Qualcomm® HW Accelerated 3D: Adreno

- Moving content-quality forward requires hardware acceleration
  - Up to 1024x768 screen-resolution by mid-2009
  - Blend effects and composition of 3D with other media types
  - Multiple texture support with combiners

- Qualcomm is enabling Hardware 3D on all its chipset tiers
Adreno Product Family

- **OpenGL-ES 1.0**
  - Adreno 100 and Adreno 110
  - Commercial for 3 years, high volumes
  - Entry-level hardware acceleration

- **OpenGL-ES 1.0+/OpenGL-ES 1.1**
  - Adreno 120 and Adreno 130
  - Commercial now in Asia, entering US by July
  - High performance fixed function pipeline with texture combiners and matrix palette extensions

- **OpenGL-ES 2.0**
  - Adreno 200, Adreno 210 and Adreno 220
  - Commercial end of this year
  - High performance, flexible shader pipeline
Adreno Graphics Platform

So many devices
Challenge for developers

Current HW accelerated 3D Content

- Main SKU is Software
- HW is treated as an incremental feature: e.g. bilinear filtering, marginally higher-res textures
- Engines that do support HW do so with orthogonal render paths

Market is shifting, will you be able to make the transition from incremental hardware support, to full support?

Problems

- It’s an embedded device, you get the best performance from coding to the hardware
- **But:** HW manufacturers don’t make it easy
Architecture of one platform...

Adreno 100, 110

- ARM9 CPU
- QDSP-4
- Adreno 100/110 GPU
- Disp Ctrl

- SYS AHB
- APPS AHB
- DMA AHB
- MemCtrl
Problem: Doing R&D for each platform is way too expensive
Qualcomm’s Adreno Tools

Adreno tools

- Lowering cost of tuning content for HW
- All the tools necessary to create cutting-edge 3D content for Adreno platforms
Adreno Tool Packages

- Developed alongside the hardware: brought up alongside drivers, and system

**QX Engine™ SDK**
- Maya, 3DS Max Exporters
- Particle System Editor
- QStrip Triangle Stripper
- QXTextureBuilder texture optimizer
- Full rendering and animation engine
- Full Source Code

**Adreno Profiler**
- On-Device profiling
- Directed analysis
- Hardware metric access
- API Traces
- Powerful debugging features
First Prototype Woes

‘After porting, I’m getting 5 FPS?!’
Graphics Pipeline
Introducing Adreno Profiler

- Quickly identify and analyze bottlenecks on multiple platforms
Driver Instrumentation

- Graphics driver is extensively instrumented
- No application changes required
- Available on commercial devices
Case Study: Neocore Optimization

- Platform: Q3Dimension R4.0 (MSM7201)
- Initial performance: 5-10fps
- After optimization: 30fps
Demo
Graphics Pipeline

- CPU
- Geometry Storage
- Geometry Processor
- Rasterizer
- Fragment Processor
- Texture Storage + Filtering
- Frame Buffer

Vertices  Pixels
QXTtextureBuilder Tool

- Easily converted all textures to mipmapped + ATITC
  - Memory footprint reduced by 75%
- Huge GPU caching benefit
QStrip Tool

- Converted all meshes from discrete triangles to triangle strips
- Also enabled frustum culling in QX Engine

```c
/* compute strip */
Qstrip* pStripify = QstripCreate();
ushort* dStrip = QstripComputeStrips(pStripify, indexnum, indexdata);
ulong nStripLen = QstripGetStripLength(pStripify);
```
Adreno Profiler

- HW and system-level real-time performance metrics
- Powerful frame analysis and debugging features
- Supports current and future Qualcomm Adreno platforms, including upcoming OpenGLES v2-based cores

- Available Today on commercial devices
To Join Qualcomm’s Adreno Developer Program, simply email:

Tim Leland
tleland@qualcomm.com
QX Engine 1.2 Features for Adreno 120/130

Layered Textures

DOT3 Bumpmapping

Specular Mapping

Particle Engine & Authoring System

New APIs and utilities
Case Study – Step 1

Neocore Build 1

- Metrics graph: 11 FPS
- Profiler statistics: Statistics show textures are not ideal
- Forcing 1x1 Textures jumps performance to 22 FPS
- Bottleneck is in the texture fetch stage
Case Study– Step 2

Neocore Build 2

- Used QX Texture Converter to create ATI_TC, Mipmapped textures
- Metrics graph: 20 FPS
- We have moved the bottleneck, it is no longer in the texture pipeline
- Statistics gathering hints that triangle-strips are not being used
- Pulling back the camera shows unnecessary off-camera rendering
- Our bottleneck is in the front-end: too many unnecessary polys + not optimized
Case Study— From 11 to 30 FPS in 3 steps

Neocore Build 3

- Used QStrip to generate triangle strips, added QX Engine frustum culling
- Metrics graph: 30 FPS
- We’re done

Culled geometry

Optimized Triangles

Optimized Textures