Mesh shading best practices
VK_EXT_mesh_shader

Timur Kristóf
Table of Contents

1. Mesh shading intro
   ■ Programming model
   ■ Overview of a pipeline

2. Typical uses of mesh shading
   ■ Basic use, meshlets

3. Comparison to traditional pipeline
   ■ Pros and cons
   ■ MS vs. VS, tess, GS

4. Demo
Mesh shading

The good
New programming model that enables efficient geometry processing for highly detailed scenes.

The bad
May be difficult to integrate and achieve better perf than the traditional pipeline.

The ugly
API is very low-level and vendor-specific tweaks are necessary for optimum performance.
Mesh shading programming model

- Compute-like
- Creates vertices and primitives
- Eliminates fixed-function bottlenecks (IA, tess.)
- Very low level
Mesh shading programming model

- Not (yet?) suitable for tiling GPUs
Overview of a mesh shading pipeline
Mesh shading pipeline (not recommended)

DrawMeshTasksEXT

(X, Y, Z)

Mesh shader
(workgroups)

rasterization

Fragment shader
Mesh shading pipeline

- **DrawMeshTasksEXT**
  - \((X, Y, Z)\)

- **Task shader**
  - (workgroups)

- **Mesh shader**
  - (workgroups)

- **Payload**

- **Rasterization**

- **Fragment shader**
New shader stages

Task shader
How many MS workgroups to launch?
Optional "payload" output.

Mesh shader
Compute-like programming model to feed the rasterizer directly.
Typical uses of mesh shading
Typical uses of mesh shading

Meshlets
During asset building, split your geometry into a smaller cluster of primitives: "meshlets".

Procedural geometry
Generate geometry on the fly according to a mathematical formula without loading any data from memory.
What is a meshlet?

Subdivide your meshes into small groups of vertices/primitives during the asset building phase.

- Up to 256 vertices/primitives (typical: 128/128)
- Typically, 1 MS workgroup processes 1 meshlet
- 1 TS invocation ~ 1 meshlet
What is a meshlet?
Dispatching workgroups of Task / Mesh

- vkCmdDrawMeshTasksEXT
- vkCmdDrawMeshTasksIndirectEXT
- vkCmdDrawMeshTasksIndirectCountEXT
Task shader execution

- 1 invocation \(\sim\) 1 meshlet
- Use EmitMeshTasksEXT to launch mesh workgroups
Mesh shader execution

- 1 invocation \(\sim\) 1-2 vertex/primitive
- Use `SetMeshOutputsEXT` to allocate output arrays
- Write to output arrays
- Follow driver preferences!
Mesh shader driver preferences

- Compact vertex / primitive output
- Use local invocation index to address output arrays
- Recommended task / mesh workgroup sizes and limits
Mesh shader driver preferences

- NVidia: small workgroups, more vertices/primitives per invocation
- AMD: large workgroups, 1 vertex/primitive per invocation
- Use a compile-time loop to match both!
Mesh shader execution (incl. driver preferences)

- Per-vertex / per-primitive pre-processing
- Per-primitive culling (optional)
- Compaction (if necessary)
- SetMeshOutputsEXT
- Per-vertex / per-primitive processing
- Write output arrays
What can you do in a task shader?

- Coarse per-meshlet culling
- LOD selection
- Geometry amplification
- Replacement for compute pre-pass
What else can you do in a mesh shader?

- Per-triangle culling
- Procedural generation of vertices and primitives
Per-meshlet, per-vertex/primitive processing

1 dispatch ~ 1 mesh (all meshlets)
1 workgroup ~ group of meshlets
1 invocation ~ 1 meshlet (typical)

per-meshlet processing

per-vertex/per-primitive processing

1 dispatch ~ group of meshlets
1 workgroup ~ 1 meshlet
1 invocation ~ 1/2 vertices/primitives
What SHOULD you do?

• Use task shaders for per-meshlet culling and LOD selection
• Follow driver preferences in your mesh shaders (compile-time loop)
What more CAN you do?

• Easy to fallback to vertex shaders on older HW
• Replace ANY use of GS with MS
• Investigate meshlet building algorithms to optimize per-meshlet culling
• If you still have very high geometric complexity, consider per-primitive culling
What SHOULDN’T you do?

• DON’T do per-meshlet culling in mesh shaders
  *Please use task shaders!*

• DON’T overuse task payload
  *NOT meant for geometry data!*

• DON’T abuse MS for generic compute work
  *Please look into DGC.*
What else SHOULDN’T you do?

• DON’T disregard vendor preferences
• DON’T mix TS+MS with compute culling
If you disregard best practices...

a driver-optimized VS is going to perform better than your MS.
If you disregard best practices...

then you are going to write blog posts about how mesh shading is "bad".
Comparison to traditional pipeline
Mesh shading pros

- Avoids fixed-func bottlenecks (IA, tess)
- More flexible geometry amplification
- Includes compute pre-pass in GFX pipeline
- Can pre-compute and discard geometry early (per-meshlet culling)
- Can save bandwidth and computations on invisible primitives
Mesh shading cons

- Not useful (yet?) on mobile GPUs / tilers
- Big foot gun (like any low-level tool)
- Coupling between input data format and shader code
- Vendor specific perf preferences
Vertex shader vs. Mesh shader

- VS: output topology same as input
  MS: custom, explicit output topology
- MS is closer to how the HW works
  (workgroups)
- VS inputs are annoying
- Task shaders are more efficient than a compute pre-pass for eg. coarse culling, etc.
  (MS and TS execution can overlap)
Geometry shader vs. Task + Mesh shader

- Most GS implementations are inefficient due to the restrictive programming model of 1 invocation ~ 1 input prim
- MS gives you an opportunity to better utilize the HW
- MS not restricted to strip primitives
- MS allows per-primitive outputs
- TS is better for more serious amplification
Tessellation vs. Task + Mesh shader

- TS can use your own formula instead of what the fixed-func HW can give you
- TS creates integer number of MS workgroups instead of dealing with floating point tess factors
- No need to deal with patch primitives
- MS I/O is easier, can pass per-primitive outputs to FS
Demo
Mesh shading demo

NVidia CAD scene demo
The scene contains nine cars, but the camera focuses on a single one, most others are fully outside frustum. The total scene has 32M triangles and 16K draw calls.
Mesh shading best practices
VK_EXT_mesh_shader

Timur Kristóf