Coarse Pixel Shading

Karthik Vaidyanathan, Marco Salvi, Robert Toth, Tim Foley, Tomas Akenine-Möller, Jim Nilsson, Jacob Munkberg, Jon Hasselgren, Masamichi Sugihara, Petrik Clarberg, Tomasz Janczak, and Aaron Lefohn
Motivation

27”
2560 X 1440 Pixels

6”
2560 X 1440 Pixels
Shade Less Than Once Per Pixel

• High Density Display
• Diffuse Indirect Illumination
• Low Detail Materials
• Motion and Defocus Blurred Regions
• Foveated Rendering
• ...
New Capabilities

• Programmable control of shading rates
  • e.g. shade once every 2x2 pixels

• Different components at different rates
  • Multi-rate shading
  • e.g. Ambient occlusion at a reduced rate, other terms at pixel rate
Coarse Pixels

• Typically
  • 1 visibility sample, and 1 shading sample per pixel

• CPS
  • 1 visibility sample per pixel, but 1 shading sample per $N \times M$ pixels (coarse pixel)
**APIs**

1. Constant shading rate selected via render state
2. Interpolated from vertex shader outputs
3. Radial function of screen coordinates
Programming - Example

```
struct VS_OUT {
    float2 cpsize : SV_CoarsePixelSize;
    ...
};

VS_OUT VertexShader(VS_IN In) {
    float2 cpsize = In.pos.z * ...

    VS_OUT Out;
    Out.cpsize = cpsize;
    return out;
}
```
Pipeline Changes
Pipeline Operation

Coarse Quad → Rasterizer → Pixel Shader → Output Merge
Multi Rate Shading
Multi-Rate Shading
Multi-Rate Shading

Rasterizer → Coarse Pixel Shader → Pixel Shader → Sample Shader → Output Merge

Interpolated Attributes
Example Usage

```c
float4 CoarseShader(VS_OUT In)
{
    float4 res = ComputeDiffuse(In)
    return res;
}
```

```c
float4 PixelShader(VS_OUT In, float4 CPIn)
{
    float4 specular = ComputeSpecular(In)
    return (specular*CPIn);
}
```
Scheduling

• All phases scheduled on the same thread

• Execute coarse phase then loop over pixels/samples

• Inter-phase data in registers
Scheduling

SIMD Lanes

Coarse Phase
Scheduling

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>7</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>13</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>15</td>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>
Results
Applications

High DPI  50 %
Peripheral CPS  33 %
Defocus Blur  19 %
Multi-Rate AO  63 %
Color Compression

The graph illustrates the color bandwidth in MB across different frames. The x-axis represents the frame number, ranging from 0 to 300, while the y-axis indicates the color bandwidth in MB, ranging from 300 to 600.

Four different color schemes are compared:
- **Uncompressed**: The blue line represents the uncompressed data, showing a consistent increase in bandwidth over time.
- **No CPS**: The orange line indicates the data with no compression, which also shows a steady increase in bandwidth.
- **Peripheral CPS**: The blue line with purple dots represents data compressed with peripheral CPS, showing fluctuations similar to the uncompressed data but at a lower bandwidth.
- **CPS**: The purple line indicates data compressed with CPS, showing the lowest bandwidth among the four, with slight fluctuations.

The graph demonstrates the effectiveness of compression techniques in reducing color bandwidth, with CPS providing the most significant reduction compared to the uncompressed and peripheral CPS methods.
Acknowledgements

• Tom Piazza
• David Blythe
• Charles Lingle
• Prasoonkumar Surti
• Uzi Sarel
• Bar On Tomer

We’re hiring in San Francisco!