UE4 Mobile Performance

Niklas Smedberg
Senior Engine Programmer, Epic Games
Content

• Part 1: Understanding mobile performance
  – Mobile GPU Hardware
  – Thermal limits
  – Performance guidelines

• Part 2: Adapt and conquer
  – Cross-platform profiling
  – Platform-specific profiling
  – Scaling your game based on device
Part 1: Understanding Mobile Performance

• Mobile hardware is evolving at a crazy rapid rate

• Next-generation mobile GPUs:
  – Fully featured (DirectX 11)
  – Peak performance comparable to Xbox 360 and PS3
    • 300+ GFLOPS and 26 GB/s
    – Able to run full UE4 desktop high-end rendering pipeline (e.g. NVIDIA K1)

• Phone users upgrade hardware very frequently
  – But tablet users don’t
  – Also, new large low-price markets are opening up
  – Result: Extremely wide performance range
Performance Trends (FP16 GFLOPS)

- 2010 SGX 535
- 2011 SGX 543MP2
- 2012 SGX 543MP3
- 2013 G6430
- 2014 Adreno, K1, GX6650

Graph showing performance trends from 2010 to 2014 with data points for each year:
- 2010: 6.4 GFLOPS
- 2011: 12.8 GFLOPS
- 2012: 25.5 GFLOPS
- 2013: 154 GFLOPS
- 2014: Above 300 GFLOPS
<table>
<thead>
<tr>
<th>Family</th>
<th>Old</th>
<th>Now</th>
<th>Soon</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Qualcomm Snapdragon Adreno</strong></td>
<td>Adreno 2xx</td>
<td>Adreno 3xx</td>
<td>Adreno 4xx</td>
</tr>
<tr>
<td><strong>ARM Mali</strong></td>
<td>400</td>
<td>T604, T628</td>
<td>T720, T760</td>
</tr>
<tr>
<td><strong>Imagination Technologies</strong></td>
<td>SGX 5xx</td>
<td>Series 6</td>
<td>Series 6XT</td>
</tr>
<tr>
<td><strong>NVIDIA Tegra</strong></td>
<td>Tegra 3, 4</td>
<td>K1</td>
<td>...</td>
</tr>
</tbody>
</table>
Tile-based Mobile GPU

• Mobile GPUs are usually tile-based (next-gen too)
  
  **Tile-based:** ImgTec, Qualcomm*, ARM
  
  **Direct:** NVIDIA, Intel, Qualcomm*, Vivante

* Qualcomm Adreno can render either tile-based or direct to frame buffer
  
  – Extension: GL_QCOM_binning_control
Tile-Based Mobile GPU

Summary:

- Split the screen into tiles
  - E.g. 32x32 pixels (ImgTec) or 300x300 (Qualcomm)
- The whole tile fits within GPU, on chip
- Process all drawcalls for one tile
  - Write out final tile results to RAM
- Repeat for each tile to fill the image in RAM
ImgTec Tile-based Rendering Process

Game → Cmd Buffer (RAM) → Vertex Processing → Tile Data (RAM)

Per Tile:

Hidden Surface Removal → Pixel Processing (Top-most only) → Tile Memory → Frame Buffer (RAM)
Framebuffer Resolve/Restore

- Expensive to switch Frame Buffer Object on Tile-based GPUs
  - Saves the current FBO to RAM
  - Reloads the new FBO from RAM

- Best performance:
  - A single render target for the entire frame
  - No post-processing passes

- Does not apply to NVIDIA Tegra GPUs!
  - This made it simpler for us to use our full desktop rendering pipeline on K1
  - “Rivalry” tech demo (showing 5:00 pm today)
Thermal Limits

• Hardware CPU and GPU clock frequencies change all the time!
  – Many times per milli-second!
  – To save battery
  – To prevent overheating

• Qualcomm Trepn Profiler
Thermal Limits

- Check your performance when device is cool
- Check again when it’s hot
- CPU uses much more power and heat than the GPU
  - Also, memory bandwidth generates a lot of heat
- Avoid unnecessary CPU usage
  - Spin-loops
  - Frequently waking up threads just to put them to sleep again
Performance Guidelines

- Always make sure lighting has been built before looking at performance
- Use as little post-process effects as you can get away with
- Make sure precomputed visibility has been set up properly
- Minimize overdraw (translucent or masked materials)
- Target 100-700 draw calls per frame
- Use as few texture lookups as possible in your materials
- Documentation:
  - https://docs.unrealengine.com/latest/INT/Platforms/Mobile/Performance/index.html
Performance Tier 1 – 2

1. **LDR (Low Dynamic Range)**
   - Fastest mode
   - Use when you don’t need lighting or post-process effects
   - Disable “Mobile HDR” in Rendering section in your Project Settings

2. **Basic Lighting**
   - Allows HDR lighting and some post-process effects
   - Use only static lights
   - Use only fully rough materials, not shiny (specular)
   - Disable Bloom and anti-aliasing
Performance Tier 3 – 4

3. Full HDR Lighting
   - High-quality lighting with best support for normal maps
   - Realistic specular reflections on surfaces with per-pixel roughness
   - Use only static lights
   - Bloom and anti-aliasing are recommended
   - Place reflection captures carefully for best results

4. Full HDR Lighting with per-pixel lighting from the Sun
   - Specify one directional light as stationary (the Sun)
   - All other lights are static
   - High-quality distance field shadows
Interlude: End of Part 1

Questions?

Keep going?

Coffee break?

Ready for more?
Part 2: Adapt and Conquer

- Very difficult to scale on CPU performance
  - Gameplay features can’t easily be switched off
  - Also, CPUs aren’t as different as GPUs are
  - Make sure you are never gamethread-bound on any device

- Scale your game purely based on GPU performance
  - Primarily resolution and post-process effects
  - Ship it!
Cross-platform Console Commands

- Common commands:
  - Stat Unit
  - Stat UnitGraph
  - Stat FPS
  - Stat SceneRendering
  - Stat Slow
  - ViewMode ShaderComplexity

- Documentation:
Console Command: Stat Unit

- Always the first step when checking performance
Console Command: Stat SceneRendering

• Shows Renderthread CPU performance and drawcalls
Console Command: ViewMode ShaderComplexity

- Visualize expensive materials in the PC ES2 previewer
- Shows approximate performance cost per material
- **Green** is good, **red** is bad. Pink or white is extremely expensive!
iOS Performance

- New Metal graphics API in iOS 8
  - Much faster on CPU
  - Up to 20x faster on renderthread
  - Allows for thousands of drawcalls on iOS devices with A7 processors

- Scale graphics quality based on exact device model
  - Still very few different device models, easy to target each one specifically
  - Resolution (MobileContentScaleFactor)
  - Post-process features
  - Etc...
Platform-Specific Profiling

• Each GPU family has their own profiling tools
  – Apple: Xcode GL Debugger (and Metal)
  – Qualcomm: Adreno Profiler
  – NVIDIA: Tegra Graphics Debugger
  – ImgTec: PVRTune, PVRTrace
  – ARM: Mali Graphics Debugger

• For CPU profiling
  – Apple: Instruments (Time Profiler)
  – NVIDIA: Tegra System Profiler
  – ARM: DS-5
iOS Performance Profiling

• Screenshot from Xcode, which shows:
  – How we clear FBO at the beginning of every render pass
  – Other important performance info
Qualcomm Adreno Profiler
NVIDIA Tegra Graphics Debugger
ImgTec PVRTune and PVRTrace
ARM Mali Graphics Debugger
Device Profiles

• UE4 selects one device profile at startup
  – Detects device model and capabilities

• Tweak each device profile for your game
  – Config/DefaultDeviceProfiles.ini
  – Each Device Profile can customize engine features, like:
    • +CVars=r.MobileContentScaleFactor=2
    • +CVars=r.BloomQuality=1
    • +CVars=r.DepthOfFieldQuality=1
    • +CVars=r.LightShaftQuality=1

• Documentation:
UE4 Mobile Performance Questions?

Documentation, Tutorials and Help at:
- AnswerHub: http://answers.unrealengine.com
- Engine Documentation: http://docs.unrealengine.com
- Official Forums: http://forums.unrealengine.com
- Community Wiki: http://wiki.unrealengine.com
- YouTube Videos: http://www.youtube.com/user/UnrealDevelopmentKit
- Community IRC: #unrealengine on FreeNode

Unreal Engine 4 Roadmap
- Imgty.com/?q=Unreal+engine+Trello+