Texture Mapping

Model Space

Texture Space

mapping
Texture Mapping

Authoring Problems:
× Labor intensive
× No local resolution readjustment
× No model editing
× Wasted space

Rendering Problems:
× Seam artifacts!
× Incorrect mip-maps
× Incorrect anisotropic filtering
Texture Mapping

- Bilinear filtering errors near seams

High-resolution texture

[Lefebvre et al. 2005]
Texture Mapping

- Errors can often be hidden

High-resolution texture

[Lefebvre et al. 2005]
• Errors show up in mip-map levels
Texture Mapping

- Errors cause cracks with displacement
SIGGRAPH 2017 Course: Rethinking Texture Mapping
Texture Mapping

model

mapping

texture
Mesh Colors
Mesh Colors

- $R = 1$
- Vertex Colors
Mesh Colors

- $R = 2$
- Edge Colors
Mesh Colors

- $R = 2$
- Edge Colors
Mesh Colors

- $R = 4$
- Face Colors
Mesh Colors

- $R = 4$
- Face Colors
Mesh Colors

- $R = 8$
- Color positions from indices
Mesh Colors

- Colors are shared along edges
  - Guaranteed continuity
Mesh Colors

- Model editing after texture painting
Mesh Colors

- Non-uniform face resolutions
Mesh Colors

• Non-triangular Meshes
  – Quadrilaterals
    • Triangle pair
    • Quadrilateral positioning
  – NURBS
  – Subdivision surfaces
    • Dividing faces only
Mesh Colors

Render Mesh

Canvas Mesh
Mesh Colors vs Ptex

Per-face Texture
All pixels are inside the face.

Mesh Colors
Pixels on the edges and vertices are shared.
Mesh Colors vs Ptex

**Per-face Texture**
All pixels are inside the face.

**Mesh Colors**
Pixels on the edges and vertices are shared.

Must access mesh topology in shader
Mesh Colors vs 2D Texture

2D Texture
3 MB

Mesh Colors
(converted from 2D texture)
2.4 MB
Mesh Colors

Authoring:
- ✓ No mapping
- ✓ Local resolution readjustment
- ✓ Model editing after painting

Rendering:
- ✓ No seams!
- ✓ Correct mip-maps
- ✓ Correct anisotropic filtering

+ Similar memory use as 2D textures
Mesh Colors

<table>
<thead>
<tr>
<th></th>
<th>3 K</th>
<th>50 K</th>
<th>218 K</th>
</tr>
</thead>
<tbody>
<tr>
<td>face count</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>color count</td>
<td>530 K</td>
<td>530 K</td>
<td>9 000 K</td>
</tr>
<tr>
<td>2D texture</td>
<td>3938 fps</td>
<td>2597 fps</td>
<td>337 fps</td>
</tr>
<tr>
<td>Nearest</td>
<td>2567 fps</td>
<td>1147 fps</td>
<td>273 fps</td>
</tr>
<tr>
<td>Bilinear</td>
<td>2076 fps</td>
<td>862 fps</td>
<td>247 fps</td>
</tr>
<tr>
<td>MIP-map</td>
<td>991 fps</td>
<td>376 fps</td>
<td>180 fps</td>
</tr>
<tr>
<td>Anisotropic</td>
<td>452 fps</td>
<td>152 fps</td>
<td>109 fps</td>
</tr>
</tbody>
</table>

Software Texture Filtering is SLOW!

[Yuksel et al. 2010]
Mesh Color Textures
Mesh Color Textures

• Key idea
  – Convert mesh colors to 2D textures
Mesh Color Textures

• Key idea
  – Convert mesh colors to 2D textures
Mesh Color Textures

• Key idea
  – Convert mesh colors to 2D textures
Mesh Color Textures

• Convert mesh colors to 2D textures
  – Duplicate vertex and edge colors
Mesh Color Textures

- Convert mesh colors to 2D textures
  - Interpolated Colors

\[ c_3 = c_1 + c_2 - c_0 \]
Mesh Color Textures

• Minimize duplicated colors
  – Place neighboring faces together
Mesh Color Textures

• Nonuniform Mesh Colors
  – Use two resolutions per face
Mesh Color Textures

- Binilear filtering
  - Simple
- Trilinear filtering (mip-maps)
  - Different texture coordinates per level
Mesh Color Textures

- Trilinear filtering (mip-maps)
  - 4D texture coordinates
    \[ u_\ell = u_s / 2^\ell + u_\delta \]
    
    - Packing problem
    - Intuitively,
      
      - \( u_s \): # of preceding edge colors (+1 per face)
      - \( u_\delta \): # of preceding faces + offsets
Mesh Color Textures

• See paper for
  – A packing algorithm for defining 4D coordinates
  – An optimization scheme for mip-map levels beyond vertex colors
Fragment Shader for Mip-map Filtering

```glsl
#version 420 core
layout(location = 0, index = 0) out vec4 color;

in vec4 vs_texCoord; // interpolated 4D texture coordinate

uniform int levelCount; // number of mip-map levels
uniform sampler2DRect level[12]; // mip-map levels

vec4 texture4d(sampler2DRect tex, vec4 tc, float scale)
{
    vec2 t = tc.xy / scale + tc.zw; // 2D texture coordinate from 4D
    return texture(tex, t); // texture lookup
}

void main()
{
    vec2 dtdx = dFdx(vs_texCoord.xy); // screen-space x derivative of scalable texture coordinate
    vec2 dtdy = dFdy(vs_texCoord.xy); // screen-space y derivative of scalable texture coordinate
    float dtdx_len2 = dot(dtdx, dtdx); // squared length of the screen-space x derivative
    float dtdy_len2 = dot(dtdy, dtdy); // squared length of the screen-space y derivative
    float len2 = max(dtdx_len2, dtdy_len2); // squared length of the maximum derivative

    float level = max(1, log2(len2)*0.5); // desired mip-map level
    int lev = int(level); // first mip-map level
    float f = level - lev; // trilinear interpolation weight

    int lev1 = clamp(lev-1, 0, levelCount-1); // first mip-map level
    int lev2 = clamp(lev, 0, levelCount-1); // second mip-map level

    vec4 c1 = texture4d(level[lev1], vs_texCoord, 1 << lev1); // 2D texture lookup with 4D coordinates
    vec4 c2 = texture4d(level[lev2], vs_texCoord, 1 << lev2); // 2D texture lookup with 4D coordinates

    color = c1 * (1-f) + c2 * f; // final interpolated color
}
```
Mesh Color Textures

• Avrg. Overhead – over standard 2D textures

NVIDIA GeForce GTX 1080
NVIDIA GeForce GTX 1070
NVIDIA GeForce GTX TITAN X
NVIDIA GeForce GTX 980
NVIDIA GeForce GTX TITAN
NVIDIA GeForce GTX 750 Ti

overhead in milliseconds
Mesh Color Textures

- Wasted texture space due to packing

<table>
<thead>
<tr>
<th></th>
<th>Standard 2D texture</th>
<th>Mesh Color Texture</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>26%</td>
<td>15%</td>
</tr>
<tr>
<td></td>
<td>39%</td>
<td>15%</td>
</tr>
<tr>
<td></td>
<td>19%</td>
<td>15%</td>
</tr>
<tr>
<td></td>
<td>17%</td>
<td>16%</td>
</tr>
</tbody>
</table>
Mesh Color Textures

• Limitations
  – Shader complexity
    • Can be fixed via graphics API changes
  – Anisotropic filtering leads to seams
    • Possible to fix, but requires hardware modification
  – Additional storage
    • 4D texture coordinates, instead of 2D
Mesh Color Textures bring the advantages of Mesh Colors to real-time rendering.

Conclusion

- Advantages of Mesh Colors
  - Improved authoring
  - Improved rendering

- Speed of 2D textures
Acknowledgements

- Christer Sveen
- Murat Afsar
- Lee Perry-Smith
- Paul Tosca
- Anonymous reviewers
- NSF Award #1409129 “Architectures for Energy Efficient Ray Tracing”
Pipeline

• Standard 2D textures
  - Modelung
  - Mapping
  - Painting
  - Rendering
  - costly iterations
  - seams!

• Mesh Color Textures
  - Modelung
  - Painting
  - Conversion
  - Rendering
  - easy iterations
  - no seams

  3D painting with mesh colors
  mesh colors to mesh color texture
**Pipeline**

- **Hybrid** *(not ideal, but easier to implement)*

  ![Diagram](image)

  - **Modeling**
  - **Mapping**
  - **Painting**
  - **Conversion**
  - **Rendering**

  - **not-as-costly iterations**
  - **using very high 2D texture resolution**
  - **no seams**
  - **2D texture to lower-resolution mesh color texture**
• **Hybrid** (not ideal, but easier to implement)

Pipeline:

- **Modeling**
- **Mapping**
- **Painting**
- **Conversion**
- **Rasterize**
- **Conversion**

Not-as-costly iterations:

- 2D texture to mesh colors
- Mesh colors to 2D texture

Using **very high** 2D texture resolution

Preserving texture data for iterations

No seams

2D texture to lower-resolution mesh color texture