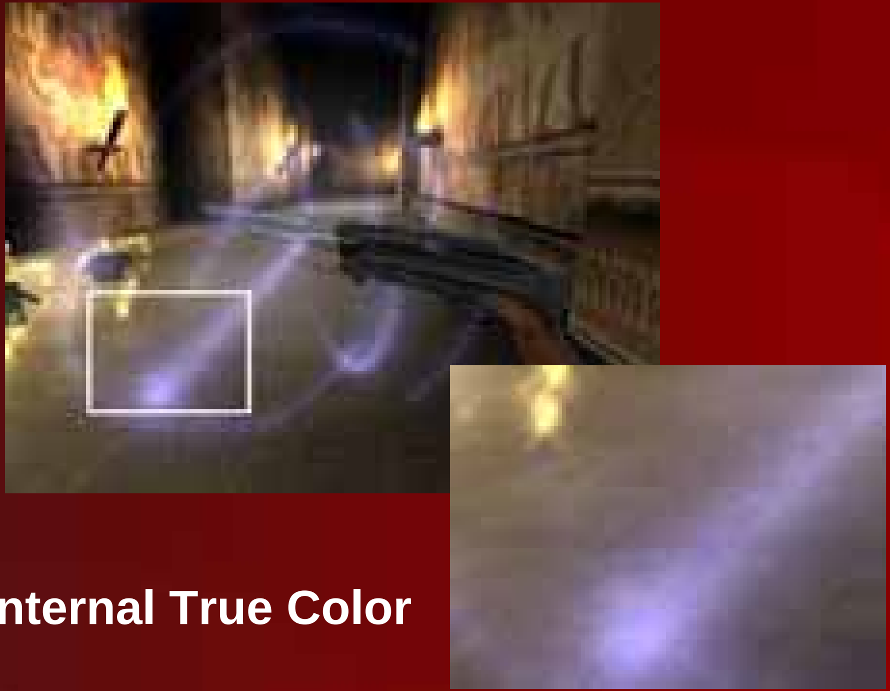
Deferred everything (2)

- Tiled rendering
  - Collect triangles to buffers
  - Process a tile at a time
    - Conceptually two-pass Z-first rendering
- Advantages
  - Can do much higher quality rendering with only a small memory overhead
    - FSAA, higher color depth
  - Read-modify-write on-chip, fast
    - Followed by a block write
- Problem
  - Needs big triangle buffers

Extra color bits



Internal bit depth == display depth 16bpp



Internal True Color

Deferred everything (3)

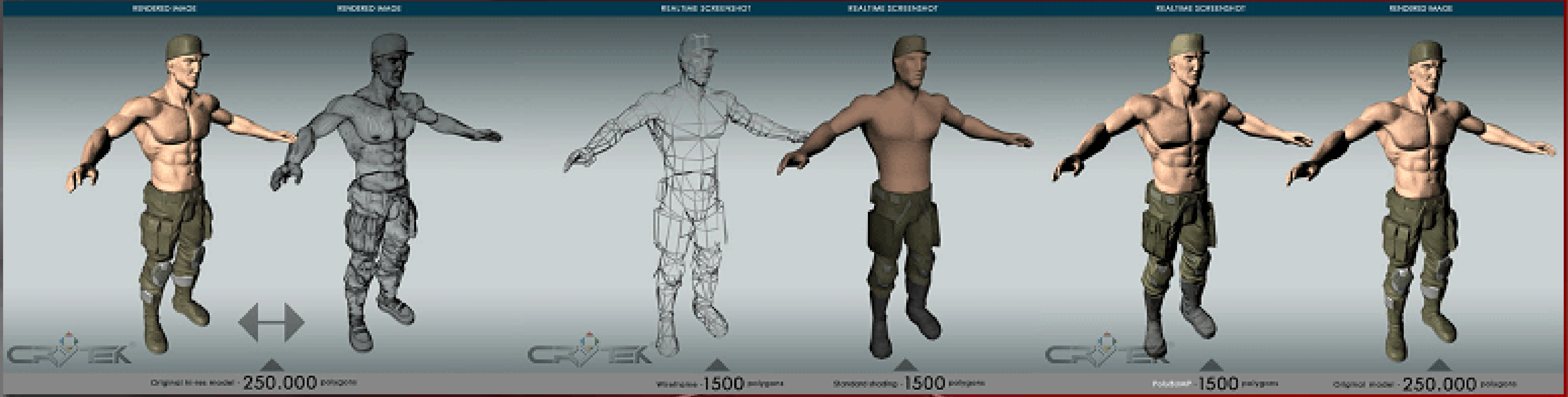
- Raytracing!
  - Don’t fake the global and dynamic lighting effects, do the real thing!
  - The ultimate and correct high quality output sensitive method
    - With good data structures you’ll hit the visible surfaces fast, ignore invisible ones

Deferred everything

- Combine different approaches
  - Hierarchical Z
  - Delay-streams
  - Intelligent caching (tiled under the hood)
  - ...

Avoid traffic between memory & GPU

- Send less data to the engine
  - Again: memory accesses drain the battery!
  - Texture compression
  - Better pixel processing gives same or better visual quality cheaper
  - Example: bump maps (supported already in OpenGL ES 1.1)

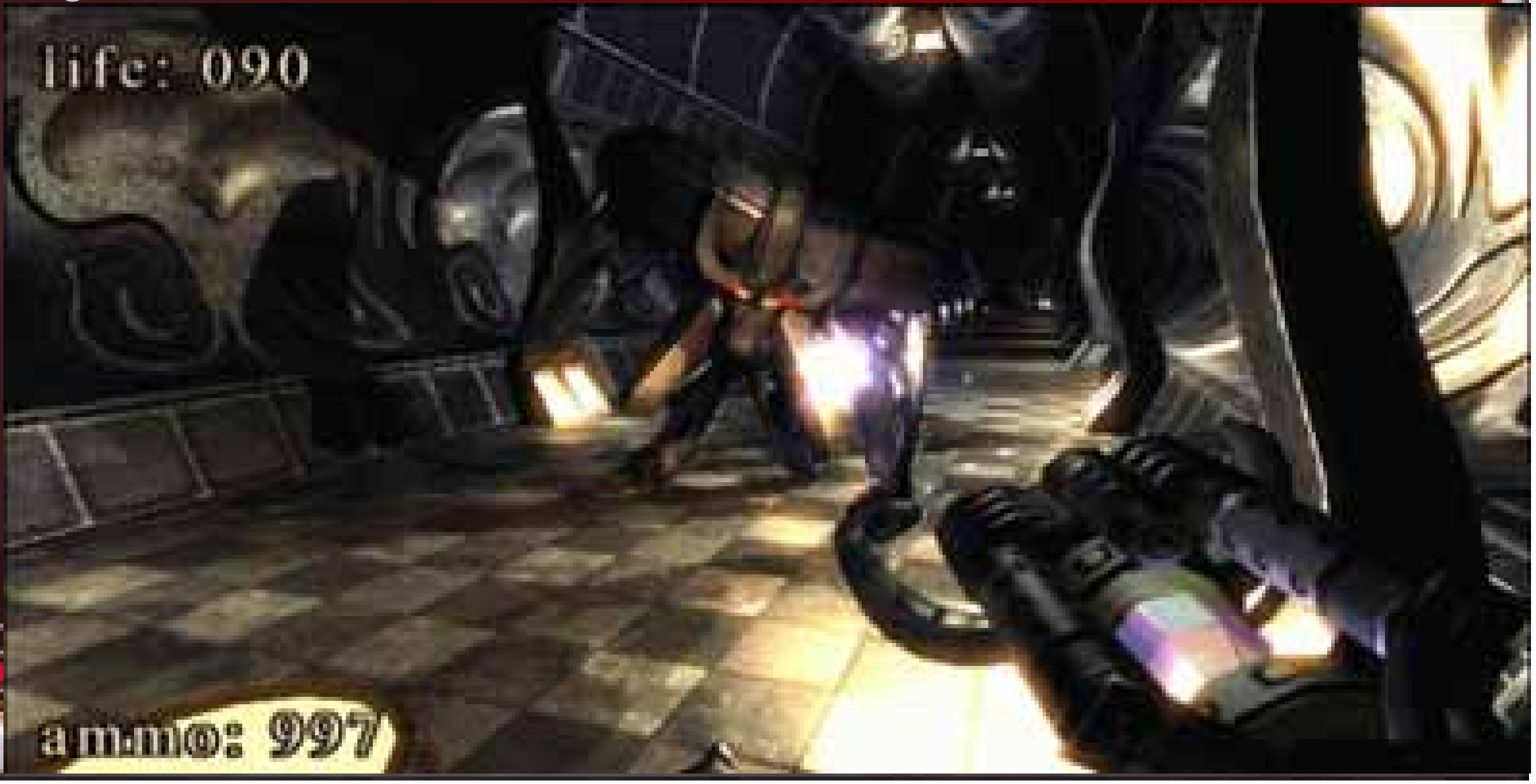

RENDERED IMAGE
RENDERED IMAGE
REALTIME SCREENSHOT
REALTIME SCREENSHOT
REALTIME SCREENSHOT
RENDERED IMAGE
CRYTEK
Original hi-res model · 250.000 polygons
Wireframe · 1500 polygons
Standard shading · 1500 polygons
Polybump · 1500 polygons
Original model · 250.000 polygons

Avoid traffic, period.

- Procedural everything
  - Ultimate compression!
    - Both for transmission *and* storage in handset *and* btw CPU and GPU
  - Send only little data, generate most of it in the shaders
  - Procedural *textures* in the pixel shader instead of texture lookups
    - Possible with current shaders
  - Procedural *geometry* generation in the vertex shader based on control points
    - Only becoming possible
  - 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 kinds of waste

- Avoid having unused HW blocks
  - 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

Mobiles are very cost-conscious

- People expect to pay ~1000 USD for a PC
  - Many people seem to think phones are free
    - Or maybe 1c with a service plan…
  - Even those that pay don’t want to pay as much as for PC
    - It’s so small, why should it be expensive?
- Implications
  - Get the silicon area down
    - It doesn’t cost just power, but also $$$
  - Integrated solution is typically much cheaper than a discrete chip
    - Especially on high-volume products, and mobiles are high-volume
  - Can’t afford wide busses to external DRAM
  - Embedded DRAM is really nice for reducing traffic, but is expensive

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
  - Certainly more active players on mobile than on desktop

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

Ideas and material from

Ed Plowman, ARM
Affie Munshi et al., ATI
Petri Nordlund, Bitboys
Nicolas Thibieros et al., Imagination
Miroslaw Bober et al., Mitsubishi
Ed Hutchins, NVidia
Philipp Slusallek, U. Saarland
Sean Ellis, Superscape
Kazuki Hirakawa et al., Toshiba
Neil Trevett, 3D Labs

And many others:

Nokia

- Tomi Aarnio
- Panu Brodkin
- Ilkka Harjunpää
- Tapio Hill
- Aki Järvilehto
- Jarkko Kemppainen
- Tapani Leppanen
- Jouka Mattila
- Kimmo Roimela
- Damian Stathonikos
- Jani Vaarala

Thank You!