Designing Games for IMAGEON™ 3D – Enabled Handsets

Claude Benoit
Technical Evangelist

August 31, 2006

CEDEC 2006
CESA DEVELOPERS CONFERENCE 2006

OVERVIEW

- ATI in the Industry
- IMAGEON™ 3D Feature Set
- Designing 3D Games for Performance

ATI In the Industry

World Leader in Visual Processing

Our Vision: Bring the most compelling and immersive multimedia solutions to PC and digital consumer markets

Handheld

- #1 media processor supplier
- Shipping in over 35 phones
- Industry leading video, imaging 3D graphics, audio, power

GPUs & Workstation

- #1 notebook supplier
- #1 desktop GPU supplier

Game Consoles

- GAMECUBE & Revolution
- Xbox 360

ATi 20 YEARS OF INNOVATION

DTV

- #1 HDTV silicon supplier
  - DTV Demodulation
  - HDTV Decoding
  - DTV Image Processing

Integrated Graphics

- #1 IGP chipset supplier (3rd party)

One of the World's Largest Fabless Semiconductor Companies

ATI & South Korea Telecom (SKTel)

- Strategic technical relationship bringing high performance hardware enabled “3D” & multimedia services to SKT customers
- Alliance formed in 2004
- SKT & ATI joint support of Graphic Instruction Graphic Acceleration (GIGA) 3D specifications
- Joint technical assistance to SKT selected OEM partners
- Joint ISV relations
  - *GIGA specific “One on One” developers support*
  - *SKT & ATI hosting GIGA Developers Conference*
- *Continued joint efforts on Next Generation GIGA specifications*

ATI / Qualcomm Strategic Initiative

"Jointly bringing the unprecedented next generation industry leading 3D technology platform for mobile devices"

- Provides highest level of 3D performance combined with MSM Launchpad Multimedia suite
- Consistent BREW/OpenGL ES for easy development of next generation content.
  - Games
  - User Interfaces
- Provides proven ISV content relationships & support programs
  - Increasing content quality and "time to market".
- Joint partnership surrounding 3D technology
  - MSM6xxx + IMAGEON W23xx Platform
  - Integrated IMAGEON 3D IP in MSM7xxx Convergence Platform

ATI / Nokia Strategic Relationship

"working closely together to drive high quality multimedia experiences such as music playback, 3D gaming, mobile TV, video and more"

- Long-term collaborative effort
- Integrated hardware, software & tools

The Khronos Family Picture

KHRONOS GROUP

OpenGL|ES

OpenML

OpenVG

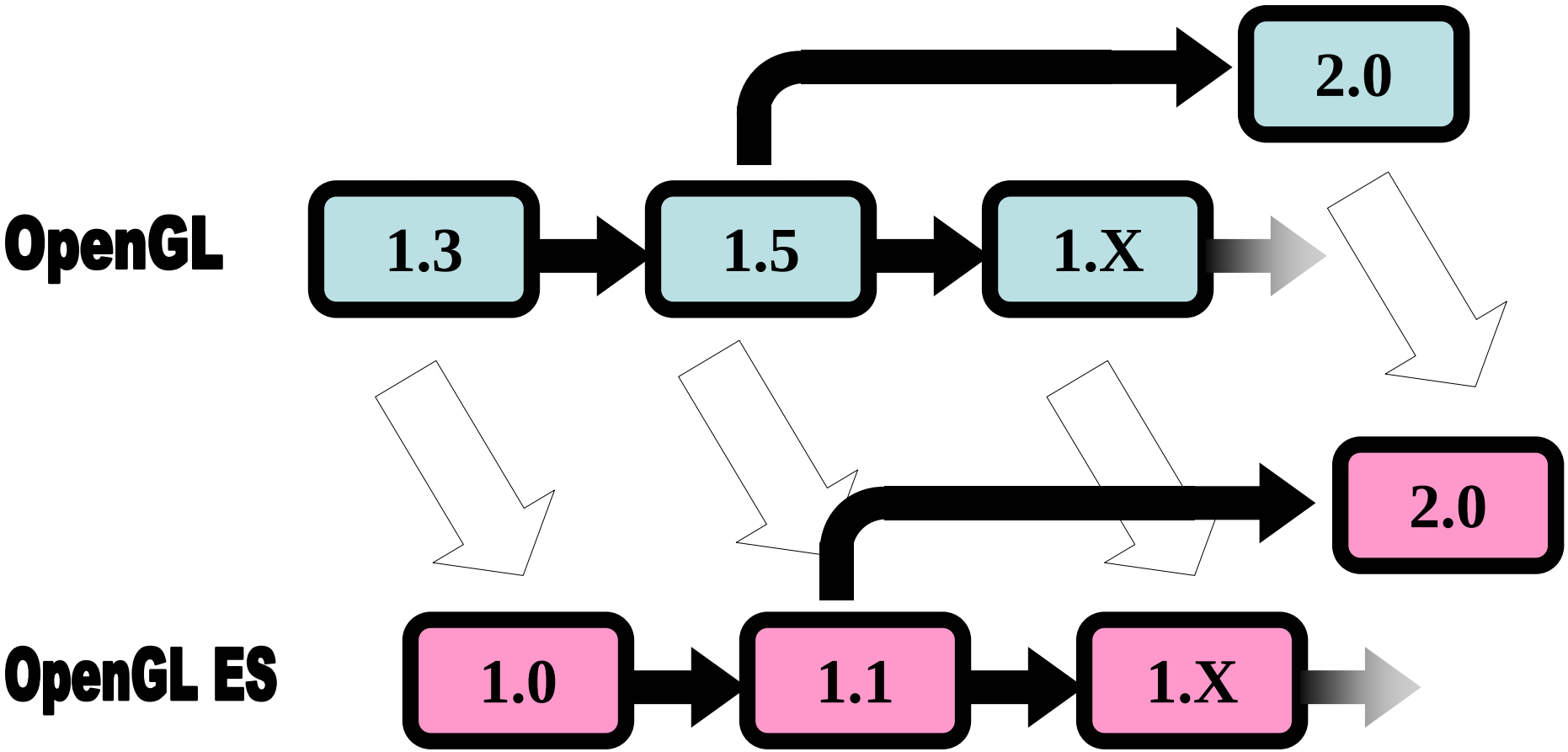
OpenMAX

OpenSL|ES

COLLADA

(*) as of April 2006

OpenGL ES Roadmap



IMAGEON™ 3D Feature Set

IMAGEON™ 3D Feature Set

	W230x	MSM75xx	W238x
	OpenGL ES 1.0	OpenGL ES 1.0 + Extensions	OpenGL ES 1.1 + Extension Pack + Extensions
Performance			
Triangle Rate (specs)	1 M triangles/sec	4 M triangles / sec	4 M triangles/sec
3D Pixel Fill Rate (specs)	100 M pixels/sec	125 M pixels/sec	125 M pixels/sec
Frame Buffer			
Supported Resolutions	QVGA Double-Buffered	QVGA / VGA Double-Buffered	VGA Double-Buffered
LCD Orientation	Portrait & Landscape	Portrait & Landscape	Portrait & Landscape
Color	16 bits/pixel	16 bits/pixel	16 bits/pixel
Z buffer	16 bits/pixel	16 bits/pixel	16 bits/pixel
Stencil	-	4 bits/pixel	8 bits/pixel
FBO: Frame Buffer Objects	-	-	Yes (1)

(1) Support for color, Z, stencil; all sub-formats according to OES_framebuffer_object

IMAGEON™ 3D Feature List

	W230x	MSM75xx	W238x
Geometry Engine			
Hardware Transforms	Yes	Yes	Yes
Vertex Buffer Objects	Yes	Yes	Yes
DX8-style Vertex Shaders	-	-	Yes
Hardware Point Sprites	-	Yes	Yes
Distance-based size factor	-	-	Yes
Vertex Skinning	-	Yes	Yes
Number of matrices/vertex	-	4	4
Total number of matrices	-	32	32
Hardware Lighting	-	-	Yes
Spotlight	-	-	Yes
Directional	-	-	Yes
Point	-	-	Yes

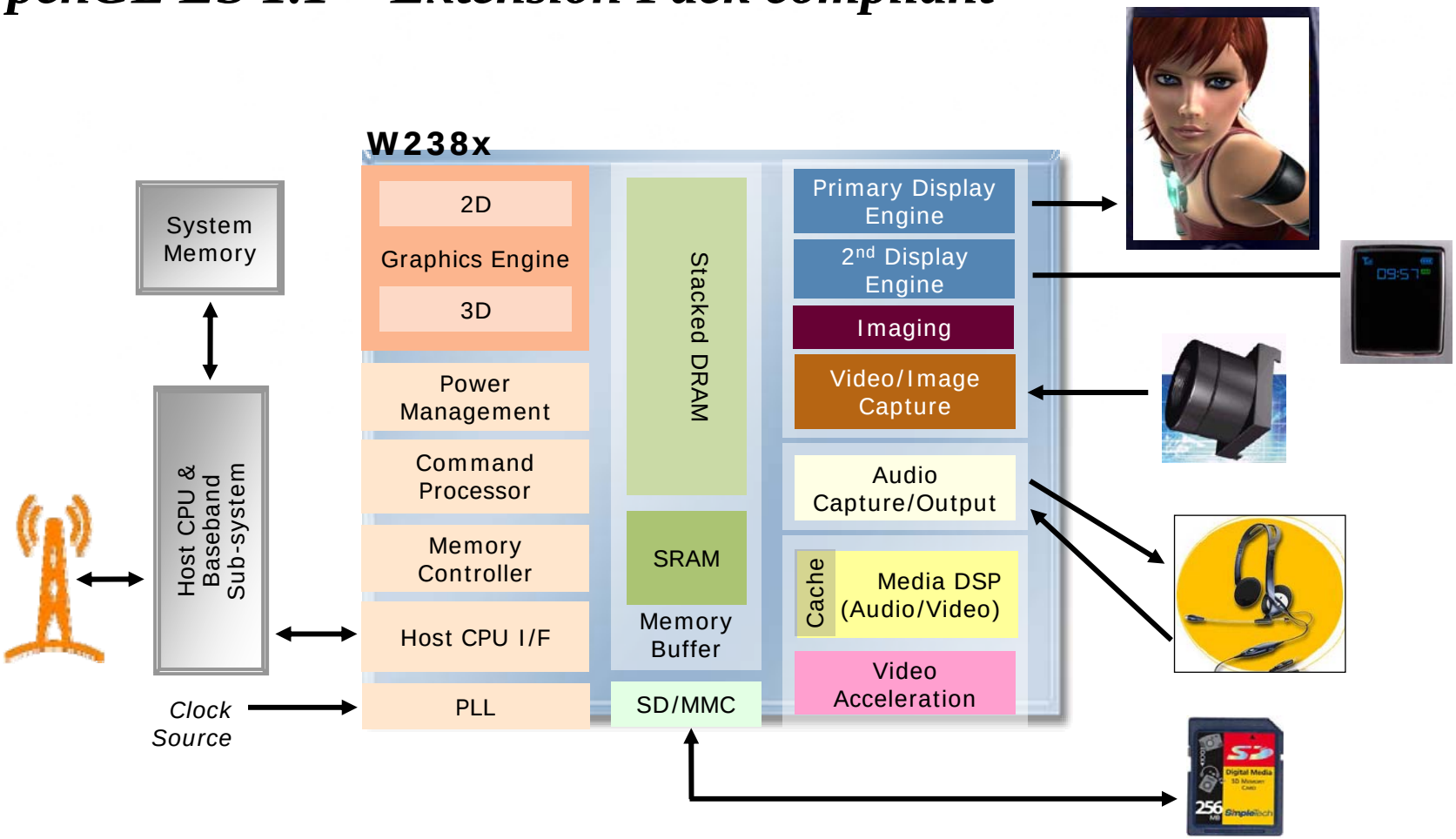
IMAGEON™ 3D Feature List

	W230x	MSM75xx	W238x
Texturing			
Texture pipelines	1	2	2
Texture Crossbar	-	Yes	Yes
Texture Compression (1)	-	Yes	Yes
Extended Texture Data Formats (2)	-	Yes	Yes
DOT3 Bump Mapping	-	Yes	Yes
Projective Textures	-	-	Yes
Cubic Mapping	-	-	Yes
Early Z Culling	-	-	Yes

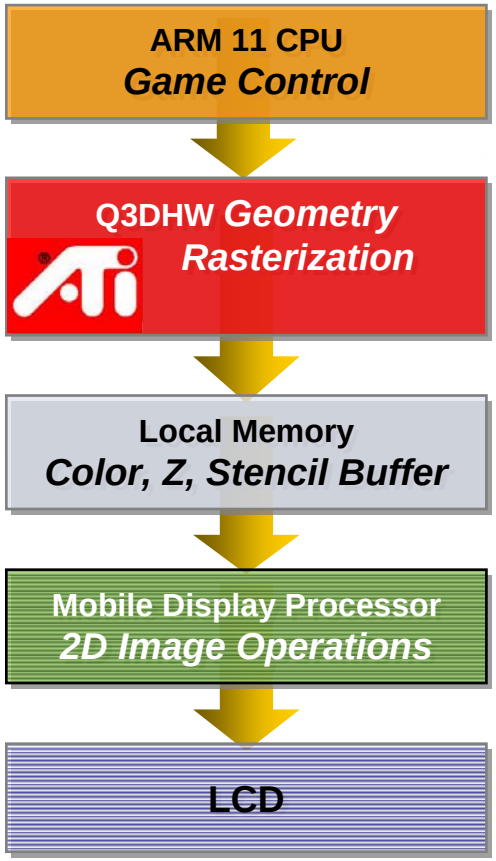
(1) ATI_TC for both RGB and RGBA
(2) Extended Texture Coordinate Data Formats: 4.4 / 8.8 / 4.12

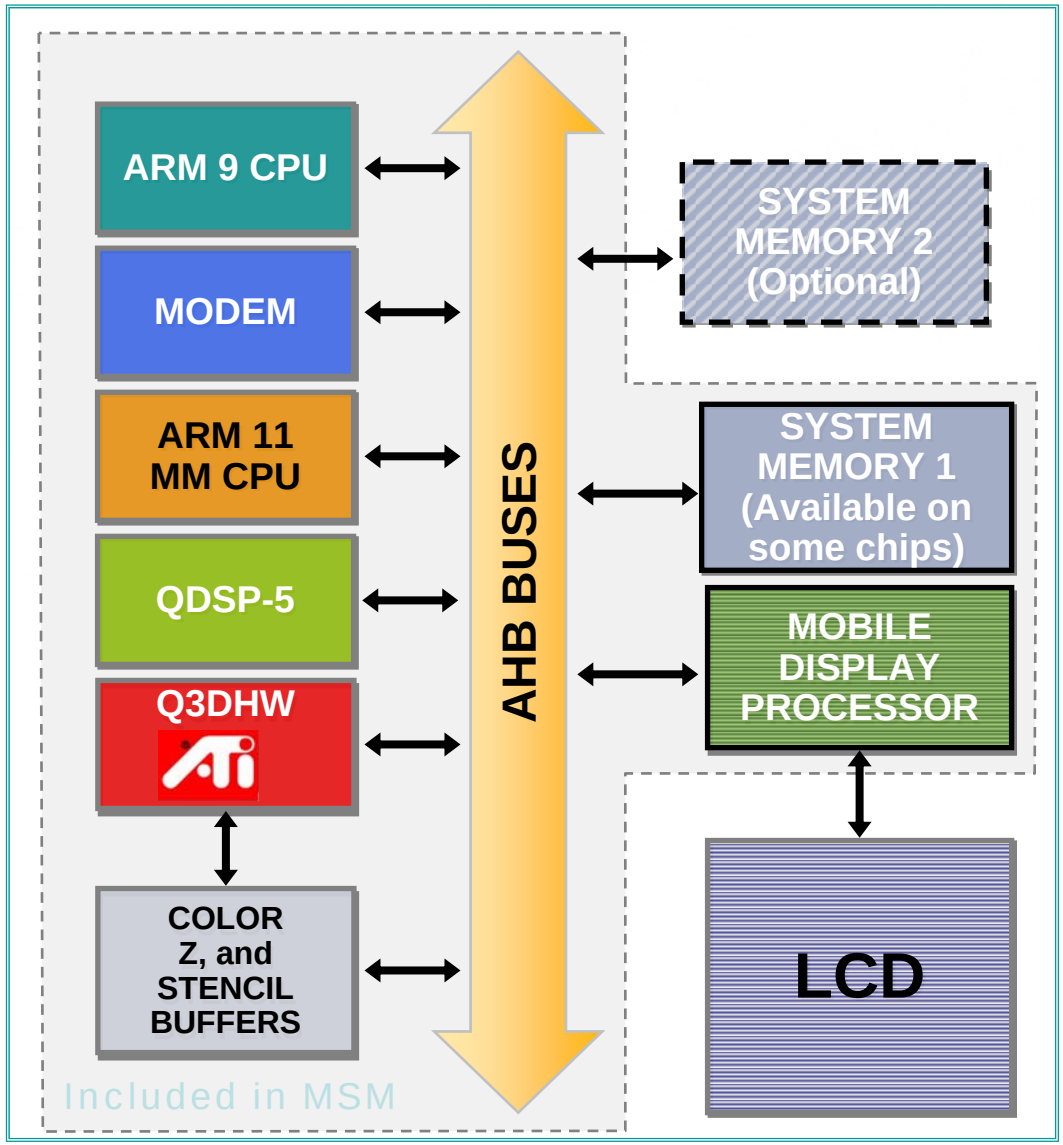
W238x – Architecture Overview

OpenGL ES 1.1 + Extension Pack compliant



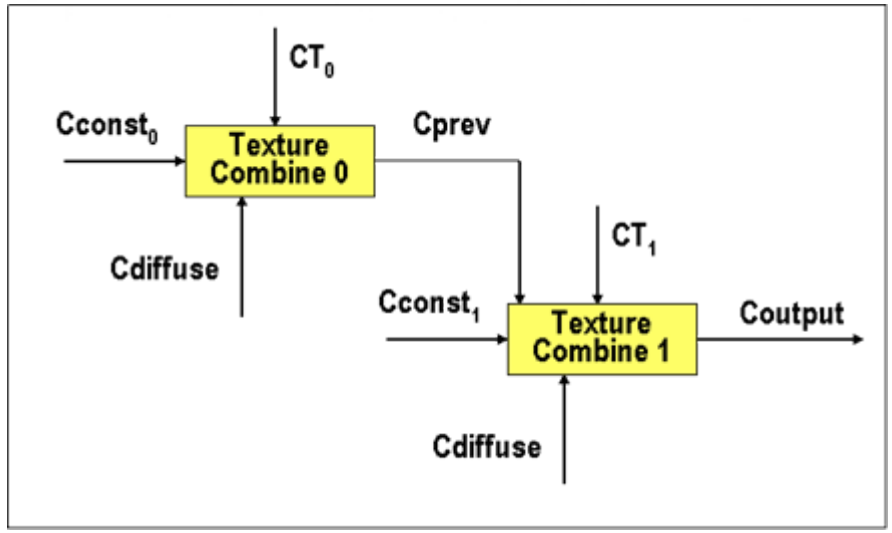
MSM7xxx -- Architecture Overview





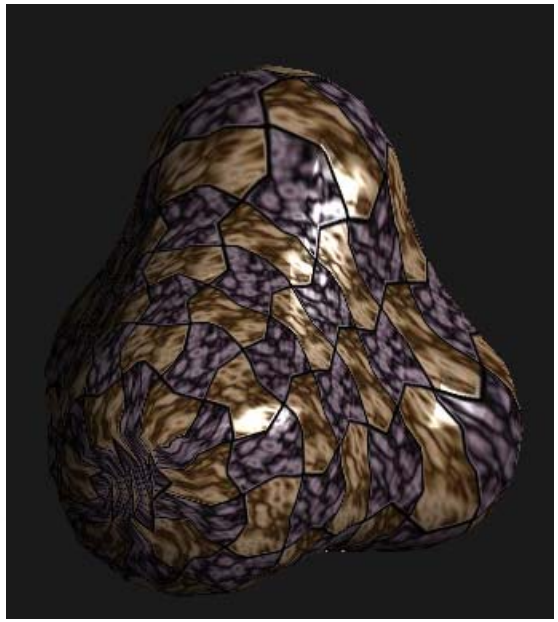
OpenGL ES 1.1 – Texture Crossbar

- Important for mobile space because:
  - Allows effects that previously had to be done in multiple passes to be done in a single pass
  - Improves performance over multipass because it reduces transform and framebuffer blending costs
  - Introduces flexible new equations and inputs that provide for new effects

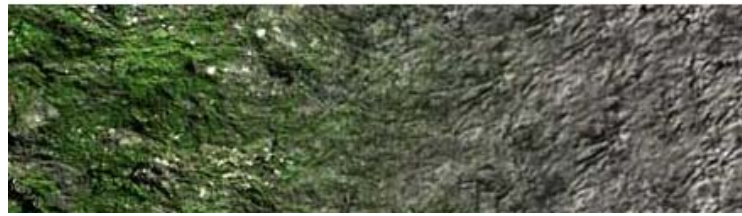


OpenGL ES 1.1 – Texture Crossbar

LightMapping
MODULATE (A*B) = T0*T1
T0 – Base
T1 – Lightmap



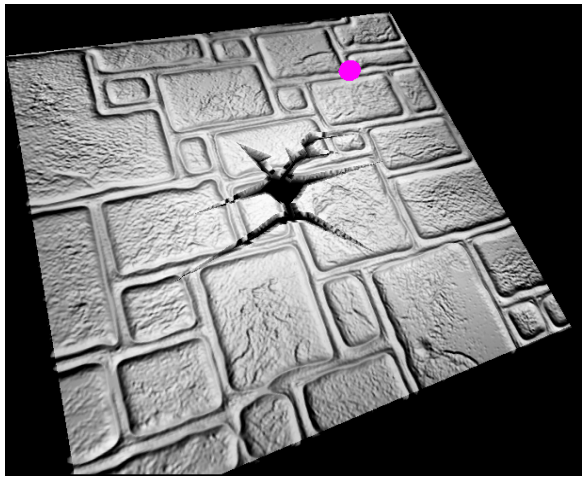
“Specular” Highlights
ADD (A+B) = (T0 * Cdiffuse) + T1
T0 – Base
T1 –Specular
Cdiffuse = N.L

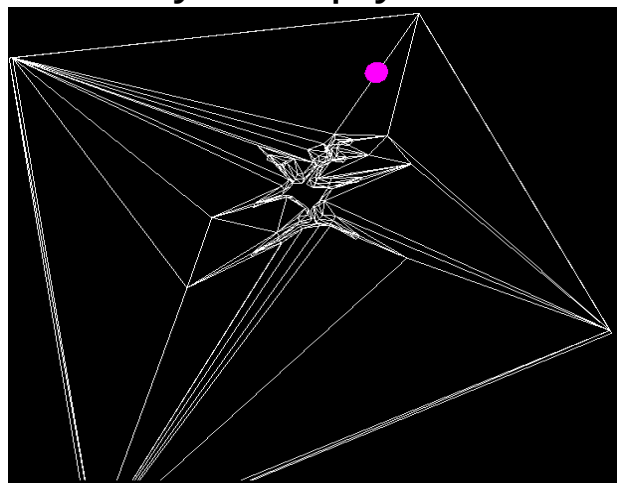


Texture Interpolation
INTERPOLATE (A,B,C) = (T0 * Cd.a) + (T1 * (1-Cd.a))
T0 – Grass Texture
T1 – Rock Texture
Cdiffuse.a = Interpolation factor
Source: Delphi3D (http://www.delphi3d.net/)

OpenGL ES 1.1 – DOT3 Bump Map

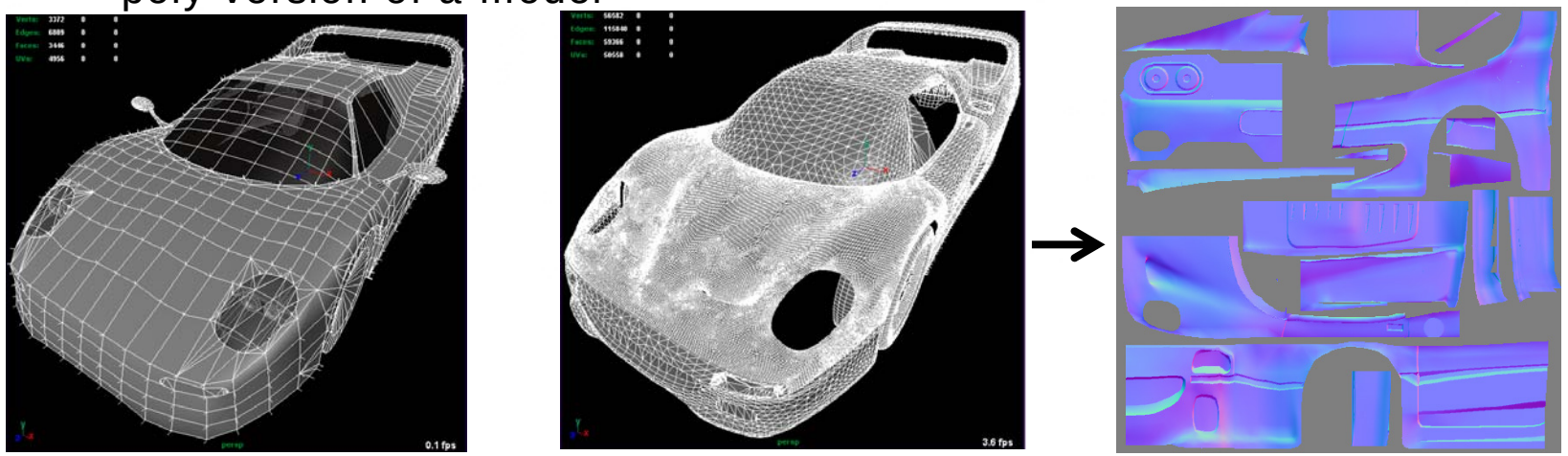
- New blend equation DOT3 introduced in OpenGL ES 1.1
- Provides the ability to do the diffuse lighting equation (N.L) on a per-pixel basis
- Important for mobile because:
  - Provides the appearance of surface complexity with less polygons
  - Reduces geometry memory footprint
  - If not fill limited, can be done very cheaply



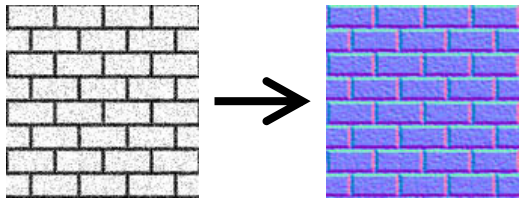


OpenGL ES 1.1 – DOT3 Bump Map

- Many tools provide the ability to generate normal maps
  - ATI's NormalMapper tool generates normal maps from a low and high poly version of a model

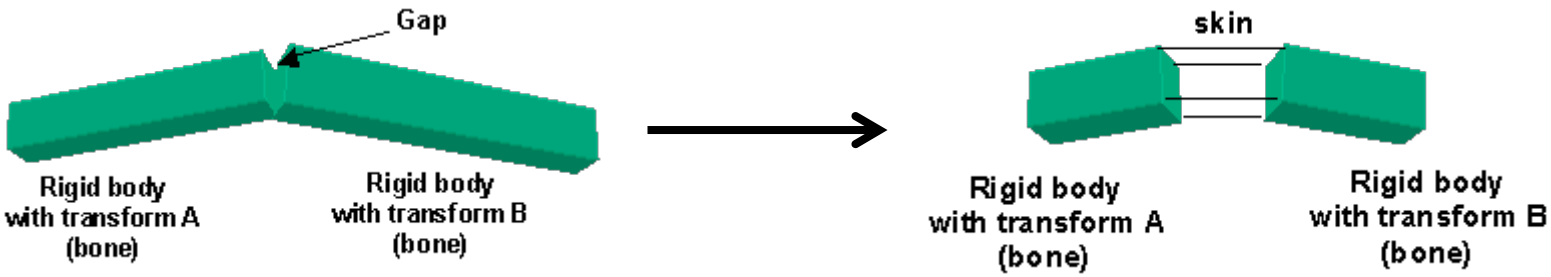


- DOT3 maps can also be created directly from height maps
  - ATI's NormalMapGenerator tool uses Söbel filter to generate DOT3 map



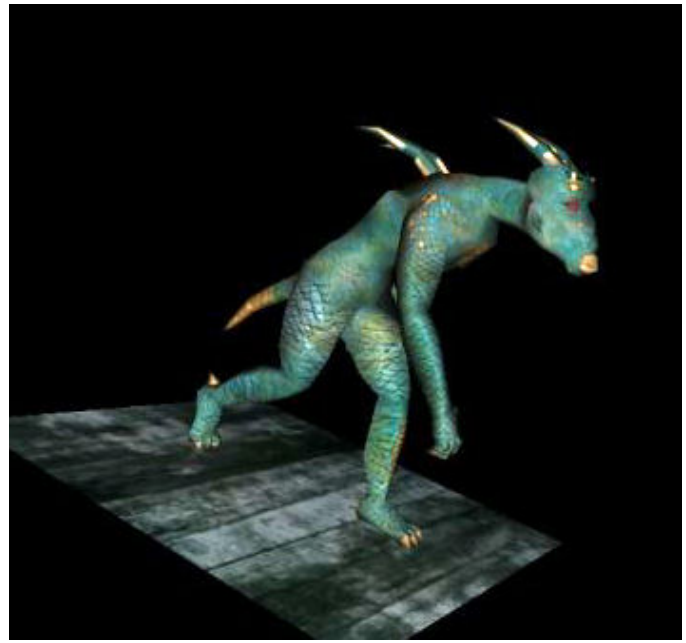
OpenGL ES 1.1 – Skinning

- What is vertex skinning?
  - Solves the problem of gaps in between bones
  - Provides for more realistic looking characters
- Matrix palettes introduced in OpenGL ES 1.1
- Skinning computation is offloaded from the CPU onto the GPU
  - Important for systems with no floating-point support



OpenGL ES 1.1 – Skinning

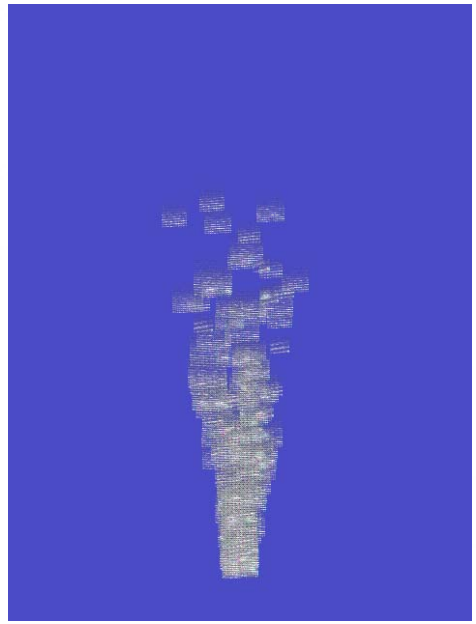
- The matrix palette size and number of vertex units are implementation dependent
- If the matrix palette size is exceeded, break up draw calls by sets of bones
- The keyframe interpolation should still be done on the CPU
  - This is a relatively simple computation which should only happen once per-frame per-animation
- ➔ This feature is fully orthogonal
  - All GL lighting works



OpenGL ES 1.1 – Point Sprites

- Provides a fast method for drawing screen-aligned sprites
- Important for mobile space because:
  - Reduces geometry footprint and transform cost – only one vertex per-point sprite
  - Reduces CPU burden for aligning billboard to screen
  - Useful for rendering particle systems as well as other 2-D effects such as:
    - Lens flares
    - Light glows
- Point sprite is rendered as a textured quad
- Any geometry rendered as GL_POINTS will be drawn as a sprite when point sprites are enabled
- Completely orthogonal
  - texture environment and blending can be used as normal

OpenGL ES 1.1 – Point Sprites



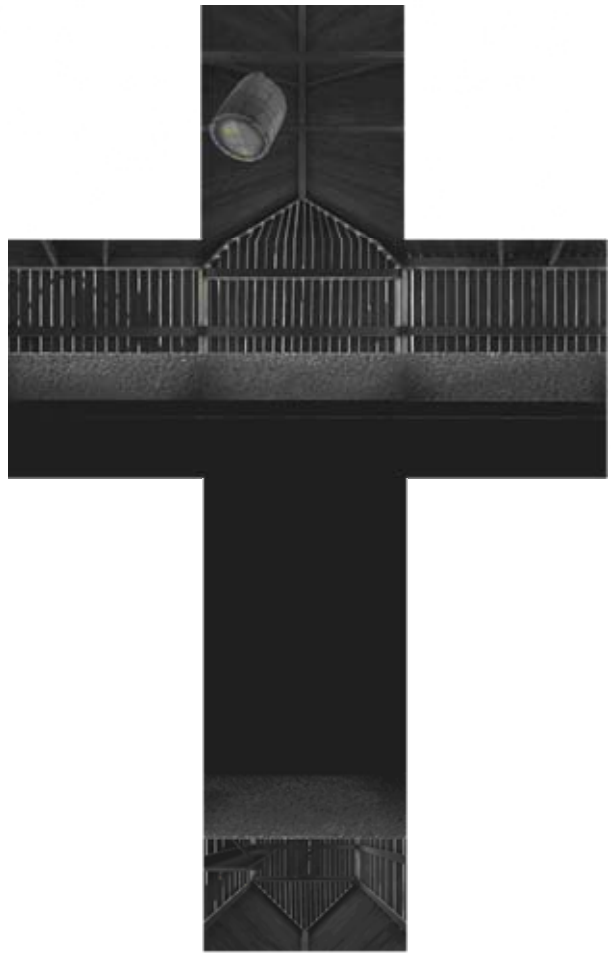
- Point size can be controlled in one of two ways:
  - A global point size can be set for all points
  - A point size can be specified per-vertex to control the size per-point
- glPointSize()
  - Specifies a global point size
- glPointSizePointer()
  - Vertex array entrypoint to control point size per-point
  - Used like any of the other vertex arrays functions

- In either mode, points can be distance attenuated (W238x)

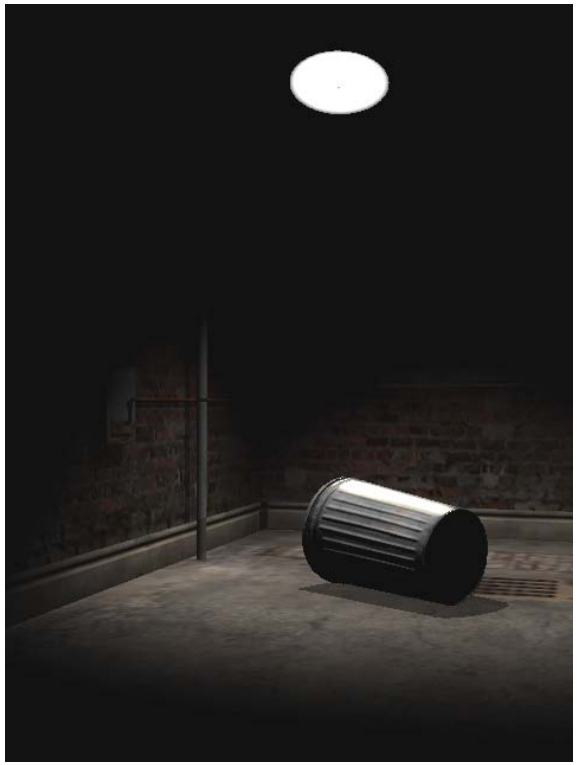
$$atten(d) = \frac{1}{a + b * d + c * d^2}$$

OpenGL ES 1.1+ – Cubic Mapping

- W238x
- Accurate real-time reflections in handheld 3D games
  - Given normal & position, compute reflection vector
  - Use reflection vector as texture coordinate
  - Per-pixel computed reflection vector is used to index one of 6 faces of cube
    - Generate per-pixel reflection value
- Per-pixel “specular” lighting
  - Per-vertex specular highlight requires very finely tessellated geometry
  - With cube-maps you can compute specular color per pixel
    - And not require lots of geometry



OpenGL ES 1.1+ – Cubic Mapping



Lighting

Water

Reflections

OpenGL ES 1.1+ – Framebuffer Objects

- W238x

- Provides for more efficient render-to-texture
- The API is contained fully within the GL API
  - no reliance on the window-system or EGL
- Multiple contexts are not required:
  - Removes the performance hit for switching contexts
  - Texture formats that differ from that of the window can be created on one context
- A single framebuffer object allows rendering to an unlimited number of texture objects
- Provides more flexible binding of buffers:
  - Depth/stencil buffers can be used for multiple renderable textures
  - This potentially reduces the memory footprint for certain applications

OpenGL ES 1.1+ – Vertex Shader

- W238x
- Alters or computes new texture coordinates, diffuse color at vertices, sprite point size, fog as part of the vertex processing step.
- Similar to DirectX vs_1_0 assembly
- Sample shader:

```
// modelview matrix is stored in c0 - c3
vs_imageon_1_x
mul    r0,   P.x, c0
madd   r0,   P.y, c1, r0
madd   r0,   P.z, c2, r0
madd   r0,   P.w, c3, r0
mov    oP, r0
```

ATI_TC Texture Compression

- MSM7xxx, W238x
- Significant reduction in texture memory & bus bandwidth requirements
- Tools
  - ATI_Compress -- compression library
  - The Compressonator™ -- standalone app
  - All available on the ATI Developer web site:
  - www.ati.com/developer

Designing 3D Games for Performance

Organizing Your Scene Data - Overview

- Scene DAG organization & traversal
  - Z ordering
  - Textures and state changes
  - Triangle batch size; triangle organization (lists / strips / fans)
  - Best GL data types
  - Bus and bandwidth limitations
  - Geometry transform; Rasterizer
- OpenGL ES feature usage
  - Per-vertex computations; Lighting and fog
  - Texturing; Size, format and filtering
  - Frame buffer

Drawing Order

- Z Ordering
  - Front to back sorting order of the rendered geometry for opaque
  - Back to front sorting order for transparent objects
  - Early Z culling increases performance! (W238x)
    - Avoids texture computations when the object is behind another one.
    - Rough Z sorting enough (main character, scene, skybox)

- State changes
  - Texture thrashing is very costly; reuse textures; best if all the needed textures are preloaded and used across multiple frames
  - State changes are costly, so you should try to render together models that share common features consecutively.
  - Handheld device drivers are much simpler than their PC counterparts, so redundant state changes are not always examined and trivially rejected.

- **Recommendation**: sort by Z first and then by texture and state

Triangle Organization

- Triangle fans and strips
  - Models specified as fans and strips allow vertices to be reused, thus lowering the cost of transferring data.
  - **Recommendation**: use strips; concatenate strips with identical material & state info using degenerate triangles
- Degenerate triangles
  - Some modeling packages insert degenerate triangles as markers or to start a new object
  - Very helpful since the hardware likes large triangle packets.
  - **Recommendation**: use degenerate triangles to help send large triangle batches
- Triangles Batch size
  - Trying to pack as many triangles in a single drawing call allows the driver to work more efficiently in passing data to the 3D graphics engine.
  - **Recommendation**: the larger the better

Best Data Types

- Using GL_FIXED / GL_SHORT / GL_BYTE
  - Internal conversion from byte/short/fixed to floating point for transform and rasterization involves **no cost**
  - Allows a smaller memory footprint for vertices and texture coordinates
  - Smaller data types saves on bus bandwidth
  - Texture coordinates can be adjusted to [0,1] using a texture matrix

- *GL_ATI_extended_texture_coordinate_data_formats*
  - Fixed point format (4.4, 4.12, 8.8) to save space
  - No need for a texture matrix; aids vertex transform performance

- **Recommendation**
  - Use GL_SHORT for vertex data and texture coordinates.
  - Use GL_ATI_extended_texture_coordinate_data_formats for texture coordinates

Bus Bandwidth

- From the CPU to the GPU
  - Determines how much data you can send to the 3D engine per frame (vertices, picture data, state commands)
- Vertex Buffer Objects
  - OpenGL ES extension designed to offset the costs of transferring data across the bus.
  - Data is preloaded once and referenced multiple times.
  - Used for static models or rarely deformed models.
- **Recommendation**: Use VBOs, *ALWAYS!*
  - It reduces the internal bandwidth requirements (W238x)
  - Avoids internal copying around (MSM7xxx)

Per-vertex Computations

- Lighting computations
  - MSM7xxx: Performed by the device driver (CPU)
  - W238x: Performed by the GPU (vertex shader unit)
  - Per-vertex lighting is computation intensive
  - **Recommendation**:
    - try to avoid GL lighting if you can; use texture maps instead for lighting effects.
    - W238x: Limit yourself to 2 lights, one point and one directional
- Fog computations
  - Performed by the device driver (CPU) on W230x and MSM7xxx.
    - CPU is doing other tasks also
  - Performed by the GPU for W238x (vertex shader unit)
  - Per-vertex fog is computation intensive
  - **Recommendation**: Turn off fog if you don't need it.

Organizing Your Scene Data - Texturing

- Texturing
  - Hardware support for perspective correction at no cost
  - Use texture formats close to HW formats
  - **Recommendation**: Best: RGB565, RGBA4444, RGBA5551
- Texture Pipeline
  - Dual texture pipeline
  - Huge performance gains with mip-mapping
  - Best setting: Linear-Map-Nearest
  - Use texture compression for higher performance
  - **Recommendation** : Use multipass, compression, mip-mapping.

Organizing Your Scene Data – Frame Buffer

- Clear: Color / Stencil / Z
  - Fast dedicated memory for color, Z, stencil
  - Avoid manual 2D ops
  - System memory is slower than dedicated graphics memory

- **Recommendations**:
  - Always clear the entire Z / stencil buffer per frame.
  - Avoid clearing the color buffer if all pixels are drawn per frame.
  - Avoid mocking around the frame buffer

W230x Benchmark Case: MotoGP

Java MIDP 1.0

Nokia N-Gage

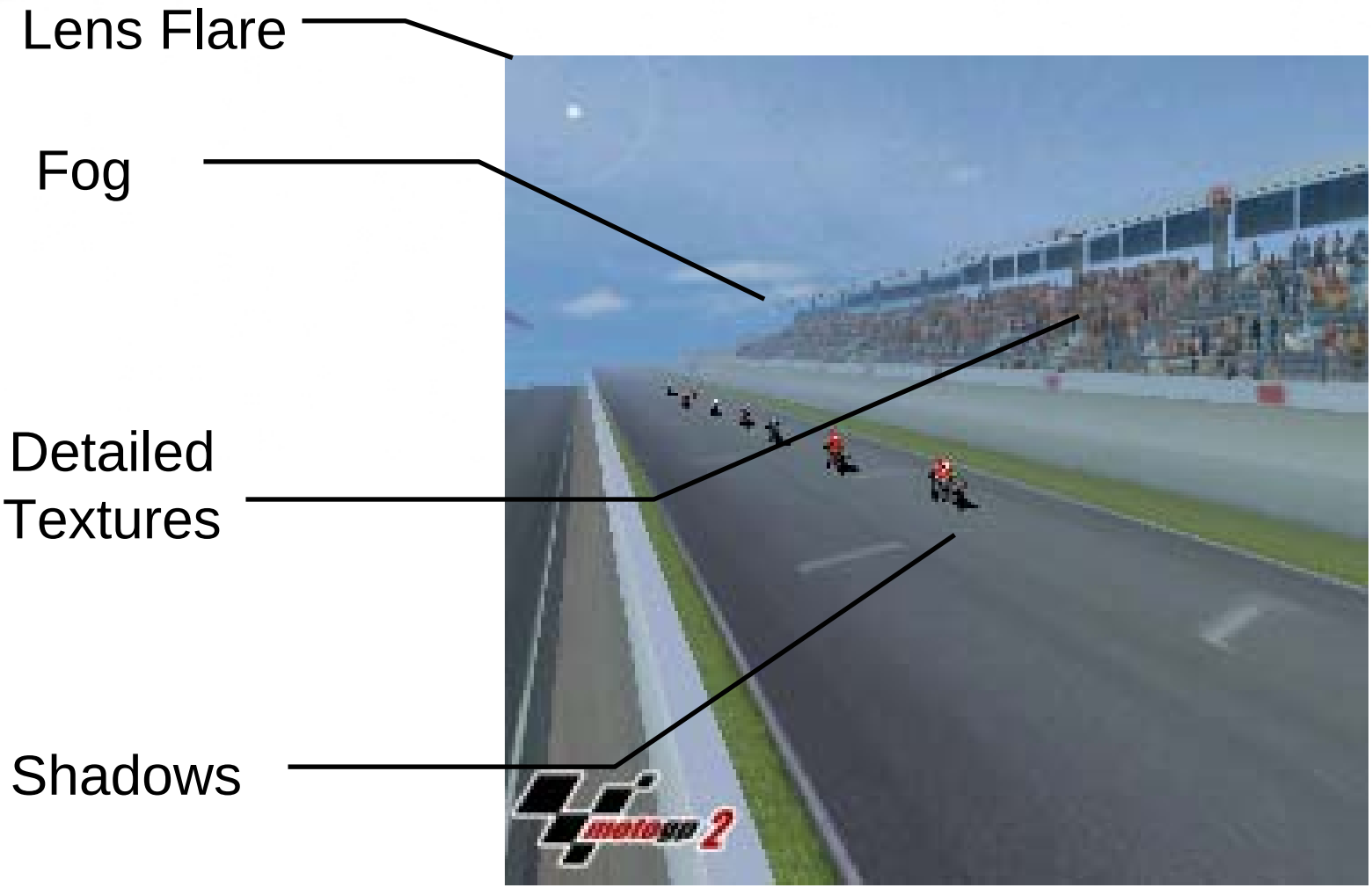
IMAGEON™ W230x

W230x Benchmark Case: MotoGP

Microsoft XBOX

IMAGEON™ W230x

W230x Benchmark Case: MotoGP



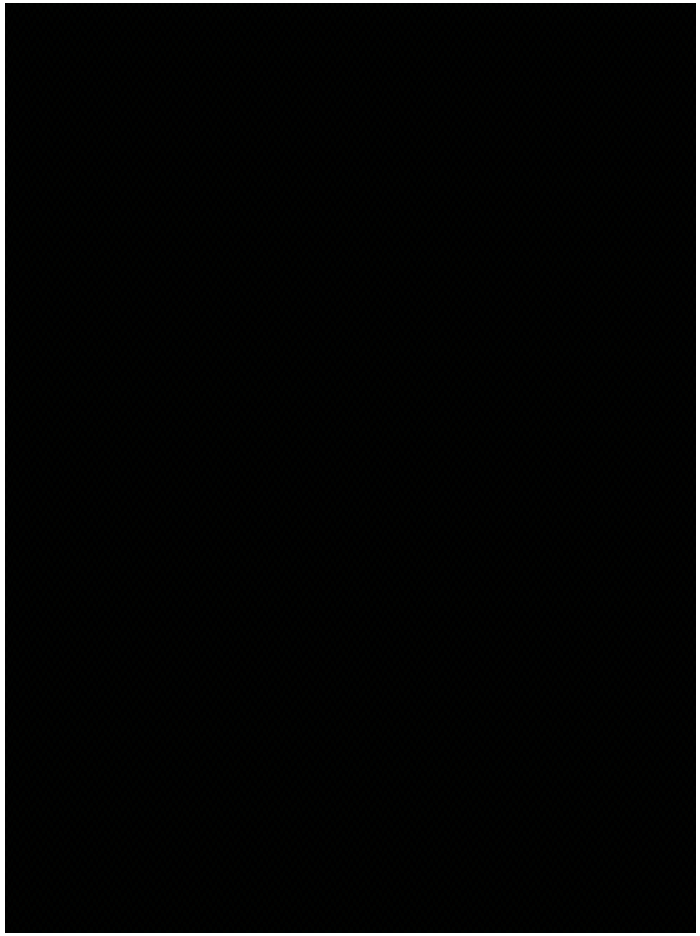
W230x Benchmark Case: MotoGP

- ATI Reference Platform:
  - ARM9-based @ 168MHz, 64Mb
  - W2300 @ 84MHz
  - 8MB framebuffer
  - QVGA screen
- Performance:
  - 3808Kb of texture
  - 7826 triangles / frame avg
  - 11862 vertices / frame avg
  - Frame Rate: 25 fps avg
- Throughput (avg):
  - ~ 200,000 triangles / sec
  - ~ 300,000 vertices / sec
  - CPU limited! (game logic)
  - Your mileage may vary!

W230x Benchmark Case: MotoGP

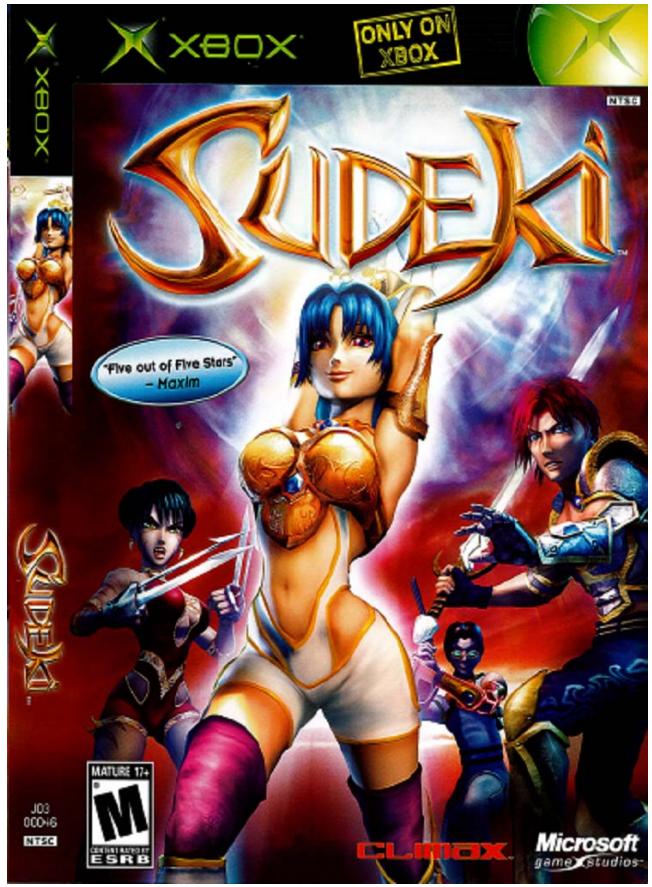
- Scene Data:
  - 8 bikes: 700 triangles each
  - Track: ~ 14,000 triangles
- Features Used:
  - Mipmapping
  - Gouraud Shading
  - Alpha Blending
  - Single textures
  - Fog
  - Alpha textures
- Optimizations Used:
  - VBO / Mesh lists
  - Mipmapping
  - Frustrum culling (CPU)

W230x Benchmark Case: MotoGP



W238x Benchmark Case: Sudeki

- Sudeki is a conversion of an existing combat Role-Playing Game (RPG) from a native XBox gaming console code base to a functioning demonstration version for ATI's IMAGEON™ 3D technology.



CLIMAX

W238x Benchmark Case: Sudeki

- The IMAGEON™ W238x multimedia processor adds more texturing flexibility, character skinning and particle effects than the IMAGEON™ W230x thus enabling games that require
  - 1) dynamic character based games such as action, sports and RPGs,
  - 2) richer, organic environments (rather than rigid), and
  - 3) higher polygon count and highly flexible texturing capabilities.

CLIMAX

W238x Benchmark Case: Sudeki

- ATI Reference Platform:
  - ARM9-based @ 400 MHz, 64Mb
  - W2380 @ 96MHz
  - 16MB framebuffer
  - VGA screen
- Throughput (avg):
  - ~ 1.2M triangles / sec (VGA)
  - ~ 3.1M triangles / sec (QVGA)
  - Your mileage may vary!

CLIMAX

W238x Benchmark Case: Sudeki

QUESTION PERIOD