Tessellation in Call of Duty: Ghosts

Wade Brainerd
Activision
May 1978
My birth

November 1978
E. Catmull & J. Clark:
Recursively generated B-spline surfaces on arbitrary topological meshes
Computer-Aided Design (6): 350-355
Compute data flow
Compute data flow
Quad Edge Mesh

```
struct HalfEdge
{
    HalfEdge  *opposite;
    HalfEdge  *facePrev;
    HalfEdge  *faceNext;
    HalfEdge  *vertPrev;
    HalfEdge  *vertNext;
    int face;
    int vert0;
    int vert1;
};
```
Quad Edge Mesh

Sample implementation in JavaScript + WebGL

http://wadeb.com/subd
E. Catmull & J. Clark:
Recursively generated B-spline surfaces on arbitrary topological meshes
Computer-Aided Design 10(6): 350-355
Regular patch
Tessellation data flow

Patch index buffer
- 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 ...

Vertex buffer

- Vertex shader: Loads control points
- Hull shader: Tweaks control points, calculates tess factors
- Tessellator
- Domain shader: Makes final vertices
- Pixel shader

MathBox created by Steven Wittens
http://acko.net/
Feature Adaptive Subdivision

8.1 Comparison to Global Mesh Refinement

Table I. Timing using the Big Guy model for our scheme (feature adaptive patching) compared against our global table driven subdivision method and the previously published GPU subdivision algorithm by Shiue et al.. Note that all timings include final rendering, while we additionally break out draw time for our global subdivision scheme.

<table>
<thead>
<tr>
<th>Subdivision Level</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature Adaptive Patching</td>
<td>0.10</td>
<td>0.20</td>
<td>0.34</td>
<td>0.81</td>
<td>2.30</td>
</tr>
<tr>
<td>Shiue Subdivision</td>
<td>0.62</td>
<td>7.26</td>
<td>13.97</td>
<td>21.42</td>
<td>34.93</td>
</tr>
<tr>
<td>Global Table Subdivision</td>
<td>0.06</td>
<td>0.18</td>
<td>0.79</td>
<td>3.07</td>
<td>12.05</td>
</tr>
<tr>
<td>Draw Time (Table Subd.)</td>
<td>0.04</td>
<td>0.06</td>
<td>0.37</td>
<td>1.45</td>
<td>5.78</td>
</tr>
</tbody>
</table>
Feature Adaptive Subdivision

Nießner's implementation


OpenSubdiv

https://github.com/PixarAnimationStudios/OpenSubdiv
Implementation steps

**Offline Global Subdivision**
- In the model converter
- Rendered as triangles
- Test DCC parity

**Runtime Global Subdivision**
- Model converter calculates influence tables
- Vertices evaluated by compute shaders
- Still rendered as triangles

**Feature Adaptive Subdivision**
- Regular faces rendered by tessellator
- Irregular faces rendered as triangles w/compute
Screen Space Adaptive

\[ \frac{(L_{u=0} + L_{u=1})}{2} \]

\( L \) = projected limit surface

\( d \) = curvature metric
Edge Extrapolation

Hull shader
Tweaks control points

\[ c_0 = 2c_4 - c_8 \]
\[ c_1 = 2c_5 - c_9 \]
\[ c_2 = 2c_6 - c_{10} \]
\[ c_3 = 2c_7 - c_{11} \]
Corner Extrapolation

Hull shader
Tweaks control points

\[
\begin{align*}
  c_3 &= 4c_6 - 2c_5 - 2c_{10} + c_9 \\
  c_0 &= 2c_4 - c_8 \\
  c_1 &= 2c_5 - c_9 \\
  c_7 &= 2c_6 - c_5 \\
  c_11 &= 2c_{10} - c_9 \\
  c_2 &= 2c_6 - c_{10} \\
  c_15 &= 2c_{14} - c_{13}
\end{align*}
\]
Cache the compute shader output

*if it's not animated*
HS thread flow

<table>
<thead>
<tr>
<th>Thread 0</th>
<th>Control point 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread 1</td>
<td>Control point 1</td>
</tr>
<tr>
<td>Thread 2</td>
<td>Control point 2</td>
</tr>
<tr>
<td>Thread 3</td>
<td>Control point 3</td>
</tr>
<tr>
<td>Thread 4</td>
<td>Control point 4</td>
</tr>
<tr>
<td>Thread 5</td>
<td>Control point 5</td>
</tr>
<tr>
<td>Thread 6</td>
<td>Control point 6</td>
</tr>
<tr>
<td>Thread 7</td>
<td>Control point 7</td>
</tr>
<tr>
<td>Thread 8</td>
<td>Control point 8</td>
</tr>
<tr>
<td>Thread 9</td>
<td>Control point 9</td>
</tr>
<tr>
<td>Thread 10</td>
<td>Control point 10</td>
</tr>
<tr>
<td>Thread 11</td>
<td>Control point 11</td>
</tr>
<tr>
<td>Thread 12</td>
<td>Control point 12</td>
</tr>
<tr>
<td>Thread 13</td>
<td>Control point 13</td>
</tr>
<tr>
<td>Thread 14</td>
<td>Control point 14</td>
</tr>
<tr>
<td>Thread 15</td>
<td>Control point 15</td>
</tr>
</tbody>
</table>

Tessellation factors
- Transform 8 control points to screen space
- Adaptive tessellation metric
- View frustum culling
- Backface culling

Idle
CS as HS

Compute shader
- Loads control points
- Culls patches
- Calculates tess factors
- Stores tess factors

Tessellation factors buffer
- patch0: 4 4 2 2 2
- patch1: 6 4 4 4 4
- patch2: 0 0 0 0 0 [culled]
- patch3: 1 1 1 1 1
- patch4: 8 6 6 4 2 2
...

Hull shader
- Tweaks control points
- Loads tess factors

Tessellator

Domain shader
- Makes final vertices

Shadow shader

Domain shader
- Makes final vertices

Lit shader
CS thread flow

Thread 0 → Patch 0 tessellation factors
Thread 1 → Patch 1 tessellation factors
Thread 2 → Patch 2 tessellation factors
Thread 3 → Patch 3 tessellation factors
Thread 4 → Patch 4 tessellation factors
Thread 5 → Patch 5 tessellation factors
Thread 6 → Patch 6 tessellation factors
Thread 7 → Patch 7 tessellation factors
Thread 8 → Patch 8 tessellation factors
Thread 9 → Patch 9 tessellation factors
Thread 10 → Patch 10 tessellation factors
Thread 11 → Patch 11 tessellation factors
Thread 12 → Patch 12 tessellation factors
Thread 13 → Patch 13 tessellation factors
Thread 14 → Patch 14 tessellation factors
Thread 15 → Patch 15 tessellation factors

Tessellation factors buffer
patch0: 4 4 2 2 2 2
patch1: 6 4 4 4 4 4
patch2: 0 0 0 0 0 [culled]
patch3: 1 1 1 1 1
patch4: 8 6 6 4 2 2
...
Analysis: Depth Only
Bottleneck mitigation
Low utilization
DS waves stall at first export

Parameter cache?
HLSL

Vertex Shader
8 VGPR 16 SGPR

Hull Shader
24 VGPR 32 SGPR

Domain Shader
60 VGPR 40 SGPR

Assembly

Vertex Shader
7 VGPR 8 SGPR

Hull Shader
24 VGPR 40 SGPR

Domain Shader
24 VGPR 40 SGPR
HLSL
Domain Shader
60 VGPR

Assembly
Domain Shader
24 VGPR

GCN VGPR Count
<=24 28 32 36 40 48 64 84 <= 128 > 128
Max Waves/SIMD
10 😊 9 8 7 6 5 4 3 2 😞 1 😞

4 waves per SIMD

10 waves per SIMD

table by Layla Mah, AMD
Wave launch rate

**Shader Engine**

VS/HS/DS wave rate is 1 CP / clock  
36 SIMDs @ 64 clocks / wave

64 clocks / wave  
2304 clocks latency

Our waves run in < 2000 clocks

*Occupancy limit is 1*
Limit to fewer CUs
Fill the rest of the GPU with async compute
Mix w/async compute
Let the GPU load balancer schedule everything
Async compute
Skinning
Tension mapping
Blend shapes
Hull shaders
Ambient occlusion
Depth decompress
Post FX?
Displacement

bias 0 scale
Parallax mapping

1. sample height in PS
2. project eye ray to extruded plane, adjust uv
3. sample normal, color, spec
Smooth normal → Phong smoothing

Hard normal → Gap fill
Distance fade

- full tessellation
- lerp out
- no tessellation
Evac to the river for exfil.

Stream
Thanks

Paul Edelstein
Andrew Aye
Michael Vance
Matthias Niessner
Jens Raab
Kefei Lei
Maceij Kot
Felipe Gomez-Frittelli

Jake Rowell
John Dobbie
Riccard Linde