EuroGraphics Italian Chapter
Catania, Feb 24th 2006

3D Graphics for Embedded Systems
From standards, through R&D to applicable innovations

Massimiliano Barone
Imaging, Rendering, Human Machine Interaction Platform
Advanced System technology

Agenda
- Introduction to reference standard
- OpenGL-ES 1.1 overview
  - Immediate mode rendering
- OpenGL-ES 2.0 overview
- Computing rendering models for low cost applications
  - Sort Middle rendering overview
  - Memory footprint and bandwidth analysis
  - Optimization techniques
- Conclusions
Agenda

- Introduction to reference standard
- OpenGL-ES 1.1 overview
  - Immediate mode rendering
- OpenGL-ES 2.0 overview
- Computing rendering models for low cost applications
  - Sort Middle rendering overview
  - Memory footprint and bandwidth analysis
  - Optimization techniques
- Conclusions

---

Khronos Group

- Founded in Jan 2000 by a number of leading media-centric companies
- Focused on the creation of open standard, royalty-free APIs for authoring and Sw/Hw acceleration
- 3 main working groups

2D Vector Graphics 3D Vector Graphics Multimedia Audio and Video
OpenGL|ES

- Light-weight version of OpenGL
- Not a simplified version about pipeline implementation!
- Rarely-used commands no more supported
- Few simplifications
  - Only triangles, no quads or polygons
  - Only vertex arrays, no vertex-per-vertex communication
  - Only textures, no bitmap
  - Only 2D textures, no 3D textures
  - No accumulation buffer, feedback,…
OpenGL ES Roadmap – Two Tracks

Spec release              2003  2004  2005

OpenGL ES 1.0             OpenGL ES 1.1     OpenGL ES 1.2
Streamlined Subset        Minimized Fixed Function          Package of extensions that enhance functionality and will be included in OpenGL ES 1.2

OpenGL ES 1.0             OpenGL ES 1.1     OpenGL ES 1.2
Minimized Fixed Function  Extension Track                      Package of extensions that enhance function and will be included in OpenGL ES 1.2

OpenGL ES 1.5             OpenGL 2.0        OpenGL 2.0
New Functionality        GLSL                     GLSL

Shader programmability using GLSL. Eliminate redundant fixed functionality —> simple, streamlined API

Eliminate redundancy and workstation functionality —> simple, streamlined API
Enhanced functionality for emerging hardware

Roadmap track for programmable hardware
Roadmap track for fixed function hardware

Game Complexity vs time

Game complexity is Increasing to provide better user experience
Target Market Segments

Automotive  →  OpenGL-ES  →  Consumer

Mobile

Agenda

- Introduction to reference standard
- OpenGL-ES 1.1 overview
  - Immediate mode rendering
- OpenGL-ES 2.0 overview
- Computing rendering models for low cost applications
  - Sort Middle rendering overview
  - Memory footprint and bandwidth analysis
  - Optimization techniques
- Conclusions
Fixed function pipeline

- OpenGL 1.5 as reference

- In particular
  - Minimum of 2 multitexture, support for bump mapping and per pixel lighting
  - Draw Texture
  - User defined clip planes
  - Enhanced Point sprite and point sprite arrays
  - Vertex Buffer Objects
  - Vertex skinning
  - Auto mip map generation

OpenGl-ES 1.0 1.1
OpenGL|ES 1.1 Example

Seeking better quality

Fixed-function Pipeline

Programmable Pipeline

Lighting per vertex

Lighting per pixel
Agenda

- Introduction to reference standard
- OpenGL-ES 1.1 overview
  - Immediate mode rendering
- OpenGL-ES 2.0 overview
- Computing rendering models for low cost applications
  - Sort Middle rendering overview
  - Memory footprint and bandwidth analysis
  - Optimization techniques
- Conclusions

Programmable pipeline

- OpenGL|ES 2.0 offers more programmability
  - Vertex Processing Unit replaces Transfo & Lighting, ...
  - Fragment Processing Unit replaces Texturing, Fog and Alpha, ...
OpenGL-ES 1.1 – 2.0

**Vertex Shader**
- Operate on one vertex at a time (SIMD)
- Return vertex in homogeneous coordinates
- Each vertex has own transformation

**Fragment Shader**
- Operate on one fragment and associated data at a time (SIMD)
- Each fragment has own filters
- Cannot modify (x,y) of fragment
- Update framebuffer or texture memory
GLSL for OpenGL 2.0, OpenGL|ES 2.0

- Introduction of a new programming language
  - GLSL = GL Shading Language
- Used to program Vertex and Fragment processors
  - Vertex shaders, Fragment shaders
- Common (95%+) between VS and FS
- GLSL for GL|ES 2.0 : subset of GLSL for GL 2.0

Example
Agenda

- Introduction to reference standard
- OpenGL-ES 1.1 overview
  - Immediate mode rendering
- OpenGL-ES 2.0 overview
- Computing rendering models for low cost applications
  - Sort Middle rendering overview
  - Memory footprint and bandwidth analysis
  - Optimization techniques
- Conclusions

Quake III

Sort Middle vs Immediate mode rendering
Immediate Mode vs Sort Middle

- IMR works on whole screen
- SMR works on one tile at time

Scene Capture process

For each tile
Binner overview

- Association of primitives with tiles
- Primitives and Back-end commands as input.
- Context pointers and primitive pointers generation.
- Scene buffer and tile display lists storage.
- Bounding box generation and exploration
- Flushing command.

Bounding Box generation

- Bounds into tile space
- Exploration culls tiles not covered by primitives
Example: Outcode method

- Based on
  - Edge equation and logic ops on outcodes
- For each edge, evaluate
  - Sign(E = aX + bY + c)

Parser overview

- Read Display lists
- Get data from Scene buffer for current tile
  - Context
  - Primitive’s attributes
- Works on scene already captured, stored and binned
  - 1 frame latency
Agenda

- Introduction to reference standard
- OpenGL-ES 1.1 overview
  - Immediate mode rendering
- OpenGL-ES 2.0 overview
- Computing rendering models for low cost applications
  - Sort Middle rendering overview
  - Memory footprint and bandwidth analysis
  - Optimization techniques
- Conclusions

Tile coverage or Overlap

- Average number of tiles covered by the same primitive
- Depend on application/game:
  - triangle orientation and dimension (#pixels)
- Depend on tile’s number, not its dimension
- Relation between overlap and tile’s number not linear.

<table>
<thead>
<tr>
<th>Tile dimension</th>
<th>16</th>
<th>32</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tile number</td>
<td>1200</td>
<td>300</td>
<td>75</td>
</tr>
<tr>
<td>overlap</td>
<td>32,815</td>
<td>16,673</td>
<td>10,630</td>
</tr>
</tbody>
</table>

640 x 480 screen size – Quake III
Considerations on data transfer

- Need to save power consumption on memory bandwidth
- Sort Middle uses memory to store Color, texture and scene buffers
- While Immediate Mode for Color, texture and depth buffers

SMR vs IMR on data transfer

- SMR shows overdraw factor = 1 (colour buffer)
- Increasing overlap implies increasing Texture bandwidth
- Depth/stencil memory bandwidth needed only by IMR
- Scene buffer memory bandwidth needed only by SMR:
  - proportional to tile coverage and scene complexity
  - Increases linearly with screen size at fixed tile dimension
1. **Total External Bandwidth**
   - **QVGA 30 fps Quake III**
   - Graph showing bandwidth requirements for different components:
     - Texture BW
     - Colour Buffer
     - Depth Buffer

2. **Internal vs External Memory needs**
   - Table listing requirements for different components:
     - Colour
     - Depth
     - Texture
     - Scene
   - **SMR**
     - Internal, Tile size AND External, screen size
     - Internal, Tile size
     - No
     - Internal, limited cache
     - Yes, external buffer
   - **IMR**
     - Internal OR External, screen size
     - Internal OR External, screen size
     - Internal, limited cache
     - No
An example

<table>
<thead>
<tr>
<th>Colour</th>
<th>Depth</th>
<th>Texture &amp; Stencil</th>
<th>Scene</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMR</td>
<td>64 kB</td>
<td>64 kB</td>
<td>16 kB</td>
</tr>
<tr>
<td></td>
<td>300 kB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMR</td>
<td>300 kB</td>
<td>300 kB</td>
<td>16 kB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>13 MB</td>
</tr>
</tbody>
</table>

QVGA screen size, true colour, 32bit depth&Stencil, Quake III
128*128 tile size

Antialiasing

- Increase linearly external color and depth bandwidth and memory requirement for IMR.
  - Depend on antialiasing factor
- While for SMR increase linearly the scene buffer bandwidth.
  - Function of overlap factor
  - Only internal tile buffer is at super-resolution
  - External colour buffer is at screen resolution
An example: 4x Anti aliasing factor

**Memory bandwidth**

**Memory footprint**

---

**Agenda**

- Introduction to reference standard
- OpenGL-ES 1.1 overview
  - Immediate mode rendering
- OpenGL-ES 2.0 overview
- Computing rendering models for low cost applications
  - Sort Middle rendering overview
  - Memory footprint and bandwidth analysis
  - Optimization techniques
- Conclusions
**IMR Optimization techniques**

- **Geometry pipeline**
  - Early Back Face Culling
  - Guardband Clipping
- **Rasterizer**
  - Occlusion culling based on hierarchical depth buffer
- **Fragment pipeline**
  - Early depth test
  - Texture decompression
  - Depth and Stencil lossless compression
  - Color lossy and lossless compression

**SMR Optimization techniques**

- **Geometry pipeline**
  - Early Back Face Culling
  - Guardband Clipping
- **Rasterizer**
  - Front to back display list parsing
    - Once scene is captured, objects can processed in different orders
  - Per tile Occlusion culling
- **Fragment pipeline**
  - Early depth test
  - Texture decompression
  - No color and depth compressions is needed
Early Back face culling

- Back face culling moved after ModelView
- For symmetric objects, can remove up to 50% of primitives
- Algorithm works in a different coordinate space

Occlusion Culling

<table>
<thead>
<tr>
<th>Block Side</th>
<th>Percentage of Pixel Area (%)</th>
<th>Percentage Of Triangles (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Max</td>
<td>Min</td>
</tr>
<tr>
<td>Quake 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>47.31</td>
<td>0.00</td>
</tr>
<tr>
<td>8</td>
<td>37.75</td>
<td>0.00</td>
</tr>
<tr>
<td>16</td>
<td>29.89</td>
<td>4.92</td>
</tr>
<tr>
<td>32</td>
<td>43.82</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Early Depth/Stencil Module

OpenGL-ES fragment pipeline order

- Read Textures from memory (also for hidden fragments)
- Read Z/S(x,y)
- Write Z/S(x,y)
- Read if hazard
- Read Textures from memory (visible fragments)
- Write to buffer

Texture Bandwidth reduction

- Bandwidth reduction ~ 10%-15% on Quake III compared to 10-20% of fragments killed by the late depth test.
Texture Compression

- For RGBA texture maps
  - S3’s DXTC1,3,5
  - Ericsson i-Packman (OpenGL-ES 2.0)
  - STM-TC
  - Up to 4:1
  - lossy

- For normal maps
  - ATI 3Dc (under discussion for OpenGL-ES 2.0)
  - 4:1 compression ratio
  - lossy

Buffers Compression

- Lossy ColorBuffer codec
  - Based on Vector Quantization algorithm
  - High Quality: on average 36 dB on final rendered scene (Quake3)
  - Fixed Compression ratio [2:1]

- Lossless ColorBuffer codec
  - Based on color de correlation and Huffman coding
  - No Quality loss
  - Variable Compression ratio, on average 1.9 (min:1.4 max:2.4)

- Lossless DepthBuffer codec
  - Based on color de correlation and Huffman coding
  - No Quality loss
  - Variable Compression ratio, on average 3.7 (min:3 max:4.6)
### Total IMR Bandwidth

**QVGA @ 30fps**

<table>
<thead>
<tr>
<th>Component</th>
<th>Without Caches</th>
<th>Caches + Depth &amp; Color Compression</th>
<th>Caches + Depth &amp; Color + Texture Compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color Module</td>
<td>74.4 MB/s</td>
<td>50.7 MB/s</td>
<td>50.7 MB/s</td>
</tr>
<tr>
<td>Depth Module</td>
<td>61.5 MB/s</td>
<td>26.1 MB/s</td>
<td>26.1 MB/s</td>
</tr>
<tr>
<td>Texture Units</td>
<td>244.6 MB/s</td>
<td>27 MB/s</td>
<td>6.75 MB/s</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>380.7 MB/s</strong></td>
<td><strong>103.8 MB/s</strong></td>
<td><strong>103.8 MB/s</strong></td>
</tr>
</tbody>
</table>

**Compression**

- **3.66**
  - Color Cache + Comp: 50.7 MB/s
  - Depth Cache + Comp: 26.1 MB/s
  - Texture Cache: 27 MB/s
  - **Total:** 103.8 MB/s

- **4.55**
  - Color Cache + Comp: 50.7 MB/s
  - Depth Cache + Comp: 26.1 MB/s
  - Texture Cache + Comp: 6.75 MB/s
  - **Total:** 63.55 MB/s

**381 MB/s to 84 MB/s**

---

### IMR Bandwidth  QVGA

**Graph**

- **Texture**
- **Depth**
- **Color**

- **without compression**
- **caches with color and depth compression**
- **plus texture compression**

**Y-axis:** MB/s

**X-axis:** without compression, caches with color and depth compression, plus texture compression
Conclusion

- Open standard – supported by industry is the way!
  - Khronos, OpenGL-ES

- Roadmap from fixed to programmable graphics pipeline
  - different complexity (HW support)
  - different markets
  - Better picture quality

- 2 low cost main architectures emerged
  - Sort middle pipeline vs IMR
  - There isn’t a best in absolute sense
  - Each one has pros/cons depending on target application