New Standards for Mobile Graphics

Kari Pulli
Research Fellow
Nokia Research Center
MIT CSAIL

NOKIA Connecting People

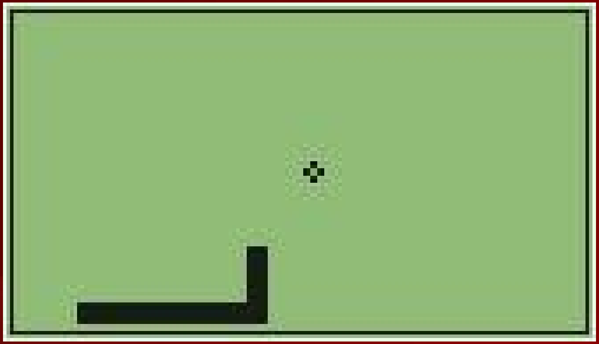
Outline

- Brief history of mobile 3D
- Key enablers and challenges
- Mobile graphics standards
  - OpenGL ES
  - M3G
  - Collada
  - OpenVG
  - JSR 226 (2D Scalable Vector Graphics for J2ME)
- Mobile graphics HW

NOKIA
Connecting People

State-of-the-art in 2001: GSM

- Snake: 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"
- We begun to work on mobile 3D engine at Nokia in 2001
  - OpenGL subset
    - low-level API
    - perspective camera / textures, lighting, blending
  - shipped in 2002



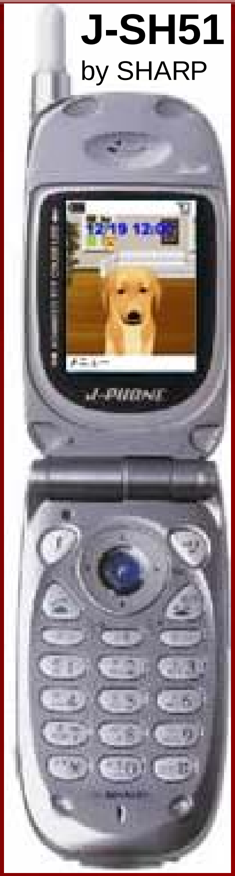
NOKIA Connecting People

Japan 2001-02

- Already color screens
  - enables nice content
- High-level API
  - skinning
  - flat shading, orthographic view, no or limited blending
- SW engines for the next few years

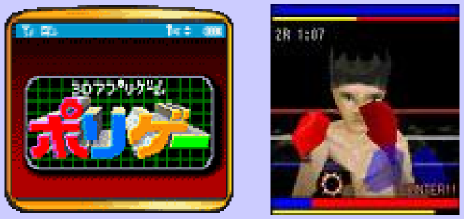
GENKI 3D Characters
(C) 2001 GENKI

ULALA
(c)SEGA/UGA.2001



J-SH51
by SHARP

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



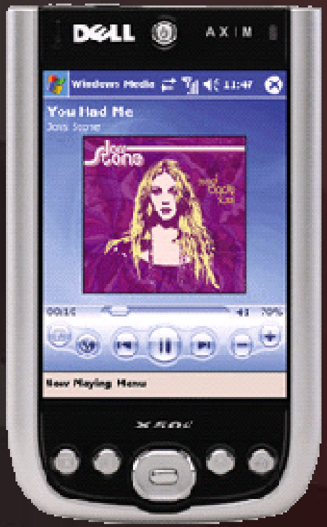
I-3D PolyGame Boxing

@ Hi Vanguard· REZO, BNW

NOKIA Connecting People

2005 and beyond: With HW

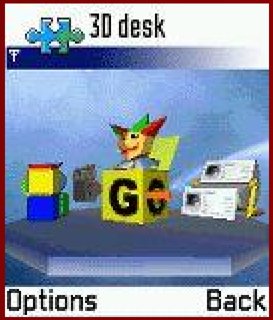
- PSP, Gizmondo, Zodiac, Axim, ...
- Gaming phones with 3D gfx HW

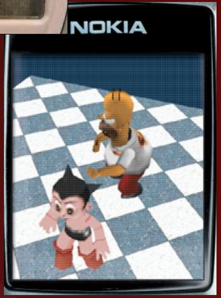


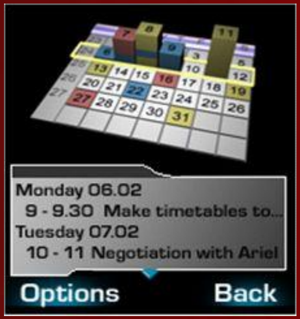
NOKIA
Connecting People

Mobile graphics use cases

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






Monday 06.02
9 - 9.30 Make timetables to...
Tuesday 07.02
10 - 11 Negotiation with Ariel

Key enablers and challenges

NOKIA
Connecting People

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?

NOKIA Connecting People

Changed? Computation!

- Moore‛s law in action
  - 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

NOKIA Connecting People

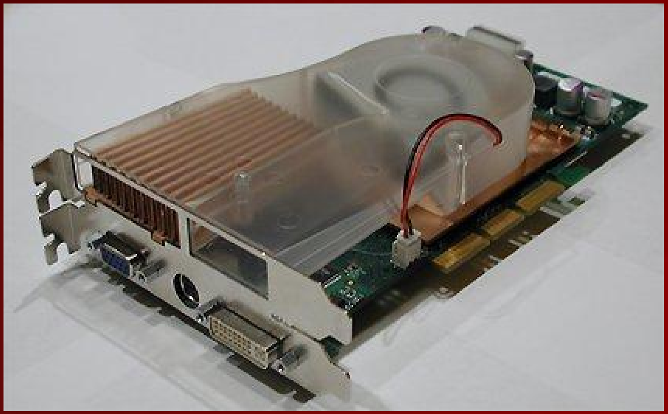
Challenge? Power!

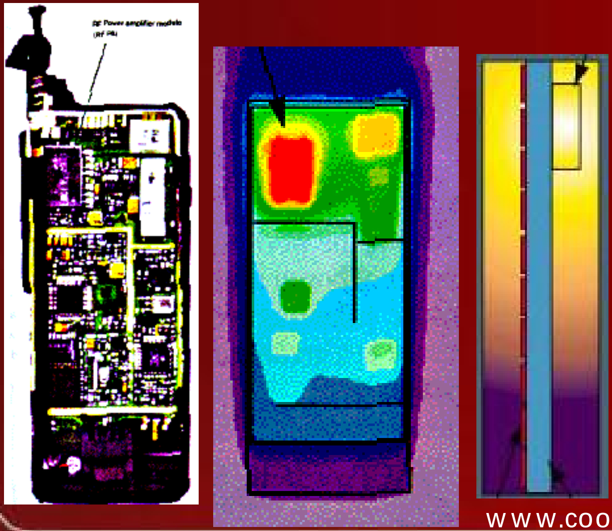
- Power is the ultimate bottleneck
  - Usually not plugged to wall while using, just batteries
- Battery improvement doesn t' follow Moore s' law
  - Only 5-10% per year
- Gene s' law
  - "power consumption of integrated circuits decreases exponentially" over time => batteries will last longer
    - 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

NOKIA Connecting People

Challenge? Thermal mgt!

- But ridiculously good batteries still won ť 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…
  - Conduct the heat through the walls, and finally release to the ambient





www.coolingzone.com

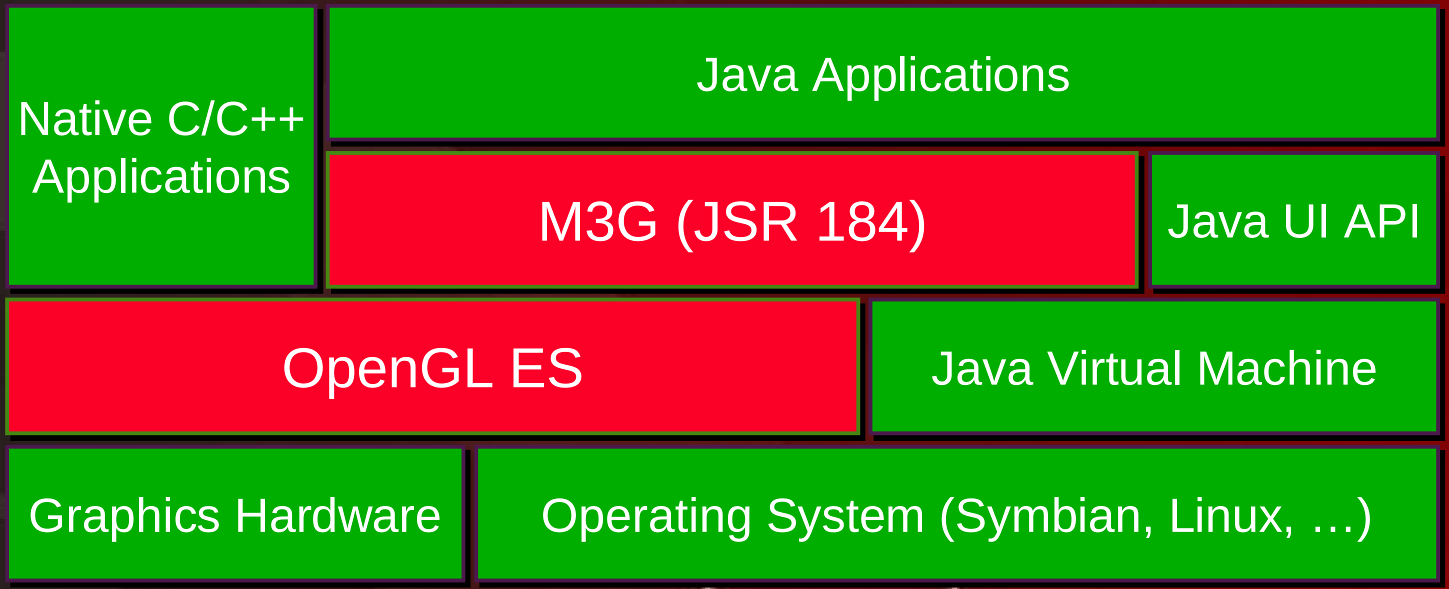
NOKIA Connecting People

Standard APIs for mobile graphics

NOKIA
Connecting People

Layering for resource reuse

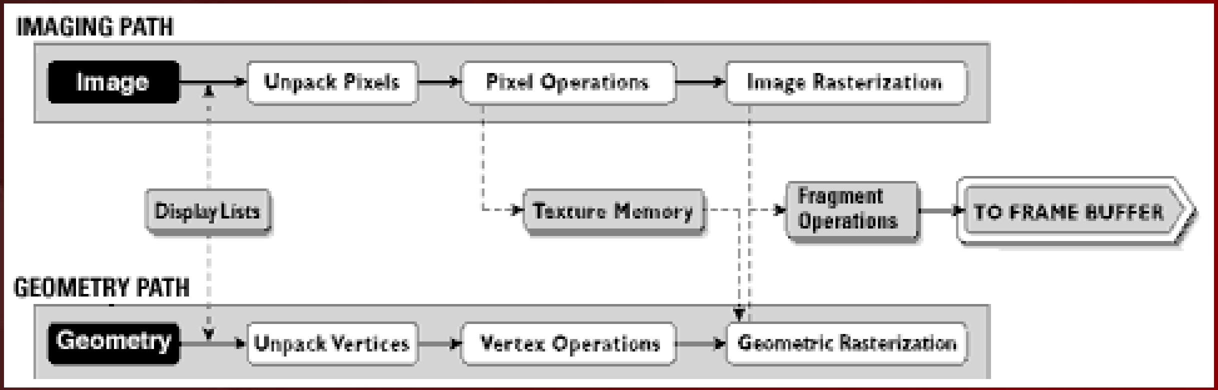
- OpenGL ES and M3G
  - Designed concurrently (and partly by the same people)
  - Influenced each other
  - Layered implementation model (immediate benefit of same HW)



NOKIA Connecting People

OpenGL

- The most widely adopted graphics standard
  - most OS ṡ, 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

NOKIA
Connecting People

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 1.3
  - that can still do almost all OpenGL can
  - eliminate un-needed functionality
    - redundant / expensive / unused

OpenGL ES 1.0 in a nutshell

- Retain functionality that apps can t' emulate
  - Keep full fragment (pixel) processing: blending, texture mapping, ...
- 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

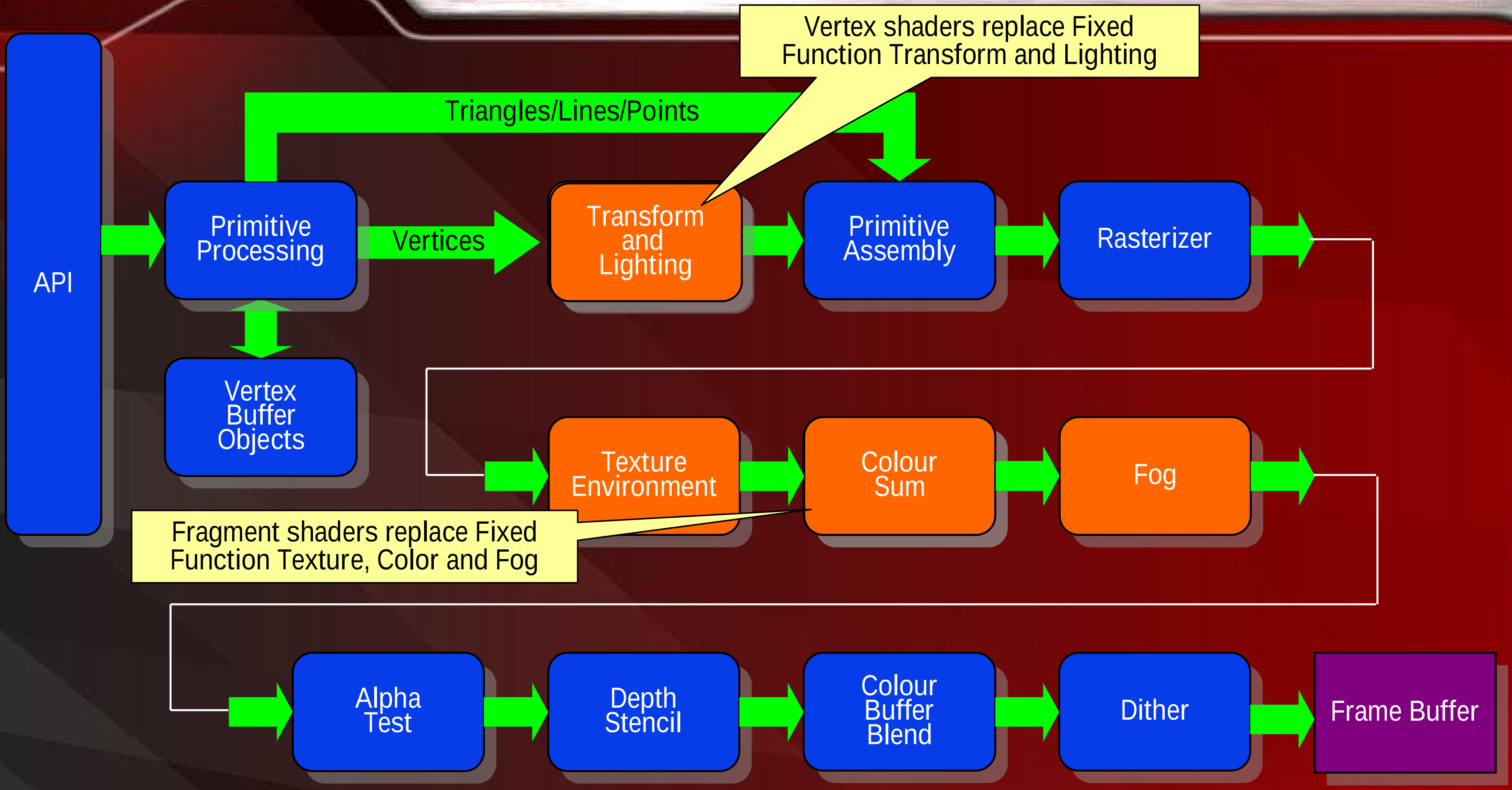
OpenGL ES 1.1 in 2004

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

OpenGL ES 2.0 in 2005

- Was released at SIGGRAPH ’05
- Features
  - Vertex and fragment shaders
    - GL Shading language (similar, but different from the desktop)
  - 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!

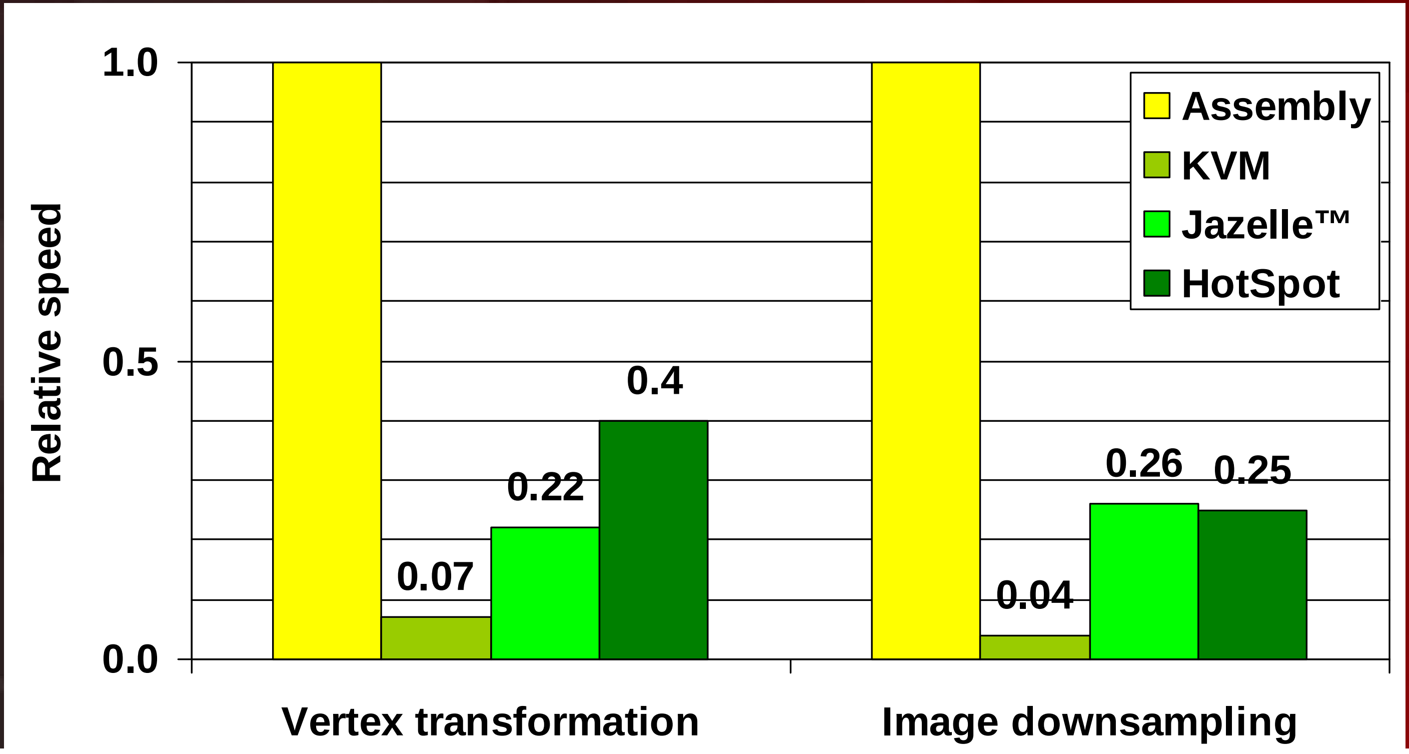
2.0 Programmable Pipeline



NOKIA Connecting People

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, …)
  - 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!

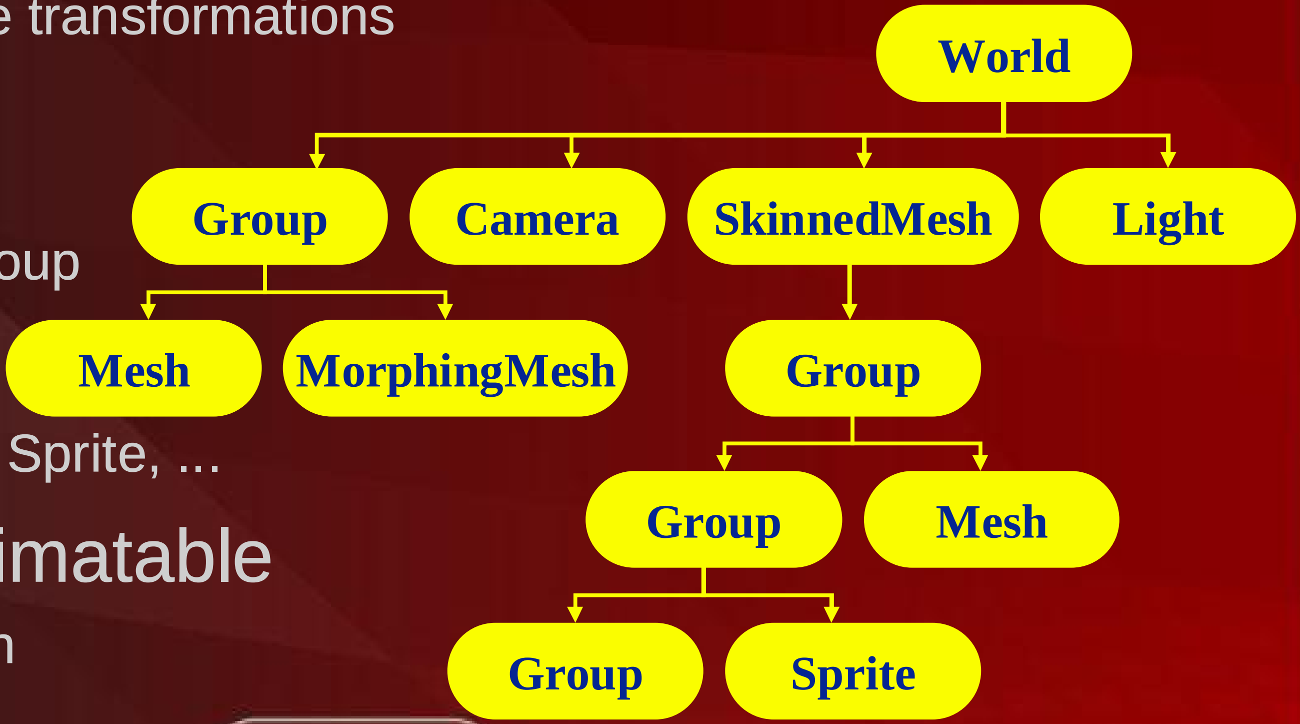
- Java3D seemed a good starting point
  - But "Java3D ES" didn't work out
  - Java3D was designed for large-resource systems
    - Java3D distribution is ~40MB (~300x too big for us)
  - Didn't really fit together with MIDP
    - a large redesign necessary
- M3G (JSR 184), a new API
  - Nodes and scene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

M3G
=
Mobile 3D Graphics for Java
!=
Java3D

M3G (JSR-184)

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, …
- Every property animatable
  - Keyframe interpolation

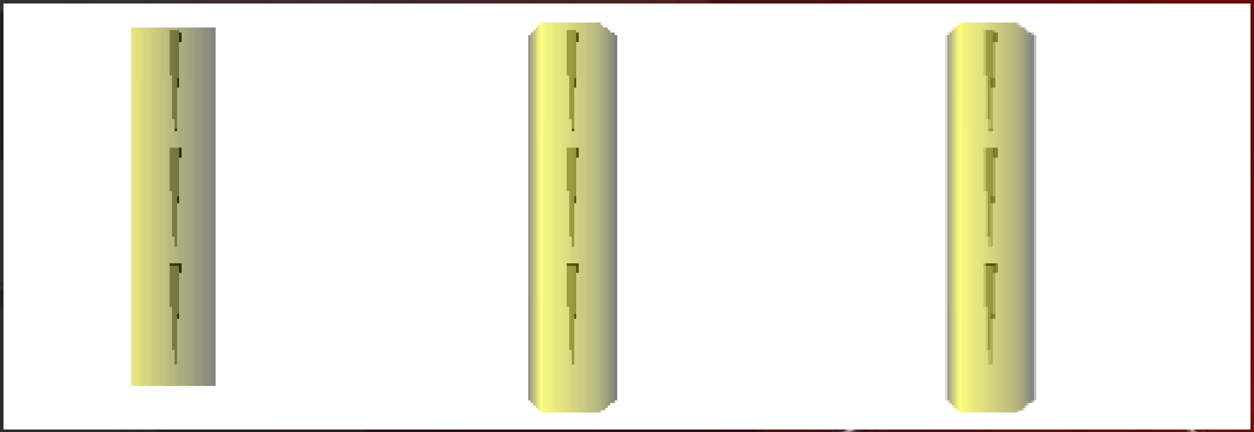


M3G (JSR-184)

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



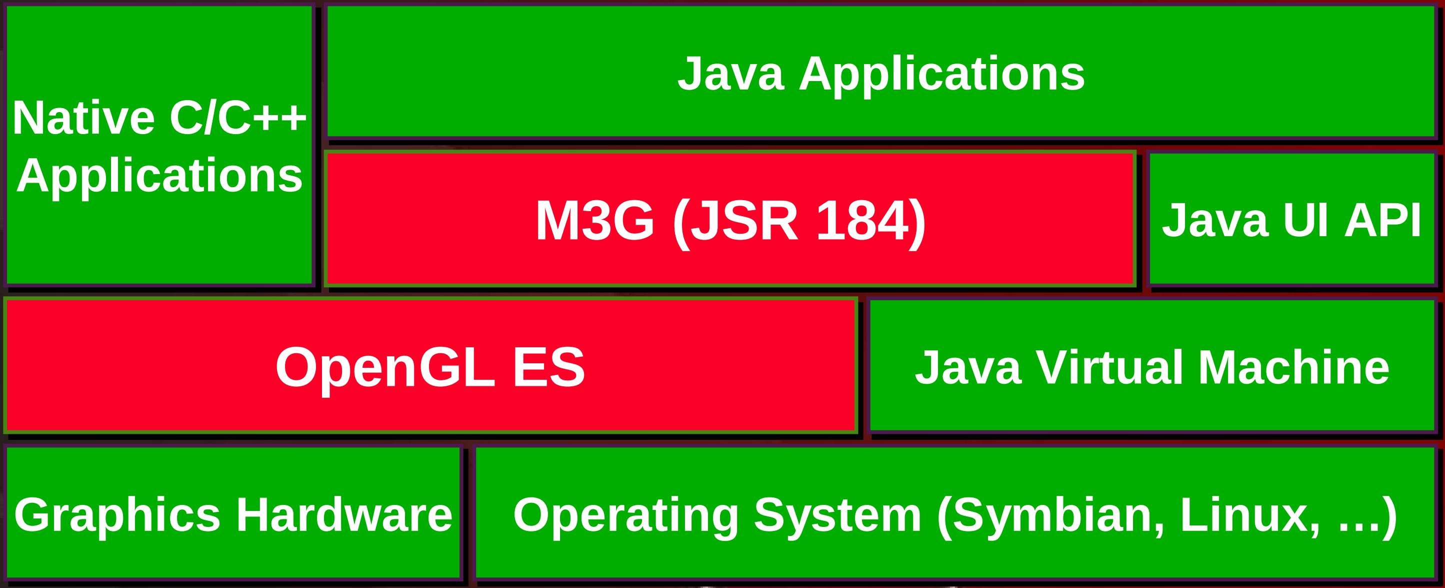
M3G (JSR-184)

M3G 2.0?

- Might start work in 2006
- Possibly
  - still high level API
  - backwards compatible
  - requires OpenGL ES 2.0
  - ES 2.0 shading language

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)

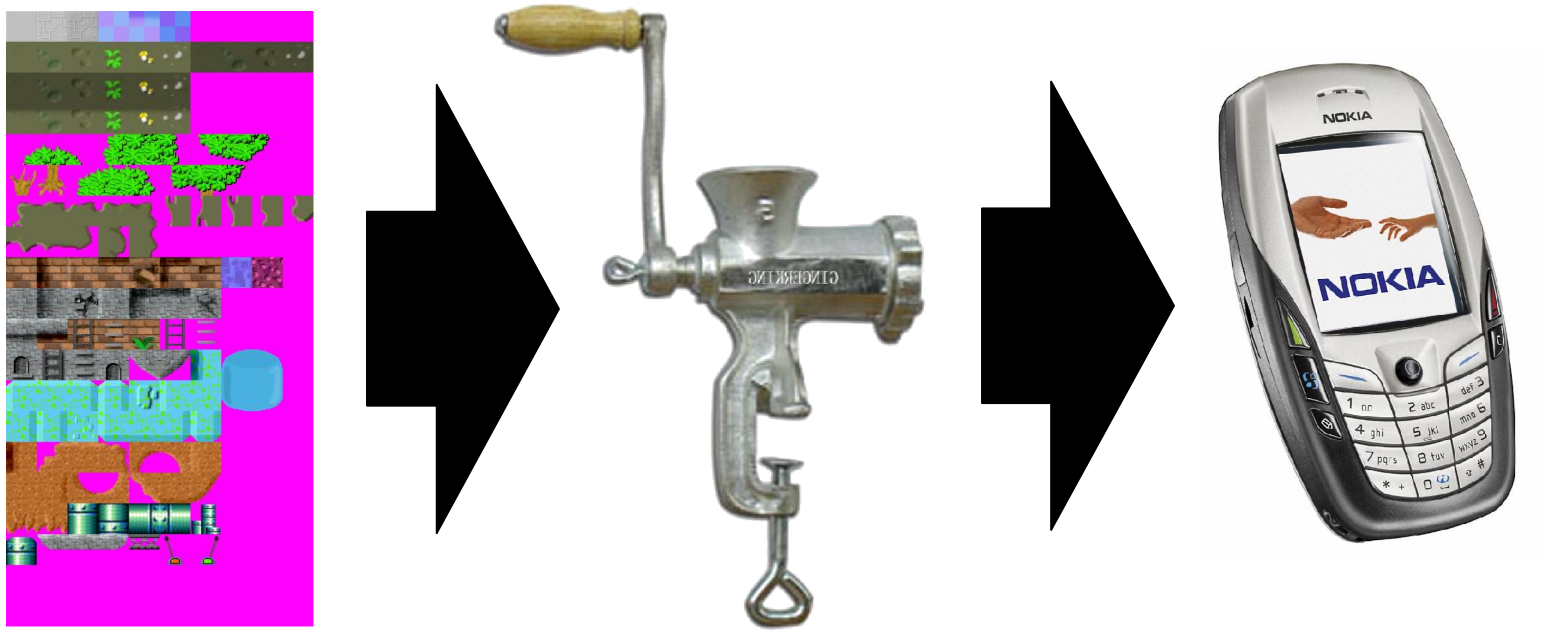


3D Pipeline Workflow

- 3D Games require meshes, textures, animations, material definitions, etc.
- Assets flow from artists, through a tool pipeline to optimize them
- Modified assets are then formatted for the target platform

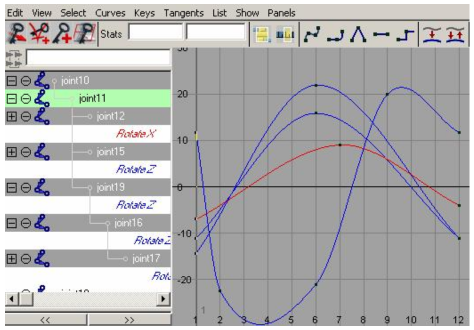
COLLADA

2D Asset Pipeline

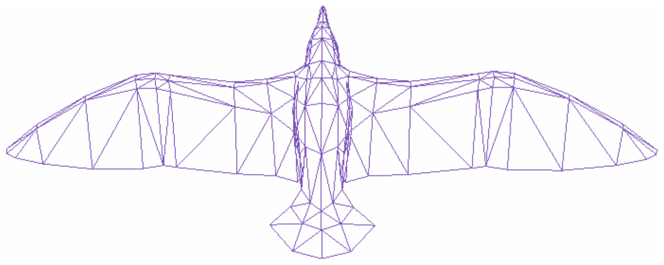


Not too bad…

3D Assets



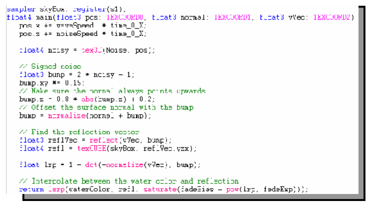
Animation



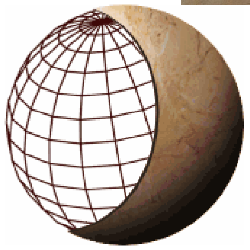
Meshes

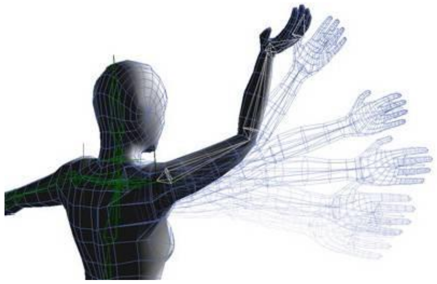
Textures

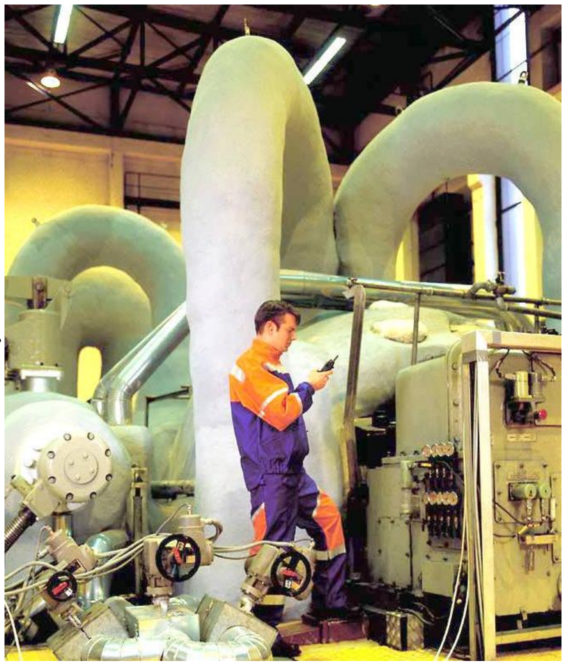
3D Asset Pipeline

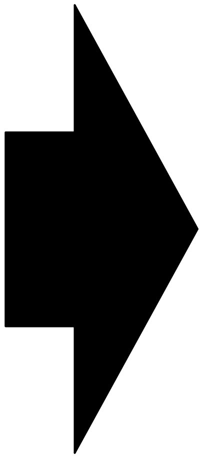


```
sampler skyBox: register(s1);
float4 main(float3 pos: TEXCOORD0, float3 normal: TEXCOORD1, float3 vVec: TEXCOORD2)
   pos.x += waveSpeed  * time_0_X;
   pos.z += noiseSpeed * time_0_X;

   float4 noisy = tex3D(Noise, pos);

   // Signed noise
   float3 bump = 2 * noisy - 1;
   bump.xy *= 0.15;
   // Make sure the normal always points upwards
   bump.z = 0.8 * abs(bump.z) + 0.2;
   // Offset the surface normal with the bump
   bump = normalize(normal + bump);

   // Find the reflection vector
   float3 reflVec = reflect(vVec, bump);
   float4 refl = texCUBE(skyBox, reflVec.yzx);

   float lrp = 1 - dot(-normalize(vVec), bump);

   // Interpolate between the water color and reflection
   return lerp(waterColor, refl, saturate(fadeBias - pow(lrp, fadeExp)));
```

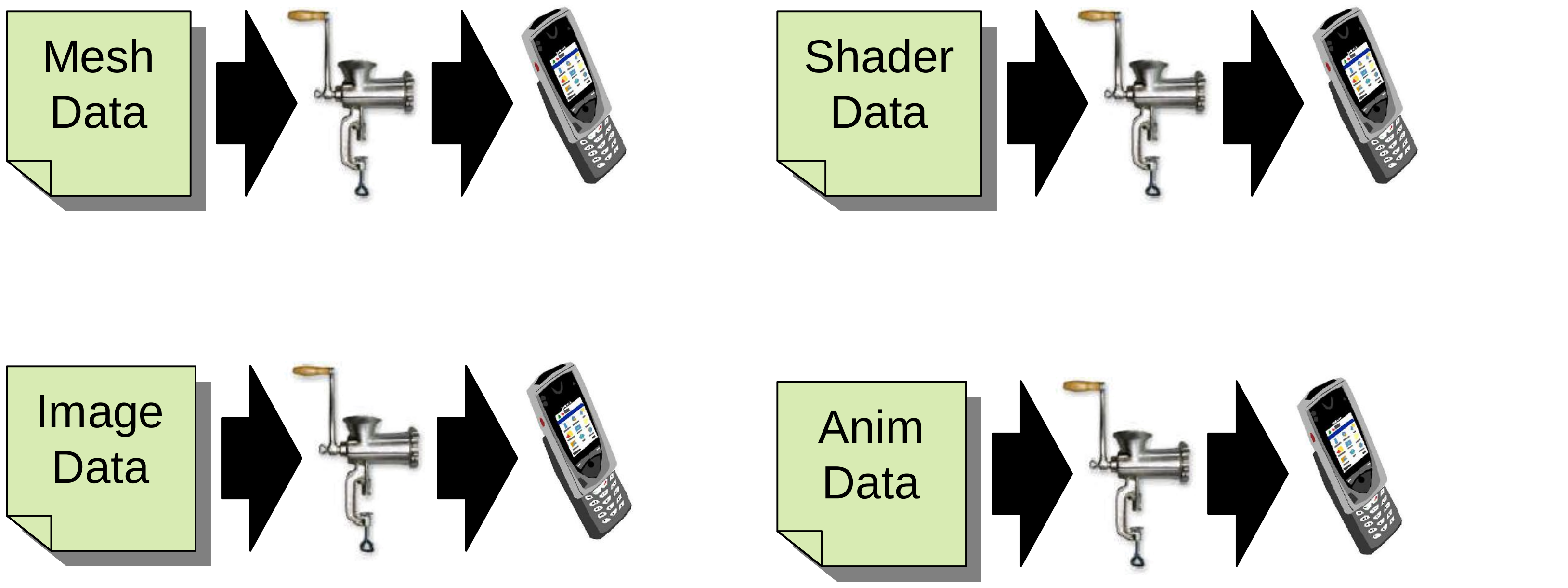






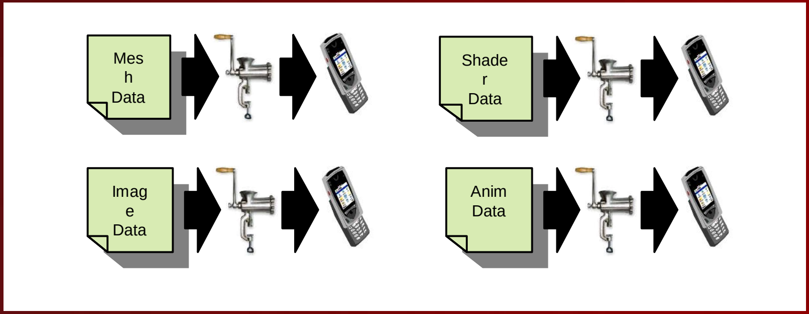


Format overload!



Format overload...

- Too many types of data to read / write !
  - Several formats to learn / implement / debug
- Tools can become too specialized
  - Can lead to fragmented workflow
- What about plugins?
  - Difficult to write / debug
  - Difficult to port to new DCC vendor

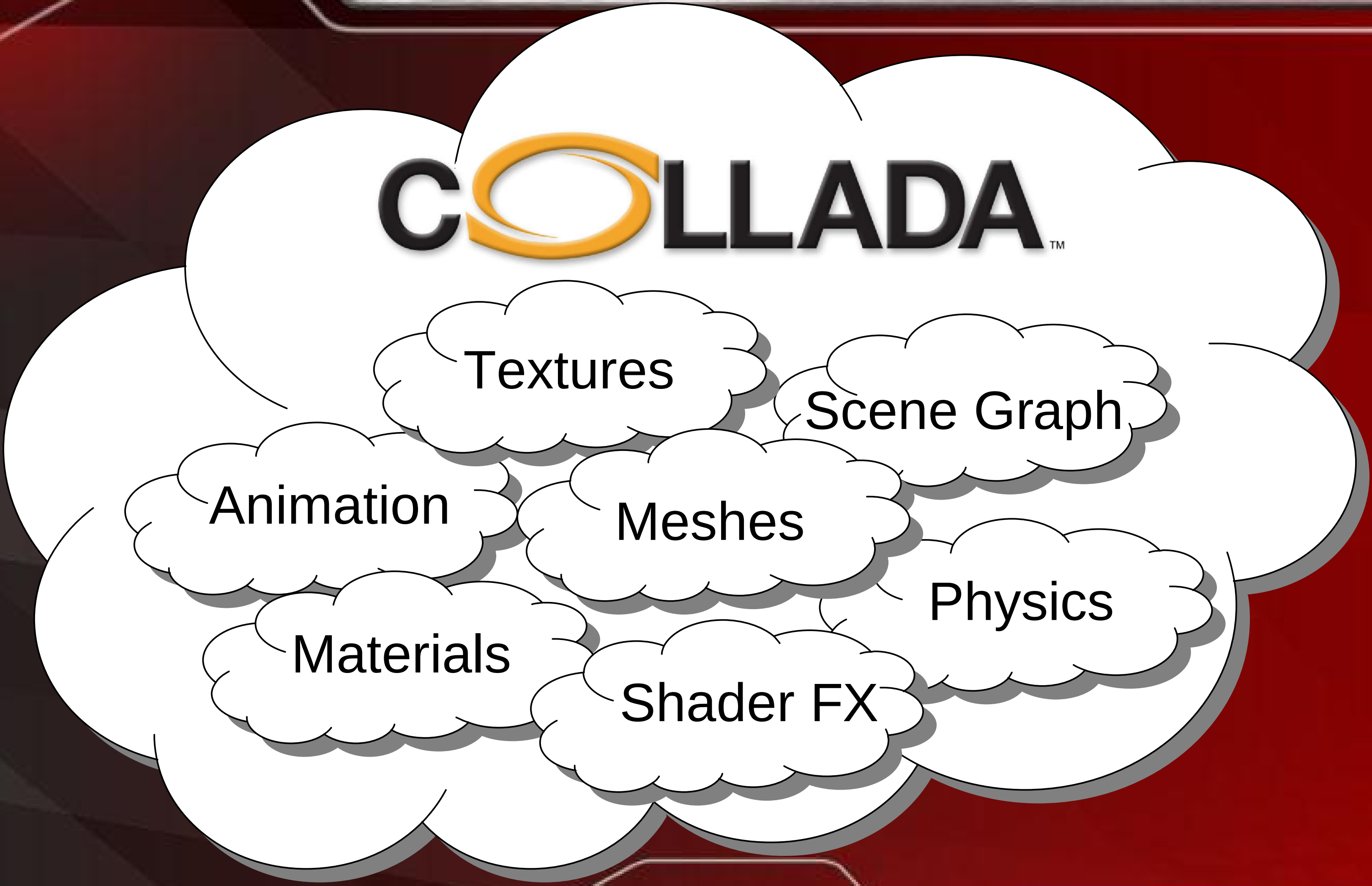


What is COLLADA?

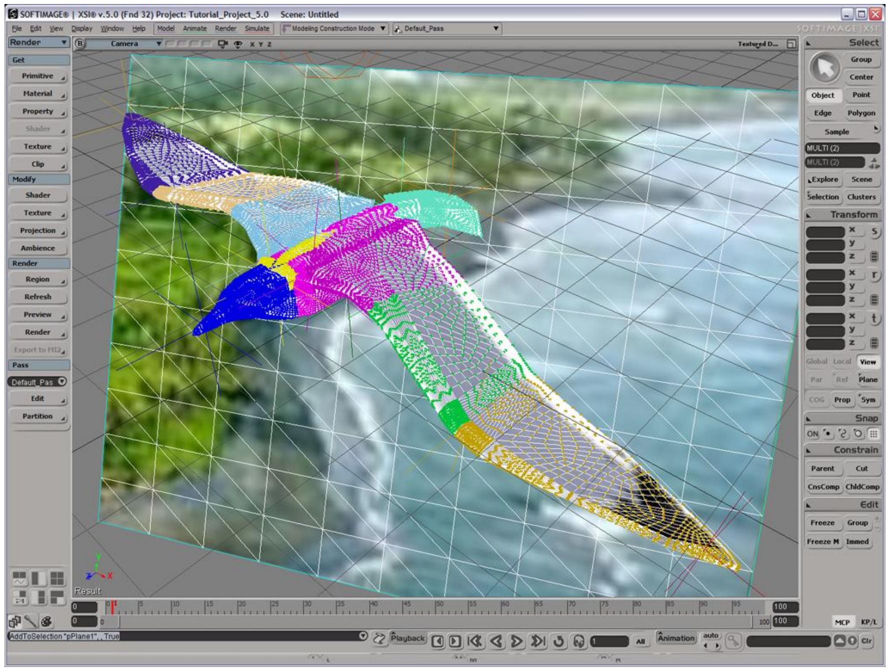
- COLLADA is designed to be an interchange format between applications

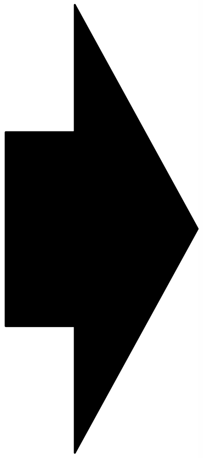
  - It is the glue that binds together DCC and 3D processing tools into a production pipeline
  - It allows applications to talk with each other using the same language
  - It allows developers and tool makers to build on each other's strengths without reinventing the wheel

What is Collada...



Use desktop tools for mobile

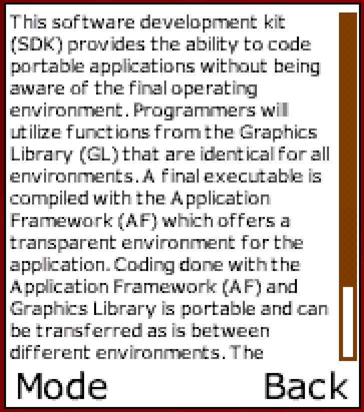

SOFTIMAGE® | XSI® v.5.0 (Fnd 32) Project: Tutorial_Project_5.0 Scene: Untitled

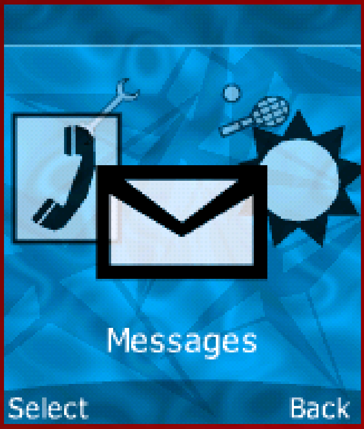


COLLADA™

What about 2D vector gfx?

- Use cases
  - Mapping / GPS, E-books, other text packages
  - Many games are essentially 2D
  - Advanced user interfaces and screen savers
- Scalable graphics = Easier to port content

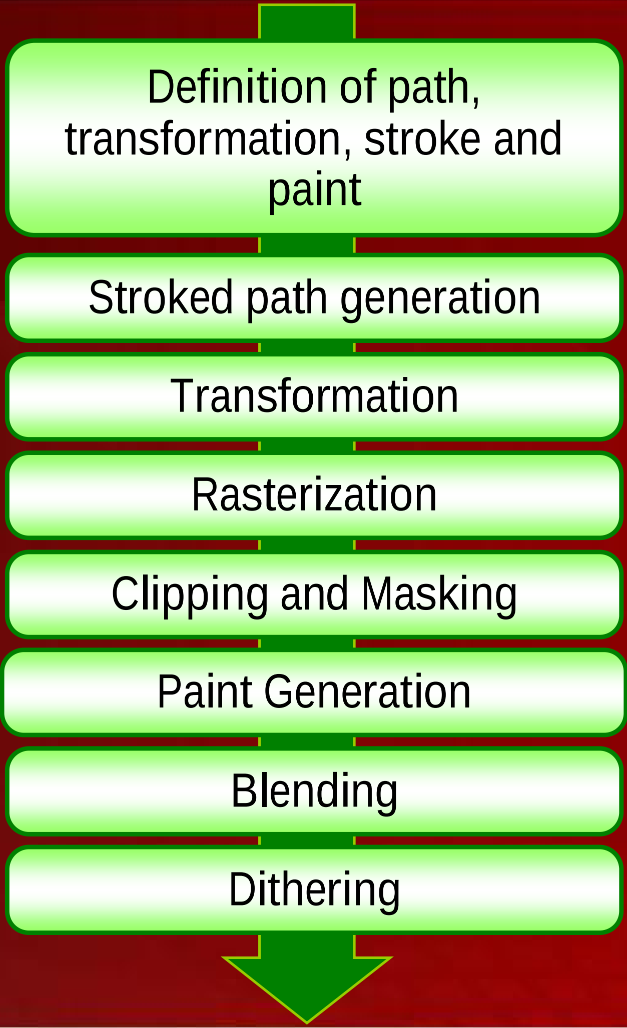

This software development kit (SDK) provides the ability to code portable applications without being aware of the final operating environment. Programmers will utilize functions from the Graphics Library (GL) that are identical for all environments. A final executable is compiled with the Application Framework (AF) which offers a transparent environment for the application. Coding done with the Application Framework (AF) and Graphics Library is portable and can be transferred as is between different environments. The



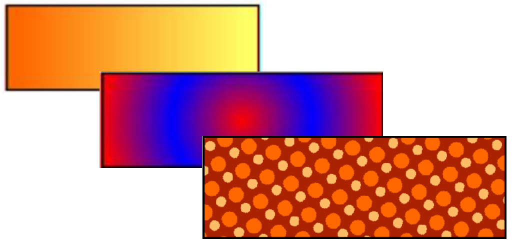
OpenVG

OpenVG

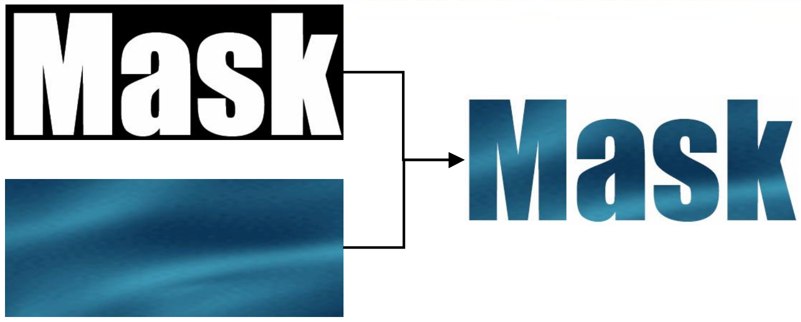
- Open standard for Vector Graphics HW acceleration
  - low-level API, similar to OpenGL ES
  - off-loads 2D graphics from CPU to GPU: power savings
  - works as a HW abstraction layer
  - can support scalable 2D formats on top of it
    - SVG, Flash, PDF, Powerpoint, ...
- Specification completed in late 2005



OpenVG: Features



Paints



Mask

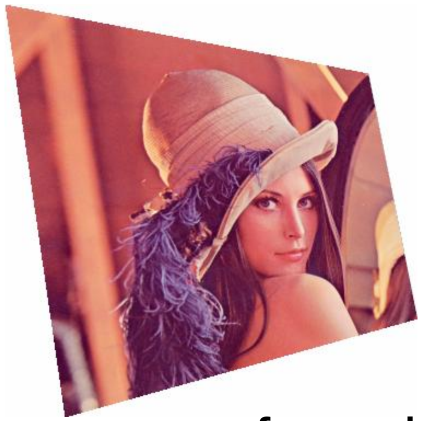
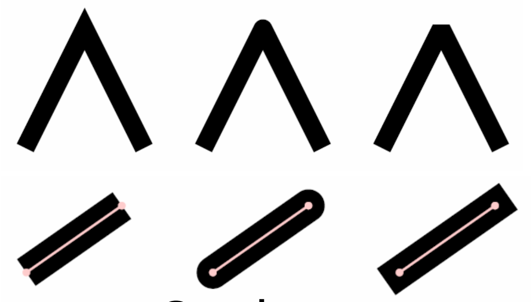
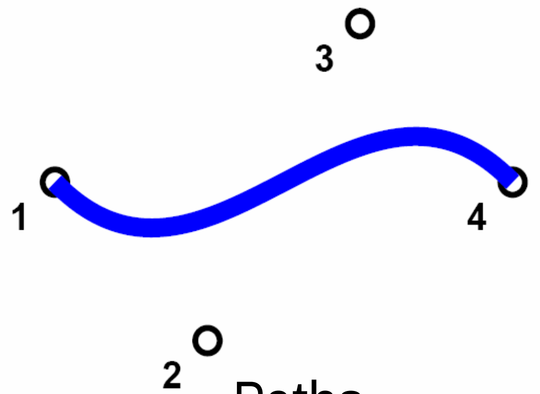


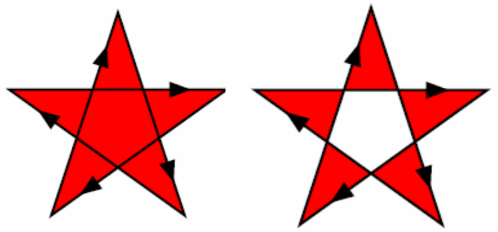
Image transformation



Stroke



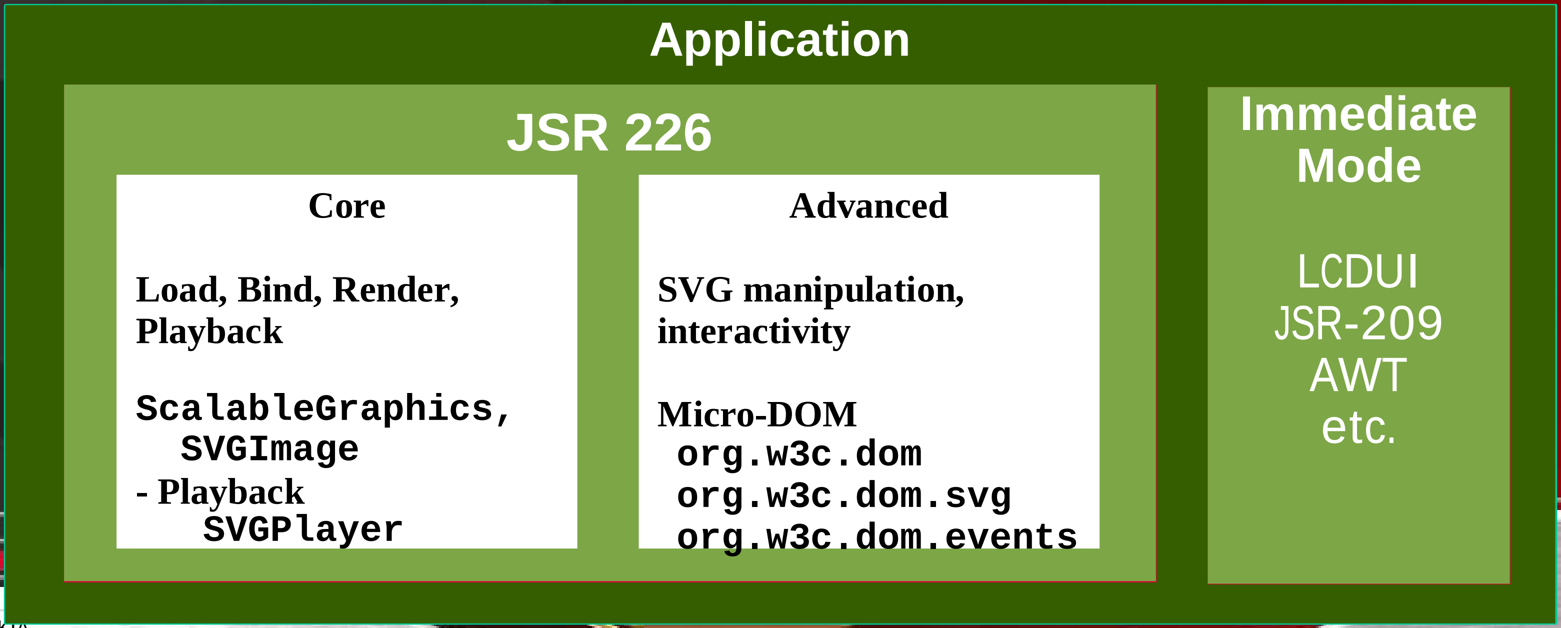
Paths



Fill rule

2D vector graphics for Java

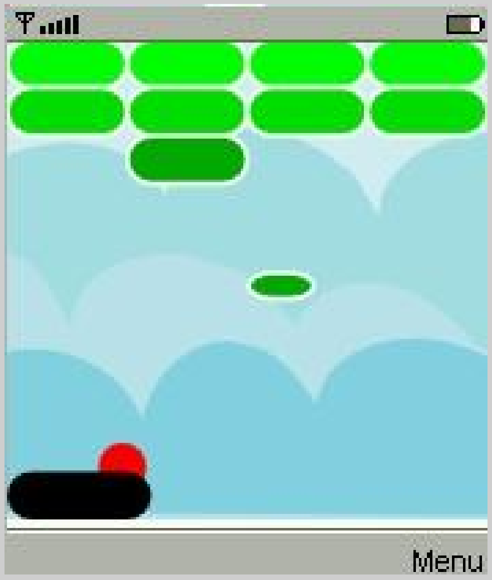
- JSR 226
  - Compatible with SVG Tiny 1.1
  - Support XML / SVG Micro-DOM
  - Two rendering modes
    - One-shot mode (Application control)
    - Playback mode (SVG Player)



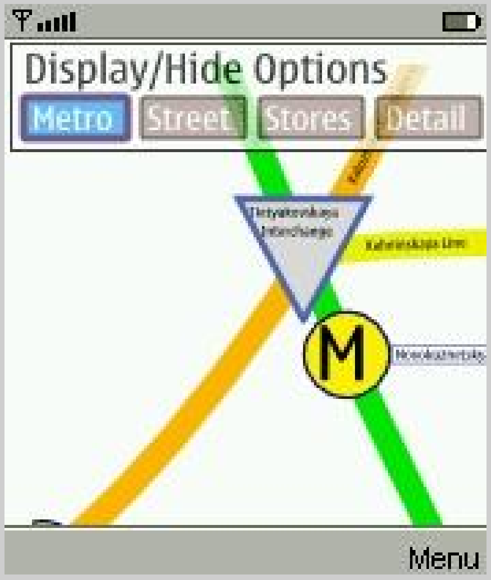
NOKIA Connecting People

JSR 226 demos

- Running on JSR 226 reference implementation

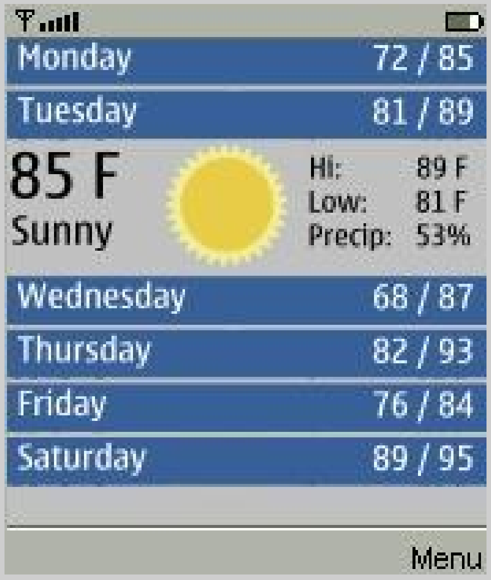


Game, with skins



Scalable maps, variable detail

Cartoon



Monday	72 / 85
Tuesday	81 / 89

85 F
Sunny

Hi:	89 F
Low:	81 F
Precip:	53%

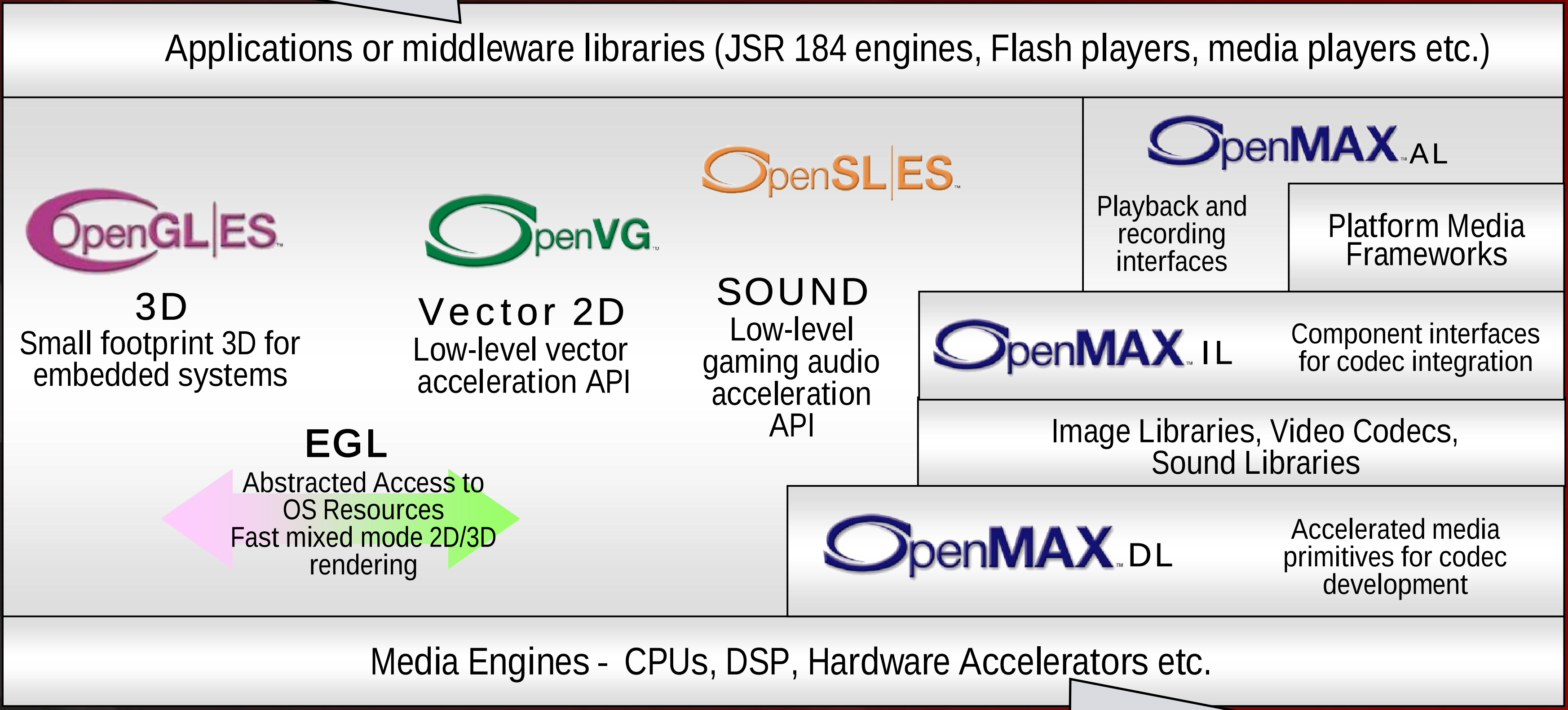
Wednesday	68 / 87
Thursday	82 / 93
Friday	76 / 84
Saturday	89 / 95

Weather info

NOKIA
Connecting People

Complete Khronos Media Stack

The Khronos API family provides a complete ROYALTY-FREE, cross-platform media acceleration platform



Applications or middleware libraries (JSR 184 engines, Flash players, media players etc.)

OpenGL ES
3D
Small footprint 3D for embedded systems

OpenVG
Vector 2D
Low-level vector acceleration API

OpenSL ES
SOUND
Low-level gaming audio acceleration API

OpenMAX AL
Playback and recording interfaces

Platform Media Frameworks

OpenMAX IL
Component interfaces for codec integration

Image Libraries, Video Codecs, Sound Libraries

EGL
Abstracted Access to OS Resources
Fast mixed mode 2D/3D rendering

OpenMAX DL
Accelerated media primitives for codec development

Media Engines - CPUs, DSP, Hardware Accelerators etc.

Khronos defines low-level, FOUNDATION-level APIs.
“Close to the hardware” abstraction provides portability AND flexibility

KHRONOS GROUP

Mobile 3D HW now!

NOKIA
Connecting People

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

NOKIA
Connecting People

ATI

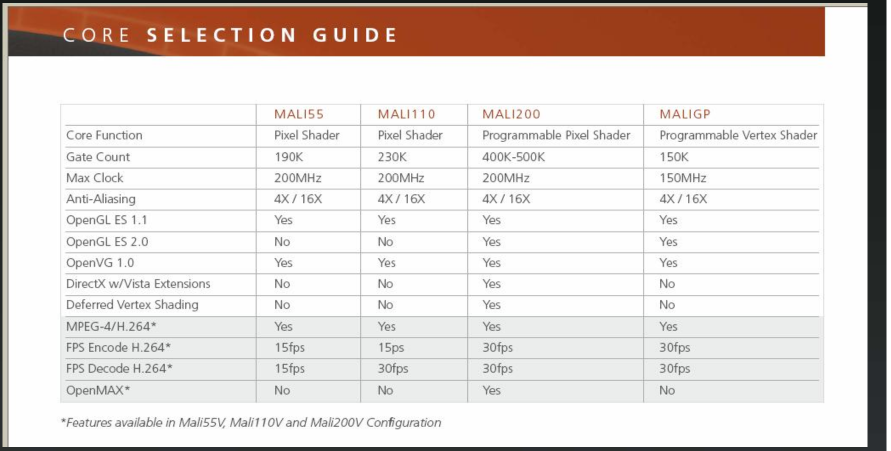
- Imageon 2300
  - OpenGL ES 1.0
  - Vertex and raster HW
    - 32-bit internal pipe
    - 16-bit color and Z buffers
    - Integrated QVGA buffer
    - Imaging / Video codecs
- Imageon 3D (for Qualcomm)
  - OpenGL ES 1.1
  - 3M Tri / s,100M Pix / s @ 100 MHz
- 2nd gen. Imageon 3D adds
  - OpenGL ES 1.1 extension pack
  - Vertex shader
  - HyperZ
  - Audio codecs, 3D audio
- Partners, customers
  - Qualcomm
  - LG SV360, KV3600
  - Zodiac

Bitboys

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

Falanx

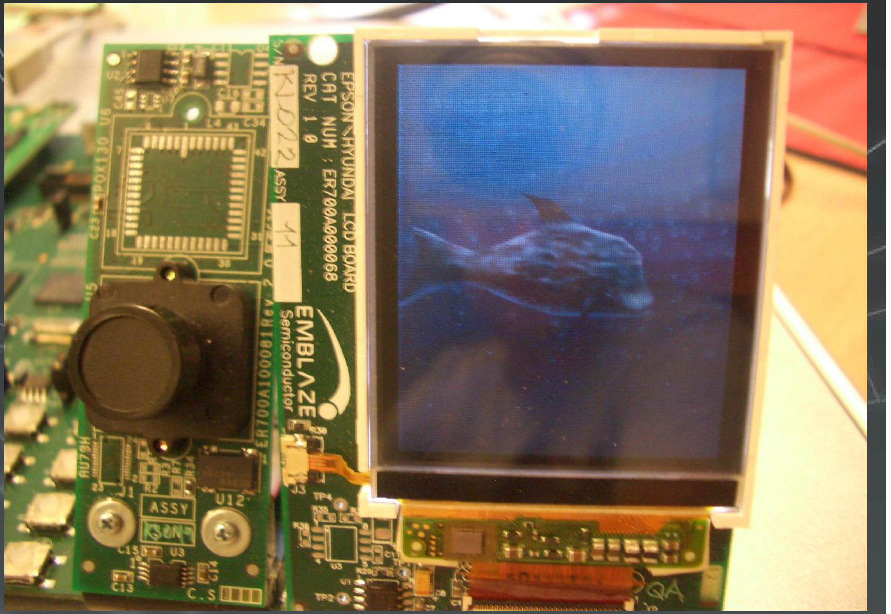
- **Mali 110**
  - OpenGL ES 1.1 + extensions
  - 4x / 16x full screen anti-aliasing
  - Video codecs (e.g., MPEG-4)
  - 170-400k logic gates + SRAM
  - 2.8M Tri / s, 100M Pix / s with 4xAA
- **Mali 200**
  - OpenGL ES 2.0, OpenVG, D3D Mob.
  - 5M Tri / s, 100M Pix / s, 11 instr. / cycle
- **Partners / Customer**
  - Zoran



CORE SELECTION GUIDE

	MALI55	MALI110	MALI200	MALIGP
Core Function	Pixel Shader	Pixel Shader	Programmable Pixel Shader	Programmable Vertex Shader
Gate Count	190K	230K	400K-500K	150K
Max Clock	200MHz	200MHz	200MHz	150MHz
Anti-Aliasing	4X / 16X	4X / 16X	4X / 16X	4X / 16X
OpenGL ES 1.1	Yes	Yes	Yes	Yes
OpenGL ES 2.0	No	No	Yes	Yes
OpenVG 1.0	Yes	Yes	Yes	Yes
DirectX w/Vista Extensions	No	No	Yes	No
Deferred Vertex Shading	No	No	Yes	No
MPEG-4/H.264*	Yes	Yes	Yes	Yes
FPS Encode H.264*	15fps	15ps	30fps	30fps
FPS Decode H.264*	15fps	30fps	30fps	30fps
OpenMAX*	No	No	Yes	No

*Features available in Mali55V, Mali110V and Mali200V Configuration



falanx MICROSYSTEMS

Imagination Technologies

POWERVR TECHNOLOGIES

- PowerVR MBX family
  - OpenGL ES 1.x
  - Vertex Geometry Processor (VS 1.1)
  - 300M Pix/s (effective) @ 100 MHz
  - 3.7M Tri/s (textured & lit) @ 100 MHz
- PowerVR SGX Family
  - OpenGL ES 2.x
  - Sizes 2-8 mm^2 (90nm)
  - 100-600M Pix / s (effective) @ 100 MHz
  - 1-7M Tri / s (textured & lit) @ 100 MHz
  - Vertex, Geometry, Pixel shaders
  - Imaging & Video codecs
- Partners / Customers
  - ARM, Intel, Freescale, Philips, Renesas, Samsung, Sunplus, TI
- Tile-based deferred rendering
  - Buffer triangles
  - Rasterize a block at a time
  - FSAA, high internal color depth

Imagination TECHNOLOGIES

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?

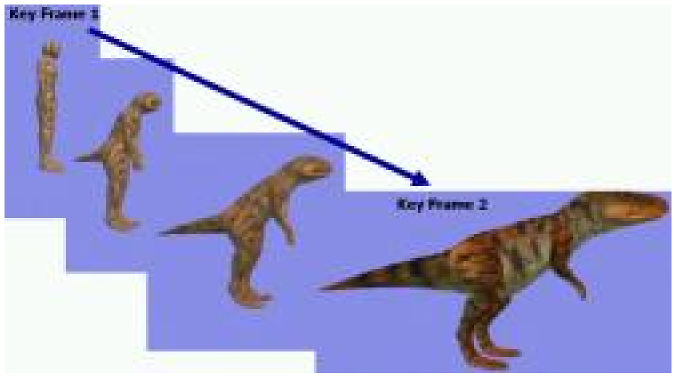
Z3D Graphics Engine

NVidia

- GoForce 3D 4800 handheld GPU
  - 3D geometry and rasterization HW
  - OpenGL ES 1.1, D3D Mobile, OpenVG 1.0
  - 1.2M tri / s, 80M pix / s (@ 100 MHz)
  - Programmable pixel micro shaders
  - 40 bit signed non-int (overbright) color pipeline
  - Dedicated 2D engine (bitblt, lines, alpha blend)
  - Supersampled anti-aliasing, up to 6 textures
  - <50mW avrg. dynamic power cons. for graphics
  - 3MPxl camera support, MPEG-4 video, audio
- Partners / Customers
  - Motorola, Sony Ericsson, Samsung, LG, Mitsubishi, Sendo, Gizmondo

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 ṡ own phones, Vodafone

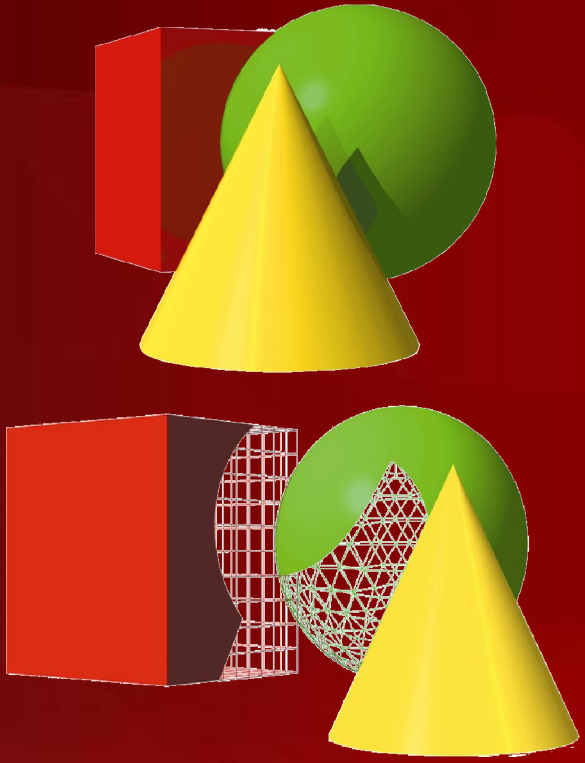
Three most important things

- Power usage
  - Internal bandwidth is the key, avoid off-chip accesses
- Power usage
  - Internal bandwidth is the key, avoid off-chip accesses
- Power usage
  - Internal bandwidth is the key, avoid off-chip accesses

NOKIA Connecting People

Output sensitivity

- 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



Local read-modify-write

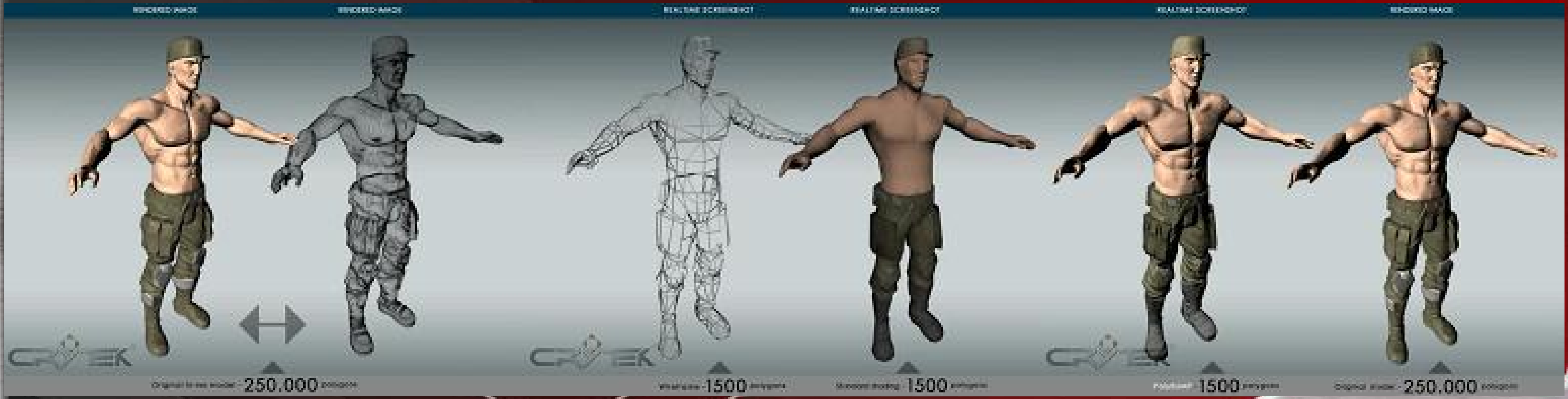
- Tiled rendering
  - Store the triangles in a buffer
  - Process triangles one screen tile at a time in local memory
  - Then block-copy out to display buffer

- Put the whole frame into eDRAM
  - It $ 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

NOKIA Connecting People

Just send less data

- Caching
  - textures, vertex arrays
- Texture compression
- Better pixel processing => fewer triangles
  - Example: bump maps (supported already in OpenGL ES 1.1)



NOKIA Connecting People

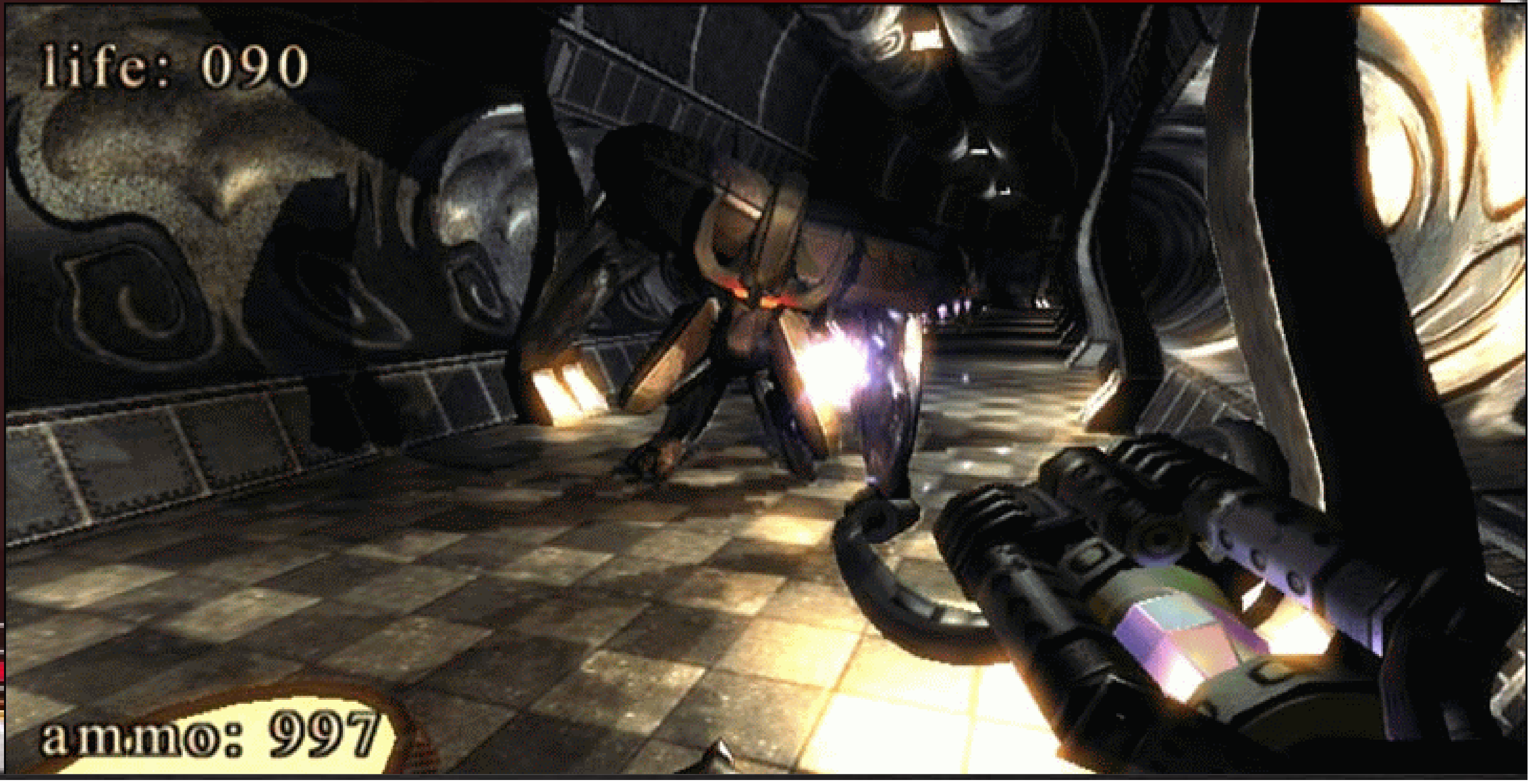
Avoid traffic, period.

- 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

NOKIA Connecting People

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!

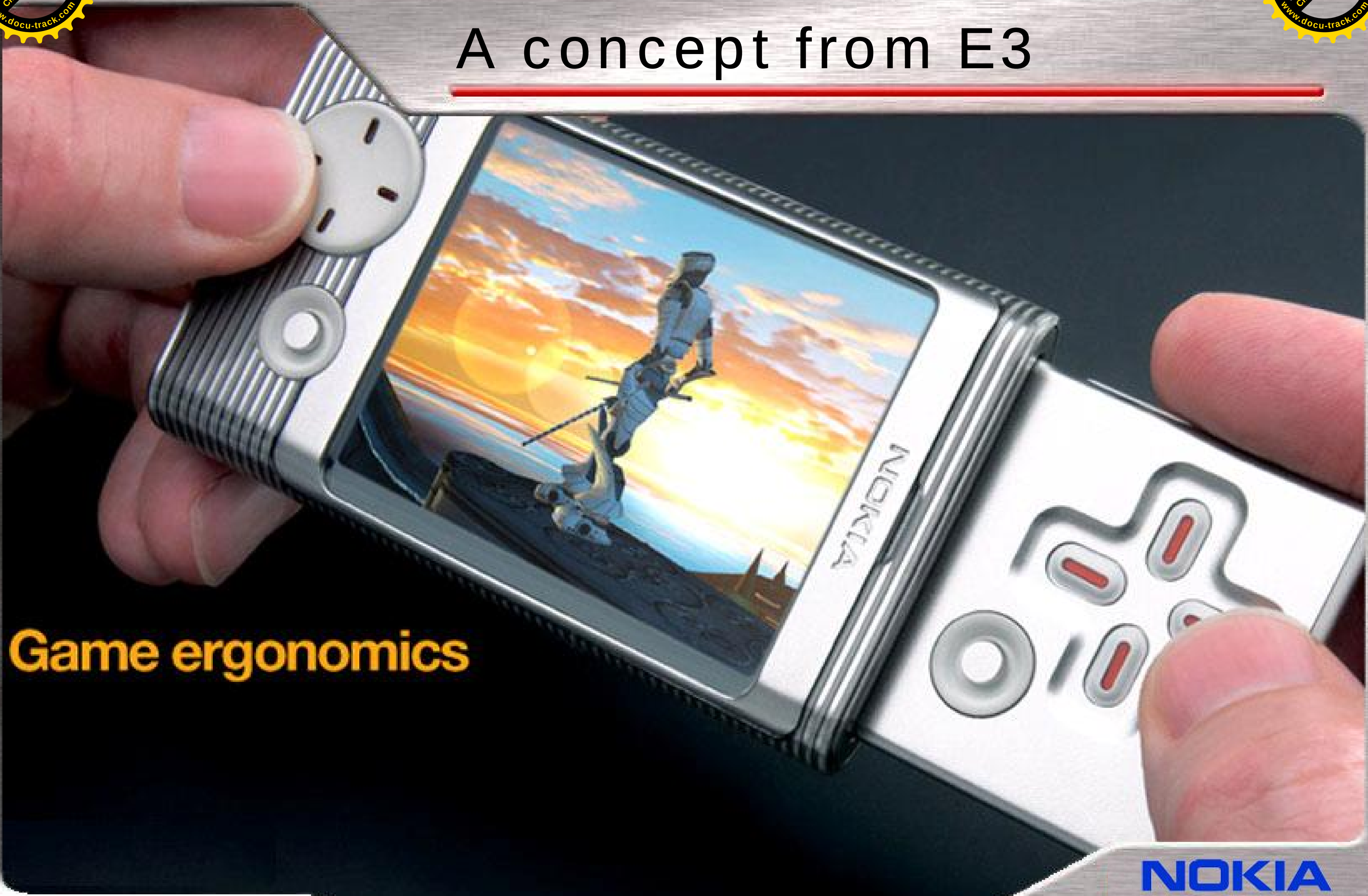


A concept from E3

Phone ergonomics

NOKIA Connecting People

A concept from E3



NOKIA
Connecting People

Demo

NOKIA
Connecting People

Summary

- Several new graphics APIs out there
    - OpenGL ES
    - M3G (JSR 184)
    - OpenVG
    - JSR 226
- They are already shipping in volumes
    - Era of HW acceleration started, continues
- Ubiquitous visual multimedia is reality
    - The next problem is to create and distribute content that uses this technology

NOKIA
Connecting People