Demand Loaded Textures in OptiX [15 mins]

Siggraph 2019
Motivation

Support 1 TB+ of textures in OptiX

❖ CPU film renderers:
   ➢ Load from HUGE datasets
   ➢ Synchronous loads, cached (ok to pause 1 thread)

❖ GPU renderers:
   ➢ Synchronous loads ruin SIMD concurrency (not ok to pause launch)
   ➢ Pre-load texture data
   ➢ No net-disk-ram cache
   ➢ Memory limit up to 48 GB / GPU

❖ Pre-loading wastes memory and bandwidth
❖ ⇒ GOAL: preserve parallelism and hardware filtering
“Cooperative Paging”

- Re-try strategy: fail gracefully & repeat until done
- *Cooperative*: kernel issues requests, then yields
- Launch* is the sync point

- If texture data is not resident when requesting it:
  - Request is recorded
  - Shader provides fallback behavior
- Upon launch completion, load requested data
- Re-launch kernel
Cooperative Paging

synchronize on launch

- Initialize on-demand loading
- Pre-launch: prepare for launch
- Mid-launch: requesting textures (in closest hit shader...)
- Post-launch: process requests
- Re-launch
Use Cases

➔ Single frame render:
  ◆ Multiple launches
  ◆ Can dovetail nicely with adaptive sampling

➔ Interactive render:
  ◆ Temporary or low-resolution assets initially
  ◆ Lazy-load to final quality
  ◆ Fast startup time, no loss of interactivity
Pre-launch: copy loaded textures

on the host

- Copy any currently loaded textures to the device

- 1st time:
  - Do nothing (fast start / no knowledge)
  - OR Pre-load some textures (good guess / some knowledge)

- next time(s): fill new requests from previous launch
Mid-launch: request textures

on the device (e.g., closest hit shader)

- Calculate LOD (using gradients / ray differentials)
- Request texture data
- YES resident: sample texture immediately
- NOT resident: fall back
  - Shader can throw a user exception: terminate the ray (but not the launch)
  - OR, substitute a default color and continue (e.g. progressive refinement)
MIP levels vs. tiles

Allocation vs. filling

❖ Remember the goal: preserve hardware filtering

❖ Sparse textures (“tiled resources” in DX) are in hardware…
  ➢ These are being exposed in CUDA soon

❖ Interim approach: allocate & fill entire MIP levels
  ➢ One contiguous allocation (spanning required MIP levels)
  ➢ Reallocate backing storage (as new MIP levels are requested)
  ➢ Fully hardware accelerated sampling & filtering
Post-launch: process requests

on the host

- Retrieve list of requests
- Determine min & max MIP levels per texture
- Reallocate textures
- Fill requested texture data on host
Re-launch: after processing requests

50 goto 20

- Go back to pre-launch
  - send new texture data to GPU & try again
Initial results courtesy AutoDesk
## Initial results courtesy AutoDesk

<table>
<thead>
<tr>
<th></th>
<th>Pre-load textures</th>
<th>On-demand textures</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Near cam</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time to first pixel (mm:ss)</td>
<td>1:57</td>
<td>0:14</td>
</tr>
<tr>
<td>Time to completion (mm:ss)</td>
<td>2:29</td>
<td>0:59</td>
</tr>
<tr>
<td>GPU memory usage (MiB)</td>
<td>17040</td>
<td>12878</td>
</tr>
<tr>
<td><strong>Far Cam</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time to first pixel (mm:ss)</td>
<td>1:57</td>
<td>0:12</td>
</tr>
<tr>
<td>Time to completion (mm:ss)</td>
<td>2:03</td>
<td>0:33</td>
</tr>
<tr>
<td>GPU memory usage (MiB)</td>
<td>17040</td>
<td>6386</td>
</tr>
</tbody>
</table>
Summary

Demand loading allows the GPU to load only what’s needed, saving time & memory

- Currently supports filling whole MIP levels
- BONUS: This works on non-texture buffers too!

- Cooperative Paging:
  - Eliminates synchronization overhead
  - Gives the user full control over application behavior
  - Choose your own fall back strategy
OptiX Demand Texture Code

OptiX 6: Demand Texturing available via OptiX API
   rtBufferCreateFromCallback()
   rtTex2DLoadOrRequest()

OptiX 7: Demand Texturing library is open-source, client side code
   Included with the OptiX SDK
   Explicit, user controlled, fewer management layers
   Example of best practices: you take the lib, fork it, modify to suit
Demand-load texture sample for OptiX 7

**optixDemandTexture**

- Encapsulates low-level OptiX paging routines.
  - All user-level code. Not part of OptiX 7 API.

- Focus on simplicity
  - Synchronous, Single GPU, No eviction

- Advanced sample demonstrates multi-GPU, etc.
  - Good starting point for a production-ready GPU texture system.
Future work

➔ Per-tile filling (for now, allocating whole MIP levels)

➔ Support CUDA sparse textures (once available)

➔ Support eviction (LRU MIP levels / tiles)
Questions?

Contributors

Mark Leone
Richard Thomson
Steve Parker
Kincaid Savoie
David Cline
Tim Grant
Daniel Rushton
Kyle Pierson