Power Friendly GPU Programming

Maurice Ribble
Senior Staff Engineer
Qualcomm
Overview

- Mobile GPU Overview
- Demo
- OpenGL ES® Power-Reducing Programming Tips
- Tools
The Future is Mobile

- Smartphone shipments overtook PCs in 2011 and are growing at a faster rate.

Source: Smartphone - Gartner, Mar. '12, PCs - Gartner and ABI, Mar. '12)
- Battery size is smaller
- Expected runtime is longer
- More subsystems to run
• OpenGL ES 3.0 has just been announced.
• OpenGL ES removes redundancy and rarely used features from OpenGL.
• OpenGL ES adds power optimizing features.
• OpenGL ES is a more testable API.
Demo: Fortress

- God rays
- Lens flare
- Real time shadows
- Per fragment lighting
- Cloth simulation
- Real time reflection and refraction
- Water simulation
Power and Performance

- Power and performance are related.
Textures

- Saving bandwidth is saving power
  - Compress textures whenever possible
    - Saves power by minimizing bus traffic
    - ETC2 and EAC compression formats are best in class (OpenGL ES 3.0)
  - Keep textures small
  - Volume detail textures can be repeated to great effect
    - The GL_MIRRORED_REPEAT wrap mode is especially useful for symmetrical textures like volumetric light maps
  - Volumetric detail textures and light maps are often greyscale
- Draw from front to back whenever possible
- Most mobile GPUs have early-z rejection or depth sorting
- Many games are still rendering the skybox first

*Electopia benchmark*
Render Target Usage

- Default Framebuffer
- FBO
Render Target Usage

- Draw all FBOs before you start rendering the default framebuffer.
- Use minimum required precision with shaders
- Use “mediump” (often fp16) most of the time
  - Often this is sufficient for fragment shaders
  - Watch out for build in functions like normalize() exceeding mediump
  - Overflow and other problem cases
- Use “highp” (fp32) where needed
  - Often required for vertex shaders
OpenGL ES 3.0 Power Optimizations

- **Renderbuffers**
  - Use `glFramebufferInvalidate` instead of `glClear`

- **Vertex Array Objects (VAOs)**
  - Reduces CPU overhead in the driver

- **Pixel Buffer Objects**
  - Streaming texture update eliminates a copy

- **MapRangeBuffers**
  - Eliminates extra copies and saves bandwidth

- **Occlusion Queries**
  - Reduces CPU and GPU workloads
Profilers Tools

Grapher Mode: Real-time Analysis

Scrubber Mode: Detailed Frame Analysis

Frame rendered using GLES Emulation

Scruber Metrics

Detailed Frame Stats

GL ES Call Stack

Auto-Optimization Suggestions

Shader Editor

Shader Stats

Texture Browser

Metrics

Overrides
Conclusion

• Mobile GPUs are extremely capable today.
• Power and performance are intertwined.
• You now know some OpenGL ES tricks that may help reduce power usage.
• There are some great tools that can help find areas to reduce power usage.
More about OpenGL ES 3.0:

- OpenGL ES BOF 5-6pm Wed. August 8th
  - Adreno™ Profiler
  - OpenGL ES 3.0 Emulator and SDK
  - Qualcomm Developer Guide

Thank You!