The Agenda

- Introduction to tile based rendering
- Tiling is most common in mobile systems
- List of common tiling hardware features
- Resolves
  - Explanation of the different types
  - Optimizing code for resolves
- OpenGL ES 2.0 emulators
- Conclusion
The Agenda

- Introduction to tiling based rendering
- Tiling is most common in mobile systems
- List of common tiling hardware features
- Resolves
  - Explanation of the different types
  - Optimizing code for resolves
- OpenGL ES 2.0 emulators
- Conclusion
The Need in the Mobile Market

- Solution to the limited bandwidth problem
- Low power (better battery life)
- Small size (cheap)
- Good performance
- Flexible shaders
Traditional Graphics Pipeline vs TBR Pipeline

- TBR = Tile-Based Rendering
- Traditional GPUs render full scene in one pass
- Tiling GPUs render scene in multiple passes
Traditional Graphics Pipeline

Vertex Data ➔ VS ➔ Triangle Setup & Rasterization ➔ FS

WWW.GDCMOBILE.COM
TBR Graphics Pipeline

- Vertex Data
- Triangle Setup & Rasterization
- FS
- Tile 1
TBR Graphics Pipeline

Vertex Data → VS → Triangle Setup & Rasterization → FS → Tile 2
TBR Graphics Pipeline

Vertex Data → VS → Triangle Setup & Rasterization → FS → Tile 3
TBR Graphics Pipeline

Vertex Data → VS → Triangle Setup & Rasterization → FS → Tile 9
The Agenda

• Introduction to tiling based rendering
• Tiling is most common in mobile systems
• List of common tiling hardware features
• Resolves
  • Explanation of the different types
  • Optimizing code for resolves
• OpenGL ES 2.0 emulators
• Conclusion
Who Uses TBR?

- **Microsoft**
  - Talisman
- **Imagination Technologies**
  - KYRO and KYRO II (Desktop PC)
  - PowerVR CLX2 (Sega Dreamcast)
  - PowerVR MBX (OpenGL ES 1.x)
  - **PowerVR SGX (targets OpenGL ES 2.0)**
- **AMD**
  - Imageon 2380 (OpenGL ES 1.x)
  - Xenos (Xbox 360)
  - **Z430 and Z460 (targets OpenGL ES 2.0)**
Why is TBR so Popular in Embedded Devices?

- Reduced bus bandwidth
  - Saves power
  - Allows for simpler system designs
  - Desktop PC’s brute force approach doesn’t work as well in the mobile space
- Lower polygon counts in mobile games are an ideal match for TBR
The Agenda

• Introduction to tiling based rendering
• Tiling is most common in mobile systems
• List of common tiling hardware features
• Resolves
  • Explanation of the different types
  • Optimizing code for resolves
• OpenGL ES 2.0 emulators
• Conclusion
Fast Local Memory

- Tile-based GPUs have a small amount of fast memory on chip
  - Each tile gets rendered here then resolved to the final buffer in system memory
  - Very high bandwidth
  - Very low latency
  - Eliminates need for many caches and complex compression algorithms found on desktop GPUs
Geometry Binning

- Could just draw the scene to each tile and let the vertices get clipped, but...
- In the real world this costs too much vertex shader performance
- TBR hardware has ways of sorting triangles into bins for each tile
  - Each hardware vendor does this differently
- Don’t forget the driver/hardware has to batch up draw calls
Geometry Binning

Bin 1 / Tile 1

Bin 2 / Tile 2

Bin 3 / Tile 3

Bin 4 / Tile 4
External Bandwidth Usage Example

- Draw two triangles with depth testing
- Each triangle is 100 pixels and there is 50 pixels of overlap
- Each triangle has a single texture fetch
- Depth and color buffers are 32 bits
- Textures are 32 bits (easy math)
## External Bandwidth Usage Example

<table>
<thead>
<tr>
<th></th>
<th>Traditional Rendering</th>
<th>Tile Based Rendering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Texture Reads</td>
<td>150*4 bytes</td>
<td>150*4 bytes</td>
</tr>
<tr>
<td>Depth Reads</td>
<td>200*4 bytes</td>
<td>0 bytes</td>
</tr>
<tr>
<td>Depth Writes</td>
<td>150*4 bytes</td>
<td>0 bytes</td>
</tr>
<tr>
<td>Color Writes</td>
<td>150*4 bytes</td>
<td>0 bytes</td>
</tr>
<tr>
<td><strong>Total Bandwidth</strong></td>
<td><strong>2600 bytes</strong></td>
<td><strong>600 bytes</strong></td>
</tr>
</tbody>
</table>

This is just for the actual rendering TBR has constant bandwidth overhead from resolves.
The Agenda

• Introduction to tiling based rendering
• Tiling is most common in mobile systems
• List of common tiling hardware features
• Resolves
  • Explanation of the different types
  • Optimizing code for resolves
• OpenGL ES 2.0 emulators
• Conclusion
Resolves

- Resolves are the copies between a GPU’s fast internal memory and the system’s slow external memory
- Light weight resolve
  - Copies from fast internal memory to slow external memory
- Heavy weight resolve
  - Restores a tile with a copy from slow external memory to fast internal memory, and then does a light weight resolve
Heavy Weight Resolves

** This is repeated for each tile **

GPU

- Fast Internal Memory (Tiles Rendered Here)

Heavy Resolve

Light Resolve

Slow External Memory
Resolve Cases

- `eglSwapBuffers`
  - Light weight resolve

- `glBindFramebuffer`
  - Light weight resolve if you start drawing with `glClear`
  - Heavy weight resolve if you don’t

- `glTexImage2D`, `glTexSubImage2D`, `glBufferData`, and `glBufferSubData`
  - Drivers should prevent the resolve
  - Mid-frame calls force driver to create an extra copy of the data
  - Starting a frame with these calls prevents that extra copy
Resolve Cases

- glCopyTexImage2D and glCopyTexSubImage2D
  - Heavy weight resolve
  - OpenGL ES 2.0 has FBOs so use them
- glReadPixels
  - Heavy weight resolve
  - Please don’t use this (especially in the middle of a frame)
- Exceeding triangle or state buffer limits
  - Heavy weight resolve (driver and hardware specific)
// Put all glTexImage2D, glTexSubImage2D, glBufferData and
// glBufferSubData commands here

// FBO block (optional - only needed if you are using FBOs)
for (int i = 0; i < numFbos; ++i)
{
    glBindFramebuffer( target[i], framebuffer[i] );
    glClear( GL_COLOR_BUFFER_BIT |
        GL_DEPTH_BUFFER_BIT |
        GL_STENCIL_BUFFER_BIT );
    Draw your scene for each FBO here
}

// If needed put a glClear here (color, depth, stencil)
Draw your scene to your backbuffer here

// If you absolutely need a glReadPixels do it here
eglSwapBuffers( dsp, backbuffer );

...Repeat...
A frame in the life of TBR

- State
  - `glDrawElements()`
- State
  - `glDrawElements()`
- State
  - `glDrawElements()`
- State
  - `glDrawElements()`

Sort Triangles Into Bins

- Bin 1
- Bin 2

SwapBuffer
A frame in the life of TBR

1. Triangles in Bin1
2. Vertex Shader
3. Rasterization
4. Fragment Shader

Fast Local Memory

Slow External Memory

Swap Triggers a Resolve

Lots of memory here

WWW.GDCMOBILE.COM
A frame in the life of TBR

Triangles in Bin2 → Vertex Shader → Rasterization → Fragment Shader

Fast Local Memory

Swap Triggers a Resolve

Slow External Memory

Lots of memory here
A frame in the life of TBR

Slow External Memory

Lots of memory here

LCD controller displays backbuffer to screen
The Agenda

- Introduction to tiling based rendering
- Tiling is most common in mobile systems
- List of common tiling hardware features
- Resolves
  - Explanation of the different types
  - Optimizing code for resolves
- OpenGL ES 2.0 emulators
- Conclusion
OpenGL ES 2.0 Emulators

- Develop OpenGL ES 2.0 code before hardware is available
- Use the Visual Studio’s build environment
- Graphics code should require no changes when porting to real hardware
- Help track down TBR performance bottlenecks
OpenGL ES 2.0 Emulators

- Two OpenGL ES 2.0 emulators are:
  - AMD
    - Future Releases will have
      - Performance throttling to simulate real hardware
      - Detailed performance stats to help find bottlenecks
  - Imagination
    - [http://www.imgtec.com/PowerVR/insider/toolsSDKs/KhronosOpenGLES2xSGX](http://www.imgtec.com/PowerVR/insider/toolsSDKs/KhronosOpenGLES2xSGX)
- Demo
The Agenda

• Introduction to tiling based rendering
• Tiling is most common in mobile systems
• List of common tiling hardware features
• Resolves
  • Explanation of the different types
  • Optimizing code for resolves
• OpenGL ES 2.0 emulators
• Conclusion
TBR Strengths

• In general TBR excels at bandwidth limited operations
  • Blending
  • Overdraw
  • Multisample AA
The Take Home Message

• Hopefully you now know what TBR is
• Avoid extra resolves
  • Costs power and reduce battery life
  • Costs performance
• Optimizing for TBR usually only takes a few small changes
• OpenGL ES 2.0 Emulators are a good way to start writing efficient code
Questions?
Maurice.Ribble@amd.com
devrel@amd.com