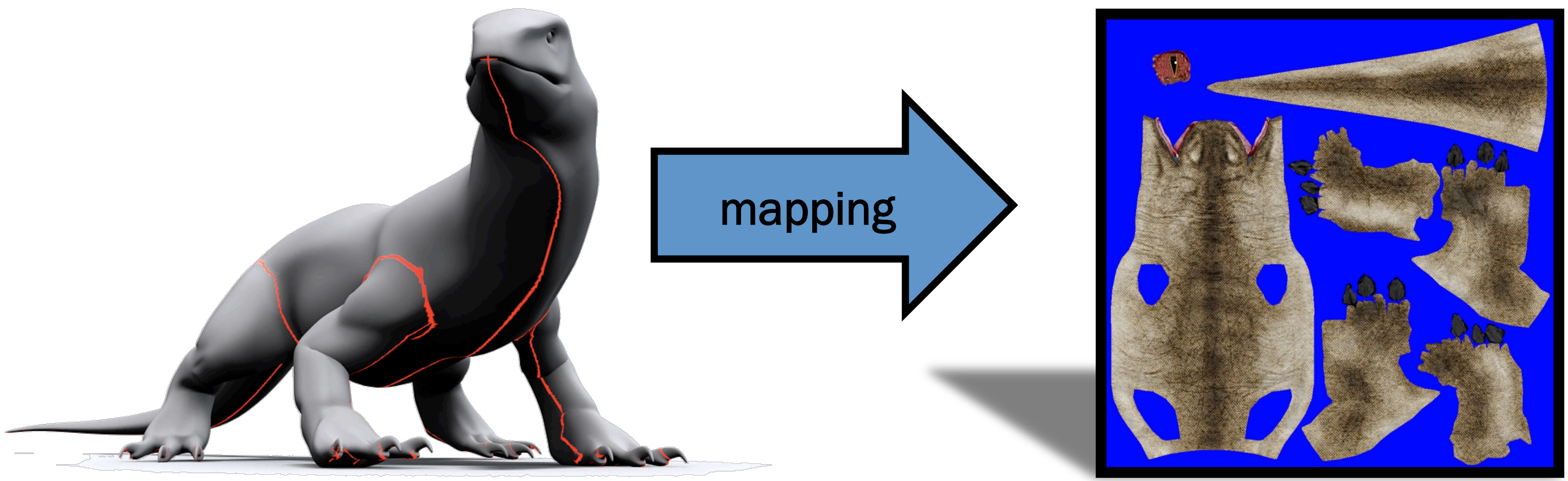
High-Performance Graphics 2017

# Mesh Color Textures

Cem Yuksel
("Jem Youksell")
*University of Utah*

# Texture Mapping




Model Space

Texture Space

# Texture Mapping

**Authoring Problems:**

- × Labor intensive
- × No local resolution readjustment
- × No model editing
- × Wasted space

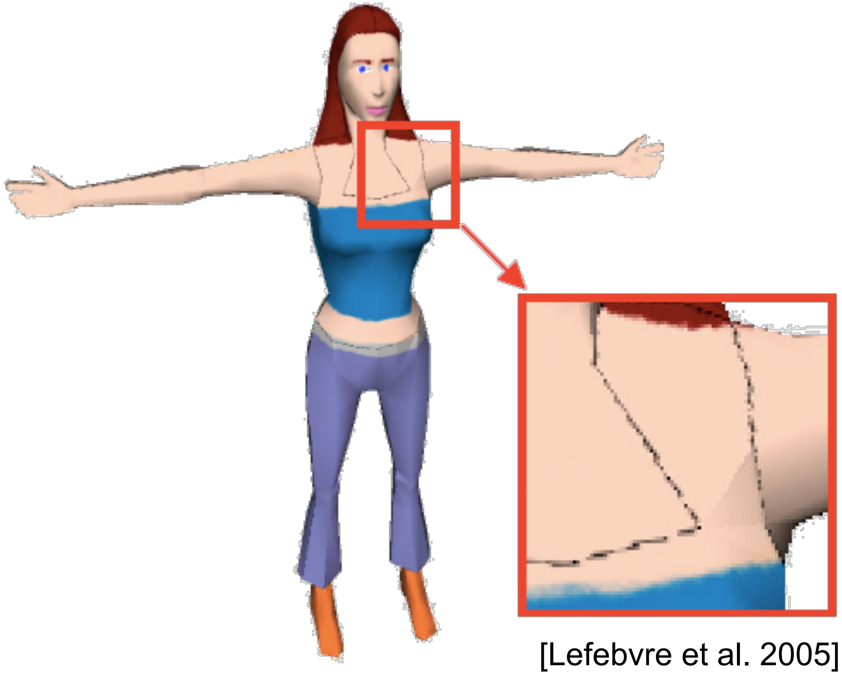
**Rendering Problems:**

- × Seam artifacts!
- × Incorrect mip-maps
- × Incorrect anisotropic filtering

# Texture Mapping

- Bilinear filtering errors near seams

High-resolution texture

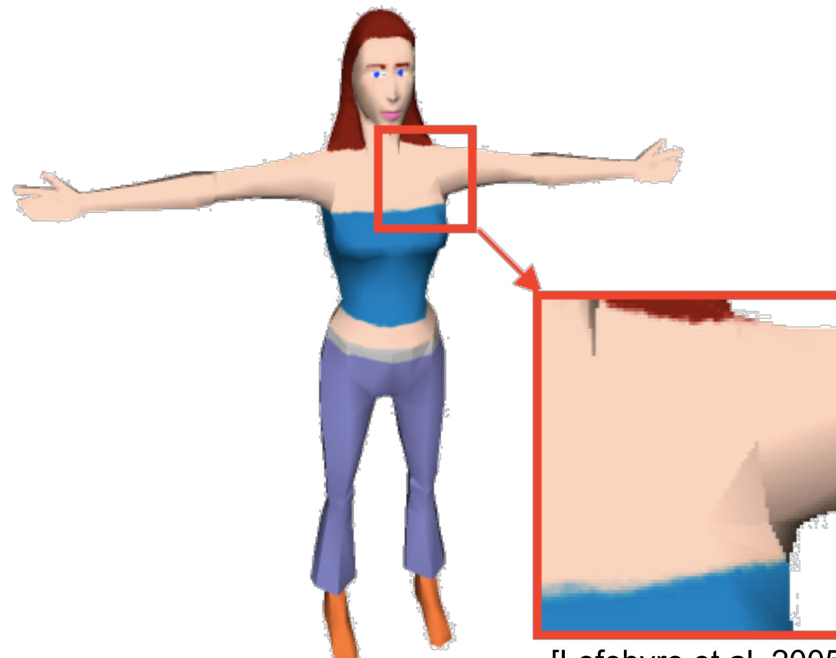


[Lefebvre et al. 2005]

# Texture Mapping

- Errors can often be hidden

**High-resolution texture**

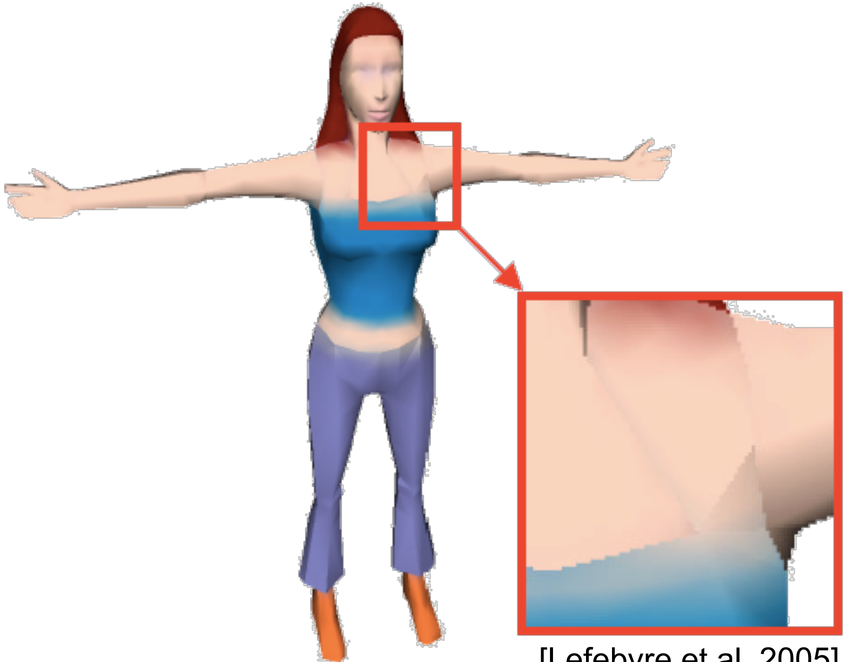


[Lefebvre et al. 2005]

# Texture Mapping

- Errors show up in mip-map levels

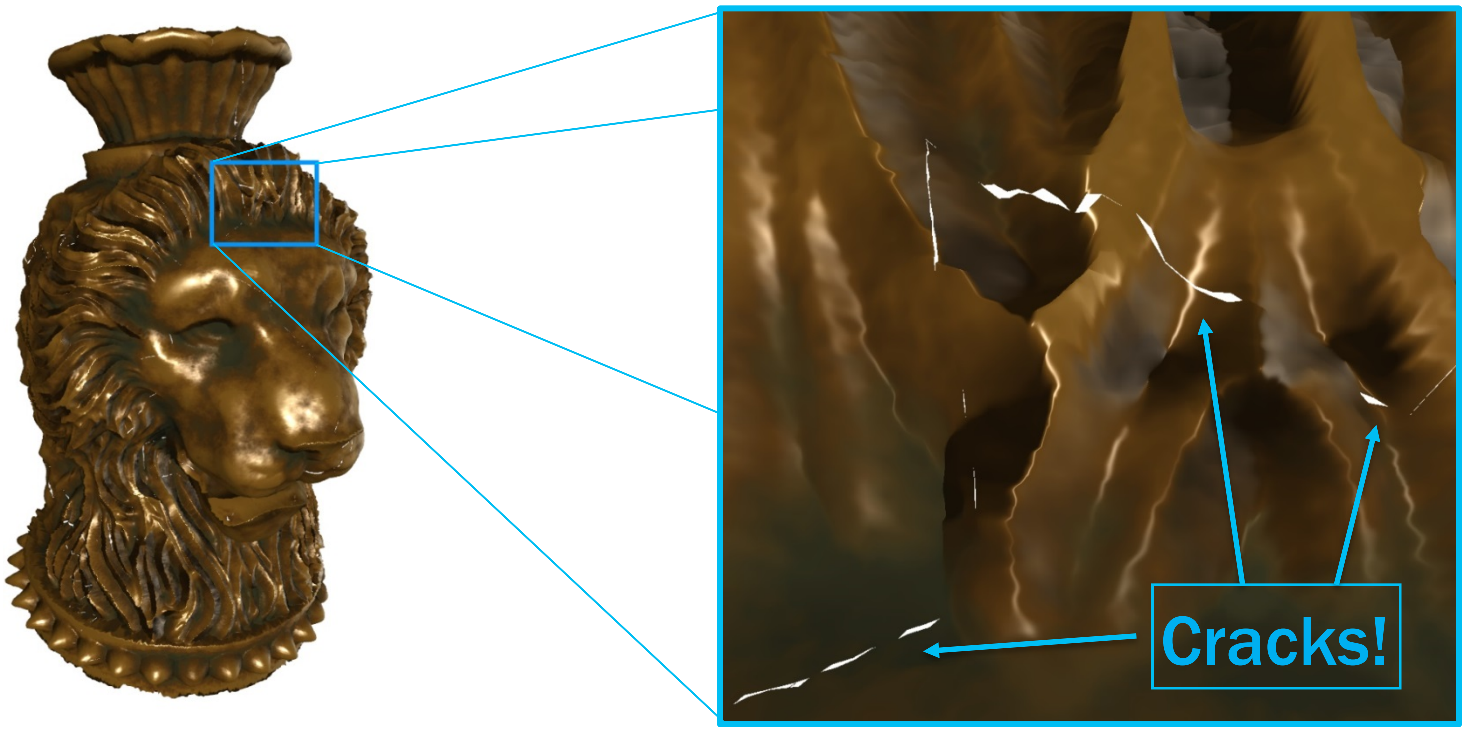
Low-resolution mip-map level



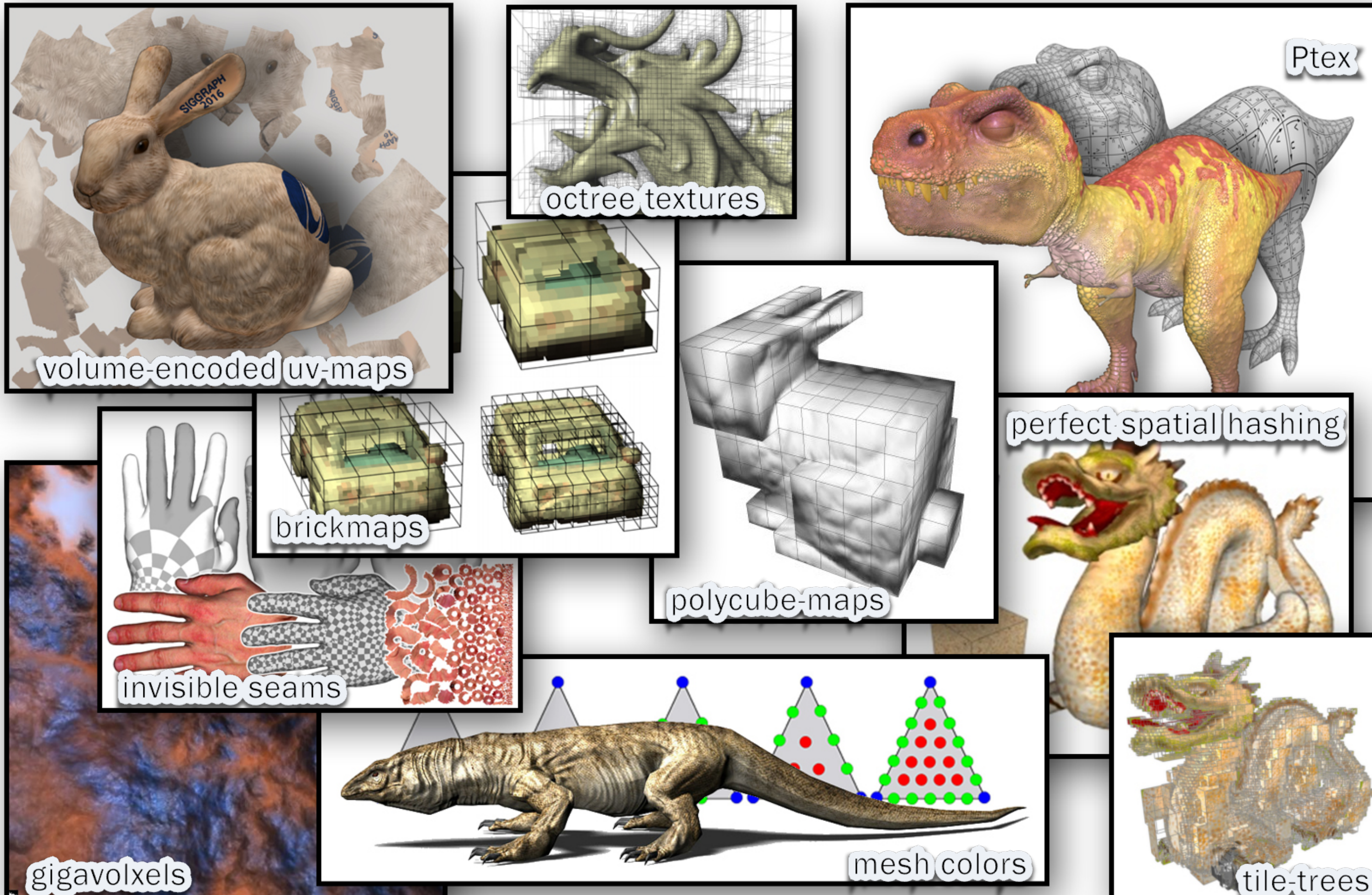
[Lefebvre et al. 2005]

# Texture Mapping

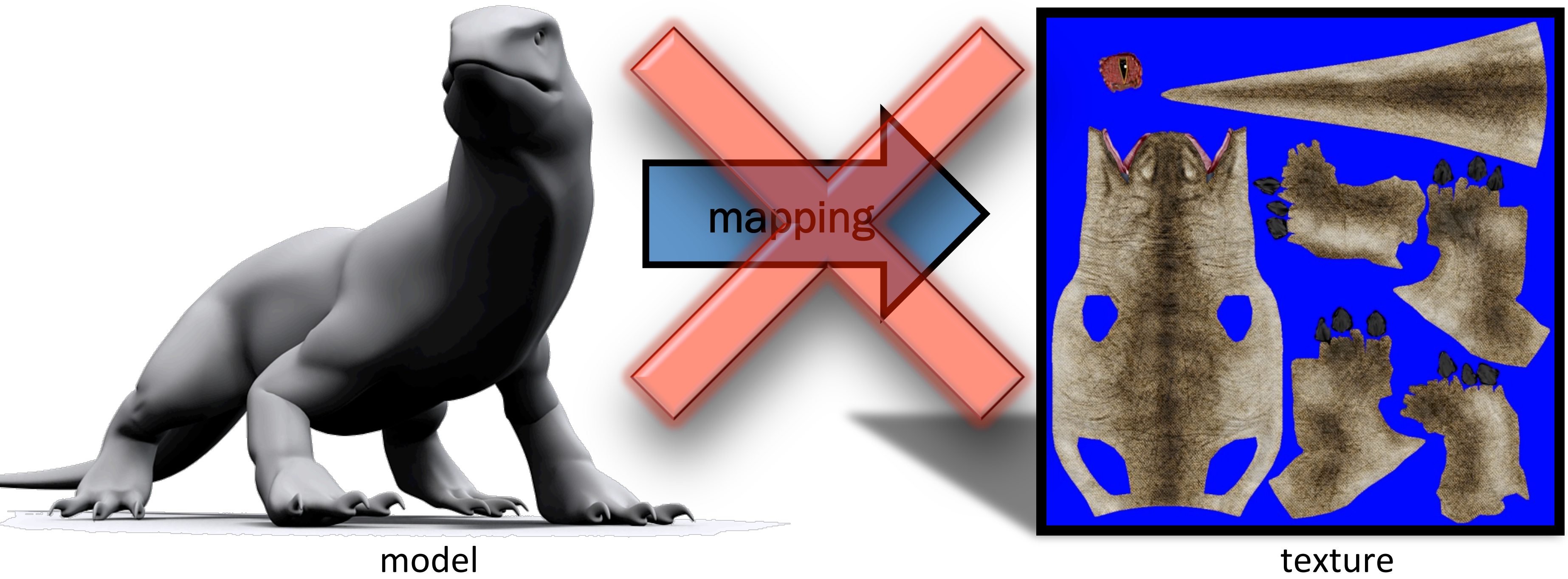
- Errors cause cracks with displacement



# SIGGRAPH 2017 Course: Rethinking Texture Mapping

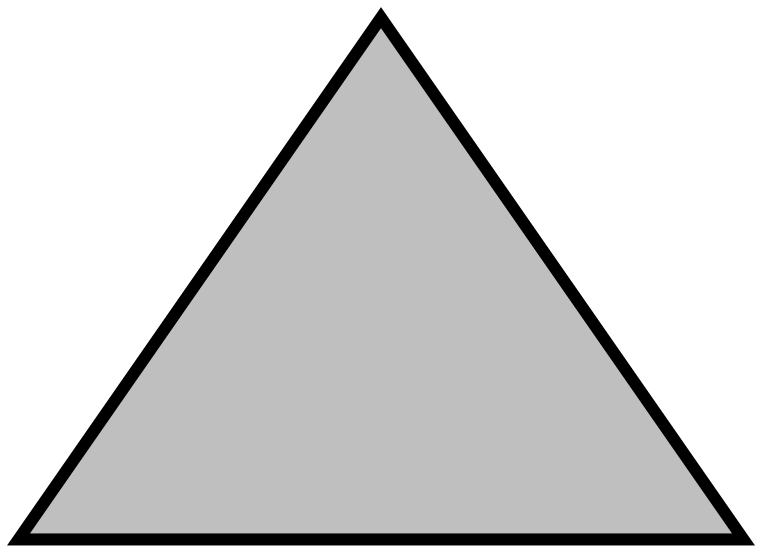


# Texture Mapping



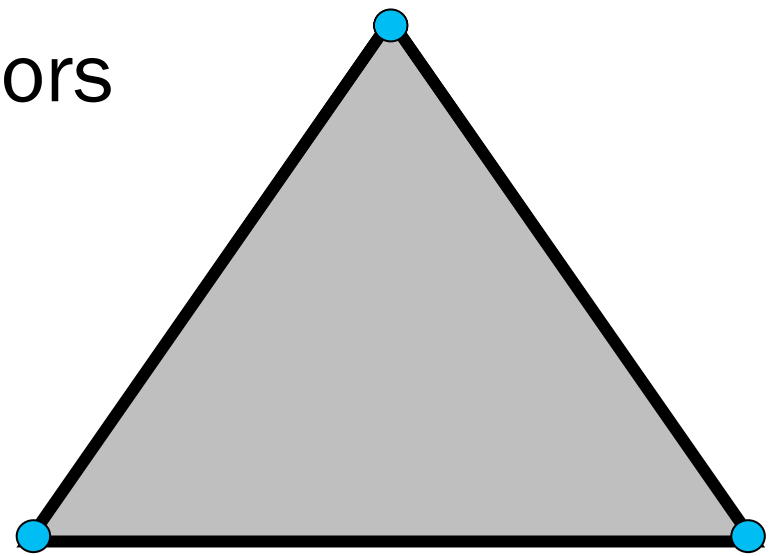
# Mesh Colors

# Mesh Colors



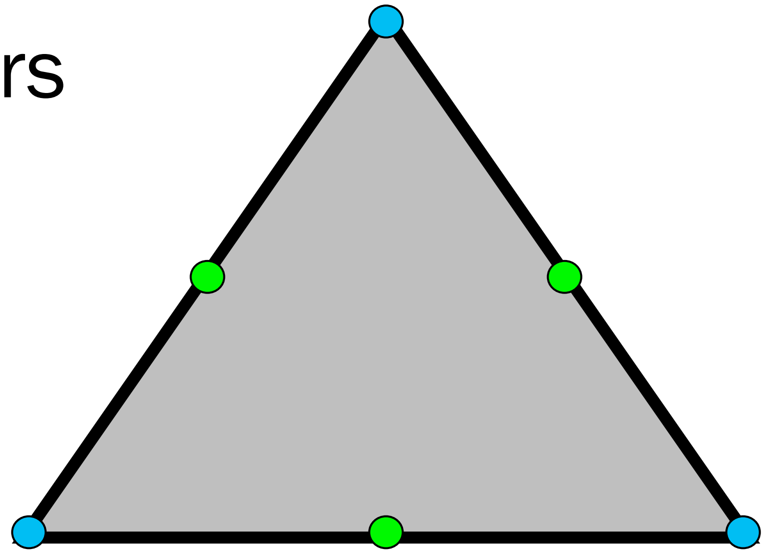
# Mesh Colors

- $R = 1$
- Vertex Colors



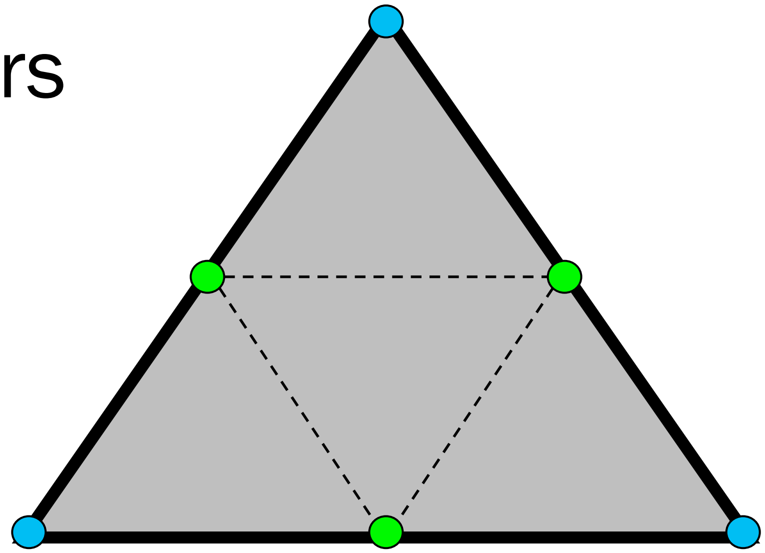
# Mesh Colors

- $R = 2$
- Edge Colors



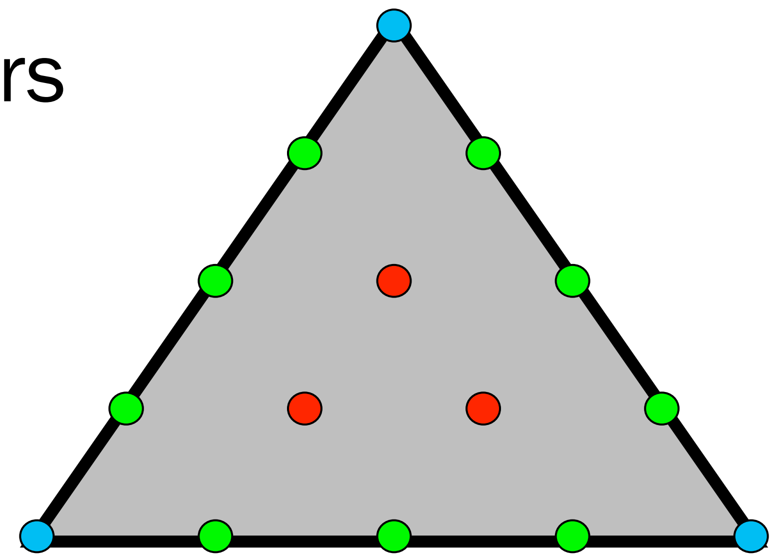
# Mesh Colors

- $R = 2$
- Edge Colors



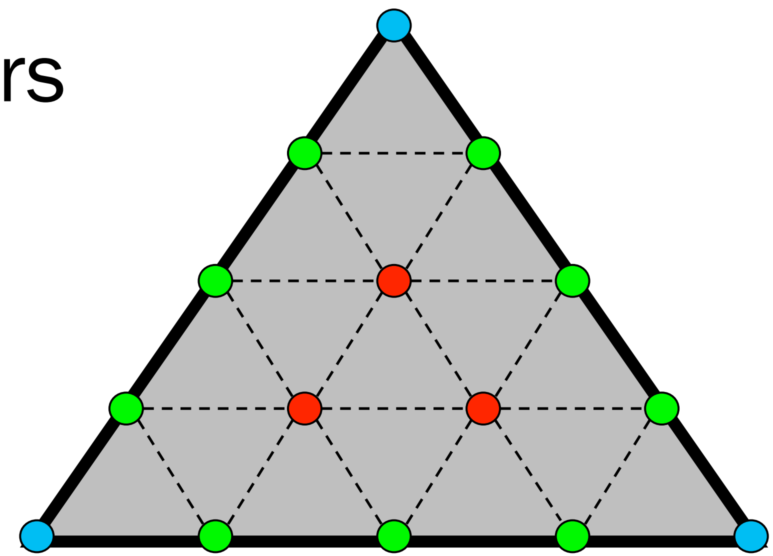
# Mesh Colors

- $R = 4$
- Face Colors



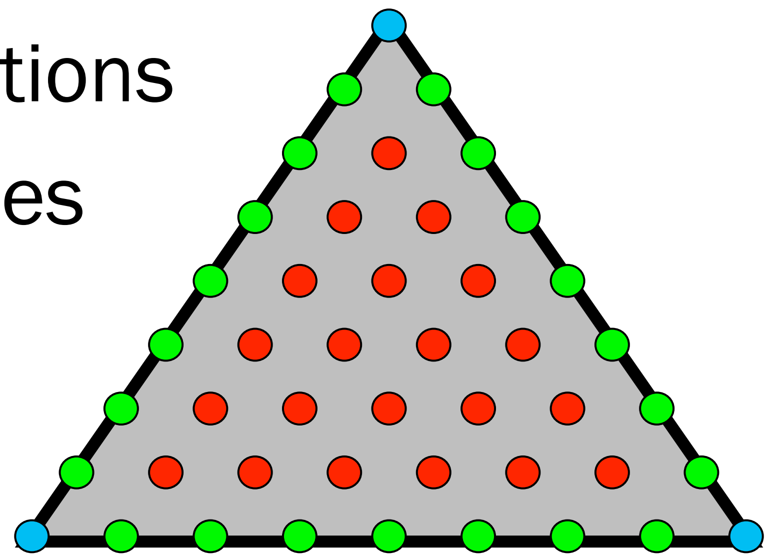
# Mesh Colors

- $R = 4$
- Face Colors



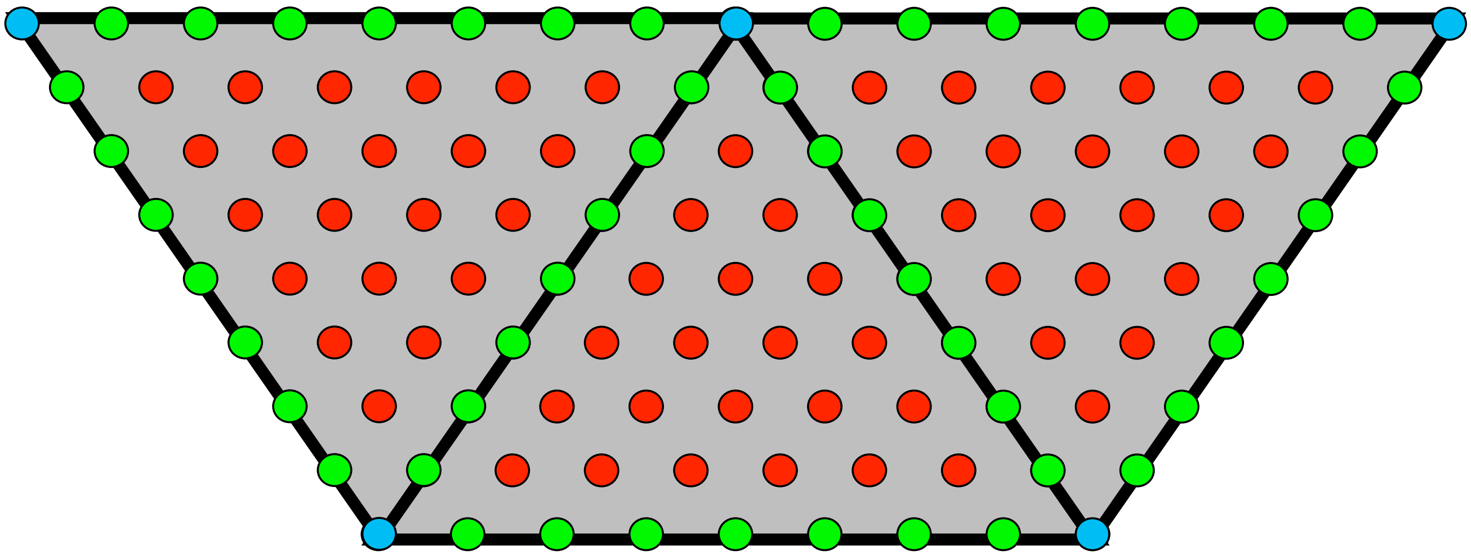
# Mesh Colors

- $R = 8$
- Color positions from indices



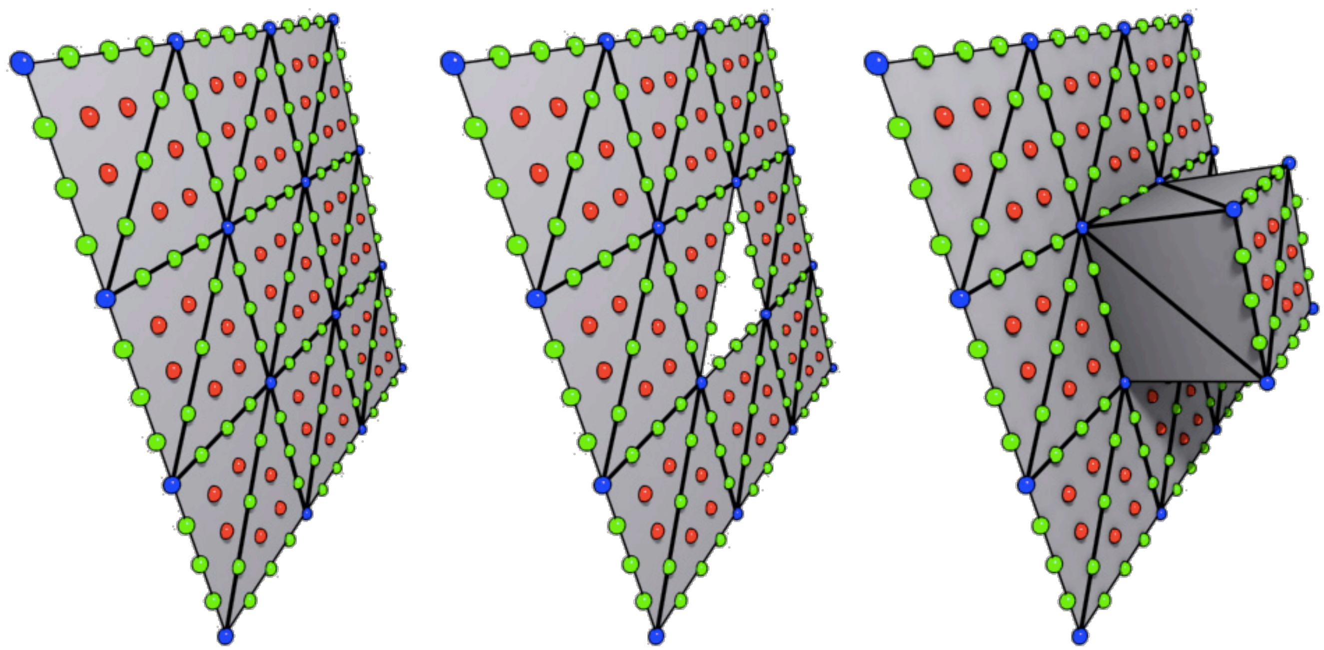
# Mesh Colors

- Colors are shared along edges
  - Guaranteed continuity



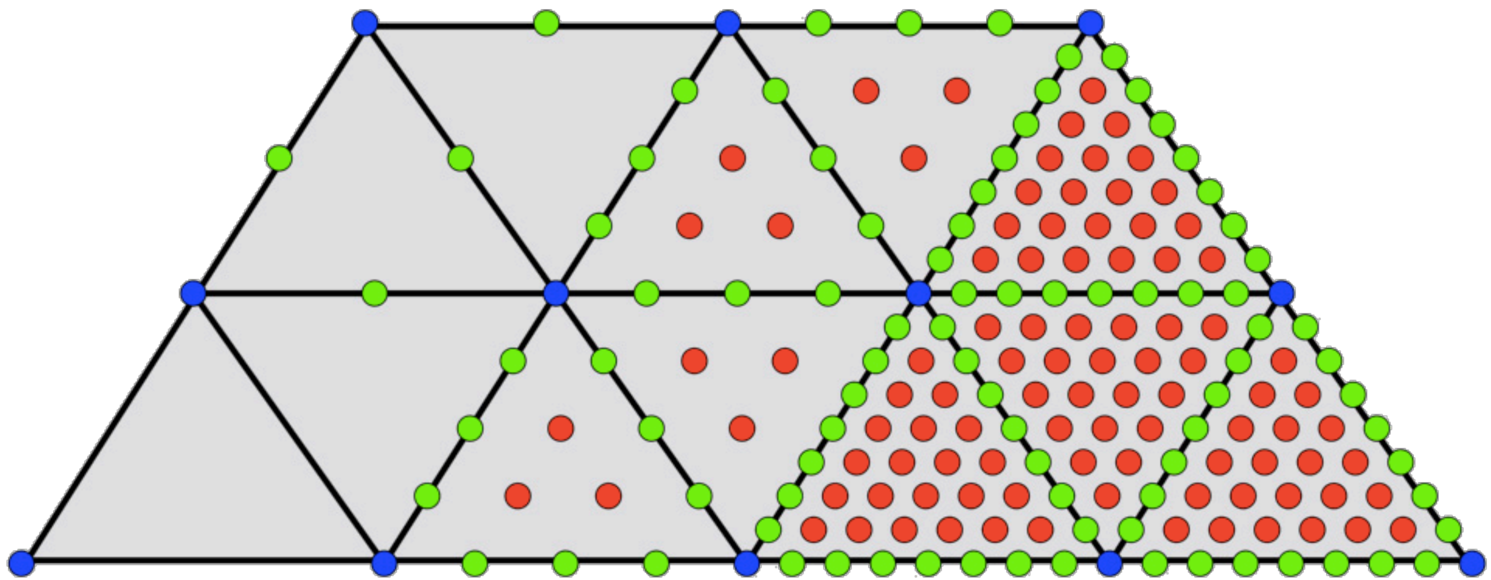
# Mesh Colors

- Model editing after texture painting



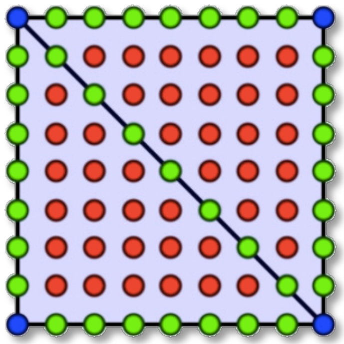
# Mesh Colors

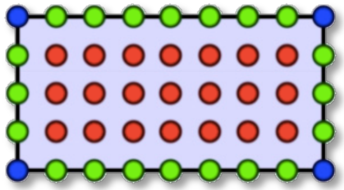
- Non-uniform face resolutions



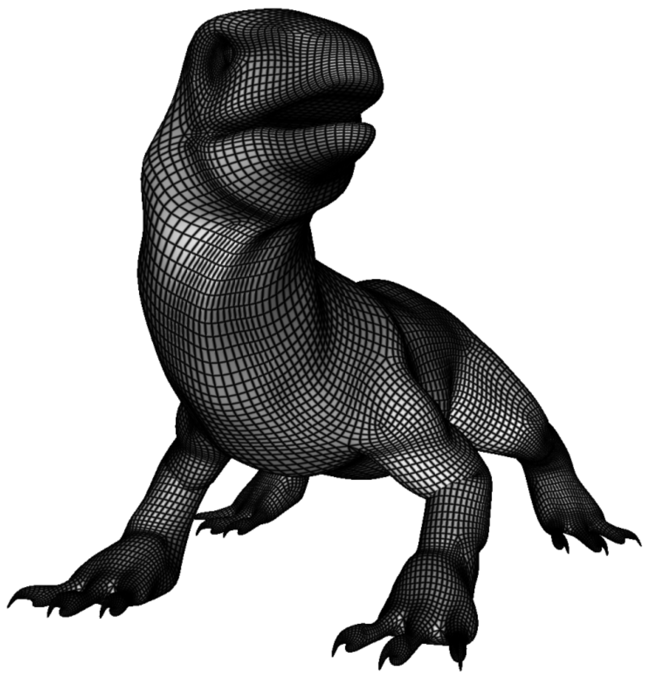
# Mesh Colors

- Non-triangular Meshes
  - Quadrilaterals
    - Triangle pair
    - Quadrilateral positioning
  - NURBS
  - Subdivision surfaces
    - Dividing faces only





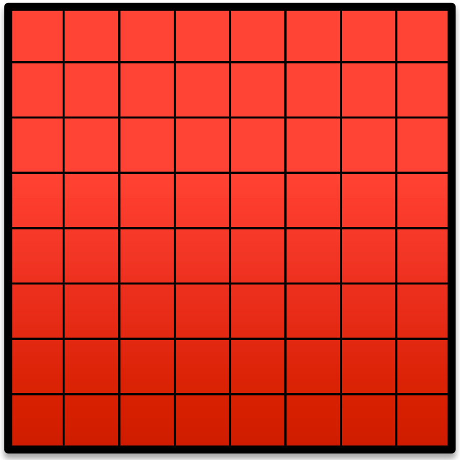
# Mesh Colors



Render Mesh

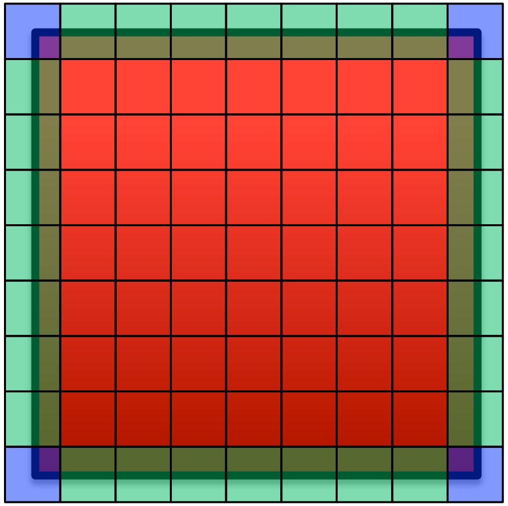
Canvas Mesh

# Mesh Colors vs Ptex



**Per-face Texture**

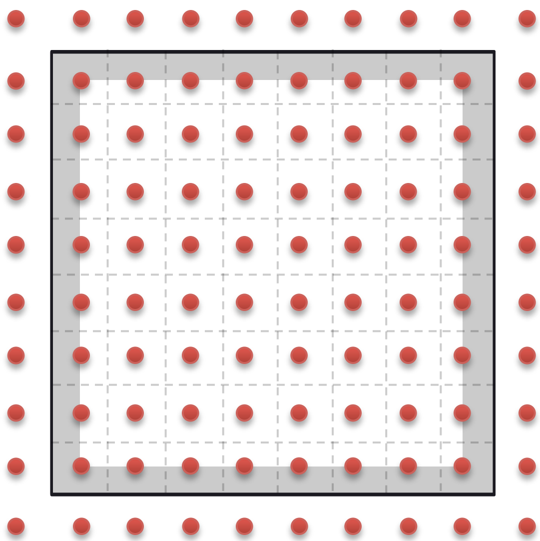
All pixels are inside the face.



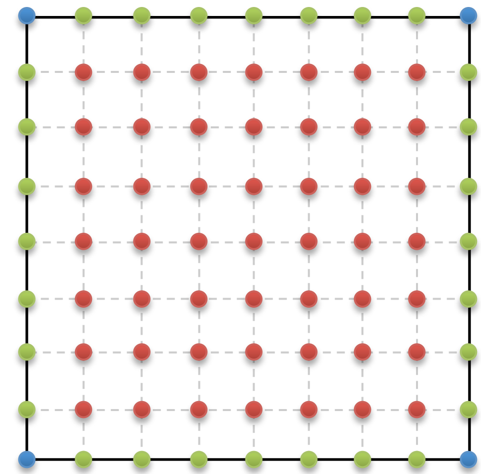
**Mesh Colors**

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

| | | | | |
|---|---|---|---|---|
| | face count | 3 K | 50 K | 218 K |
| | color count | 530 K | 530 K | 9 000 K |
| Hardware | 2D texture | 3938 fps | 2597 fps | 337 fps |
| Mesh Colors (software filtering) | Nearest | 2567 fps | 1147 fps | 273 fps |
| Mesh Colors (software filtering) | Bilinear | 2076 fps | 862 fps | 247 fps |
| Mesh Colors (software filtering) | MIP-map | 991 fps | 376 fps | 180 fps |
| Mesh Colors (software filtering) | Anisotropic | 452 fps | 152 fps | 109 fps |

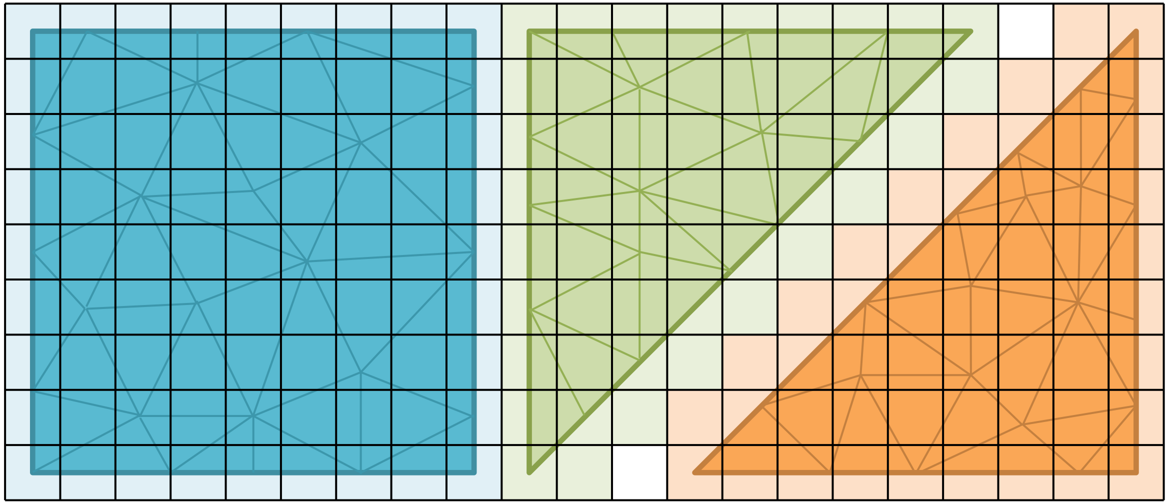
**Software Texture Filtering is SLOW!**

[Yuksel et al. 2010]

# Mesh Color Textures

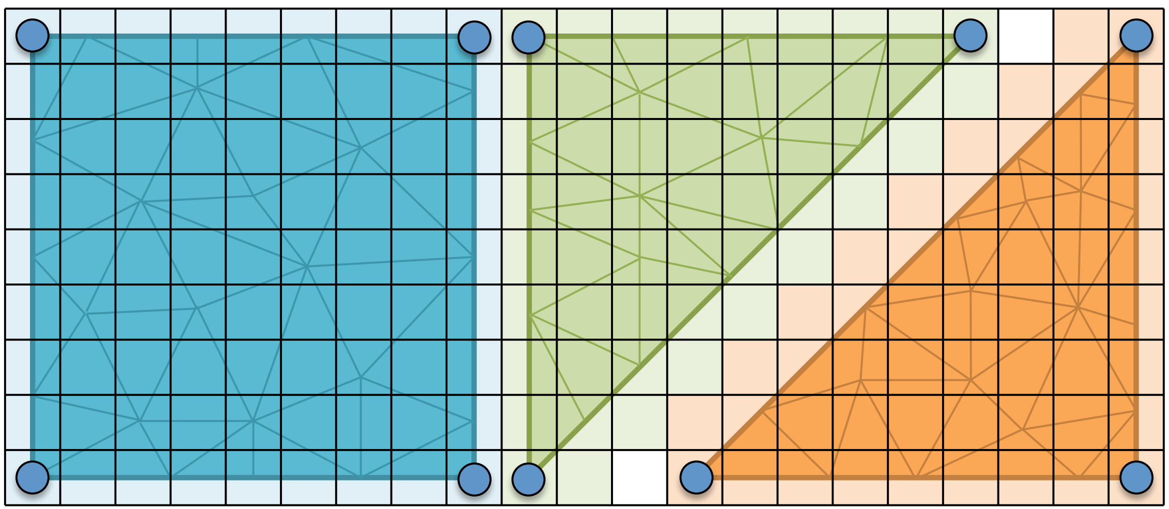
# Mesh Color Textures

- Key idea
  - Convert mesh colors to 2D textures



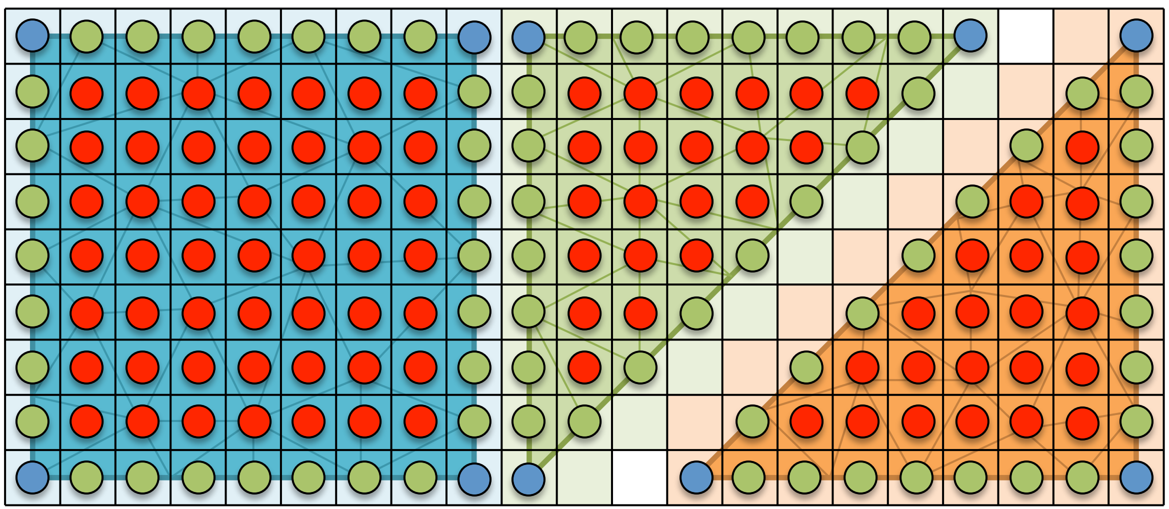
# Mesh Color Textures

- Key idea
  - Convert mesh colors to 2D textures



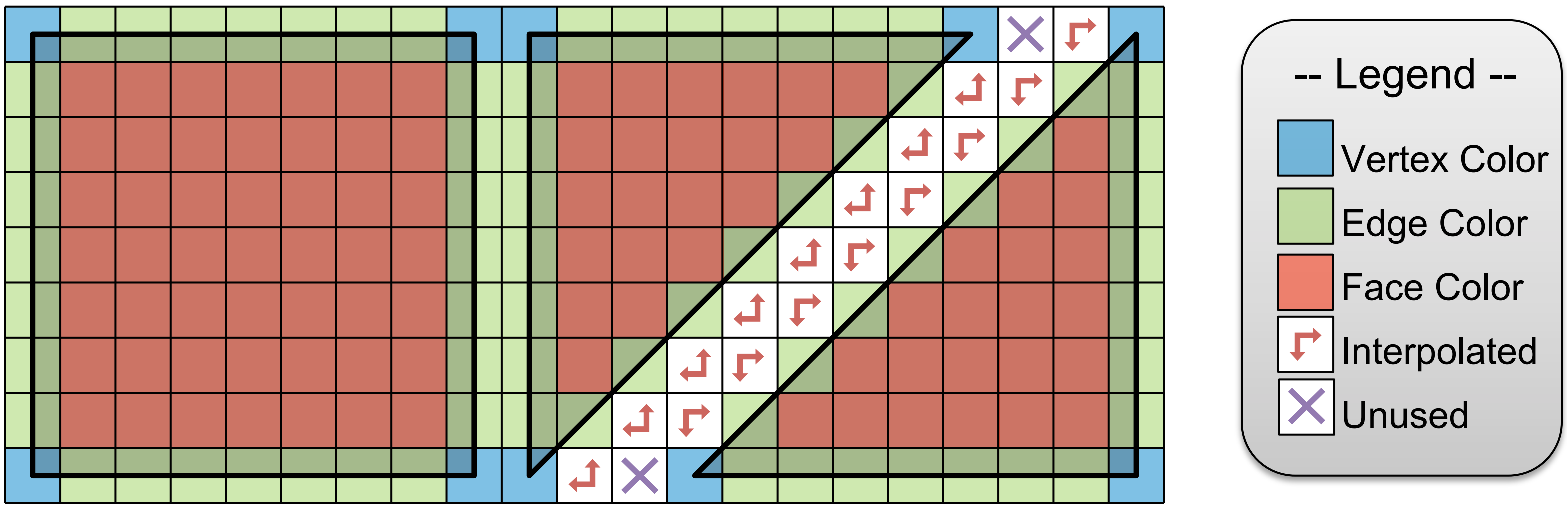
# Mesh Color Textures

- Key idea
  - Convert mesh colors to 2D textures



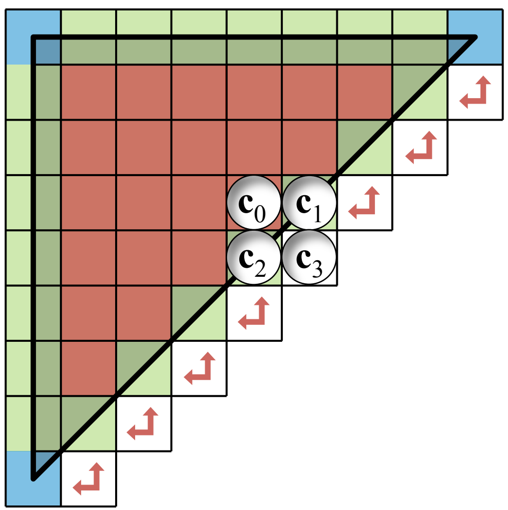
# Mesh Color Textures

- Convert mesh colors to 2D textures
  - Duplicate vertex and edge colors

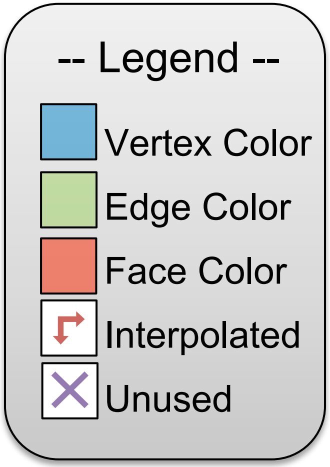


# Mesh Color Textures

- Convert mesh colors to 2D textures
  - Interpolated Colors

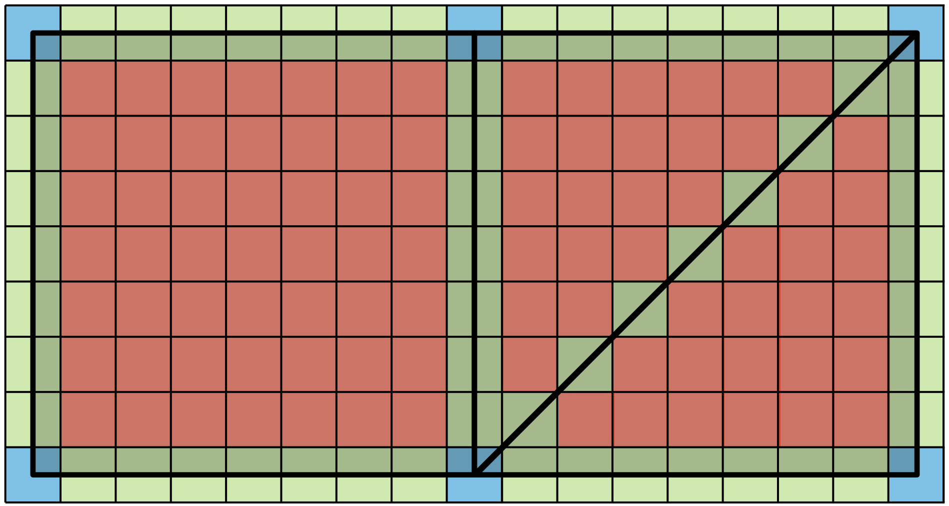


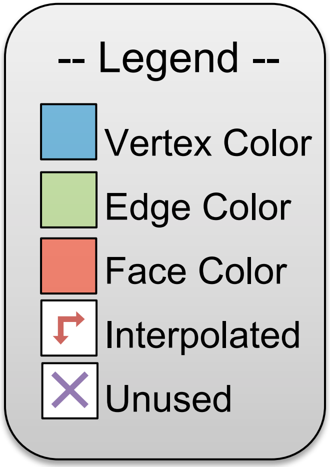

$$\mathbf{c}_3 = \mathbf{c}_1 + \mathbf{c}_2 - \mathbf{c}_0$$



# Mesh Color Textures

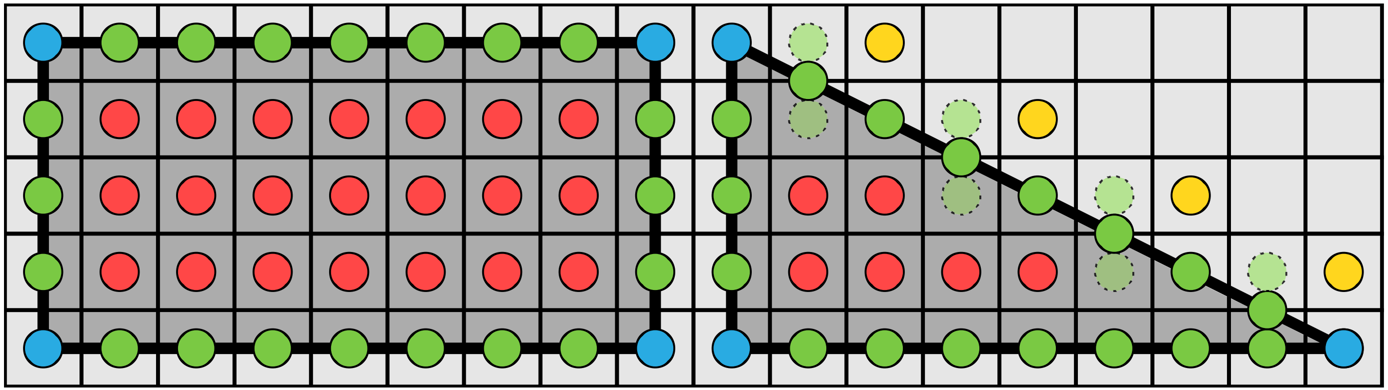
- Minimize duplicated colors
  - Place neighboring faces together





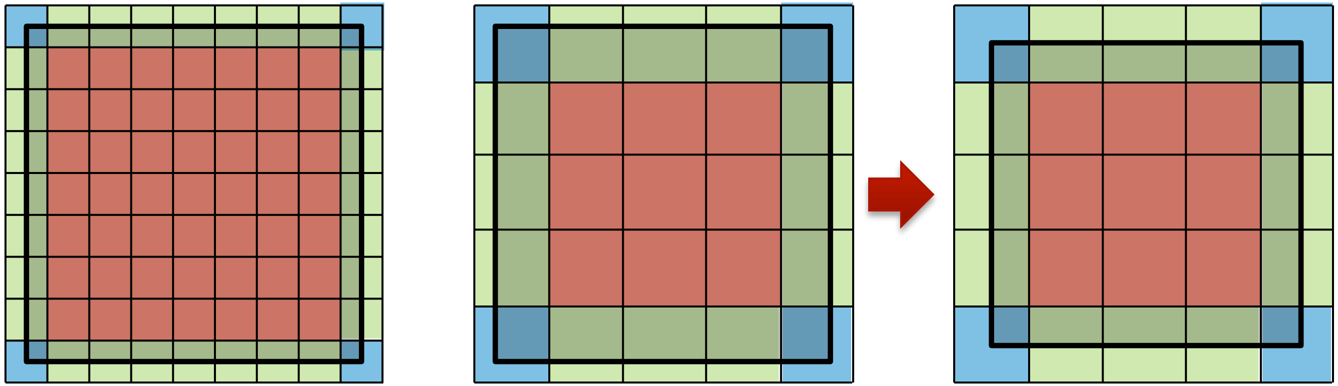
# Mesh Color Textures

- Nonuniform Mesh Colors
  - Use two resolutions per face



# Mesh Color Textures

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



Level 0 Level 1 Level 1

# Mesh Color Textures

- Trilinear filtering (mip-maps)
  - 4D texture coordinates

$$\mathbf{u}_\ell = \mathbf{u}_s \,/\, 2^\ell + \mathbf{u}_\delta$$

2D texture coordinate for level $\ell$

2D scalable coordinate

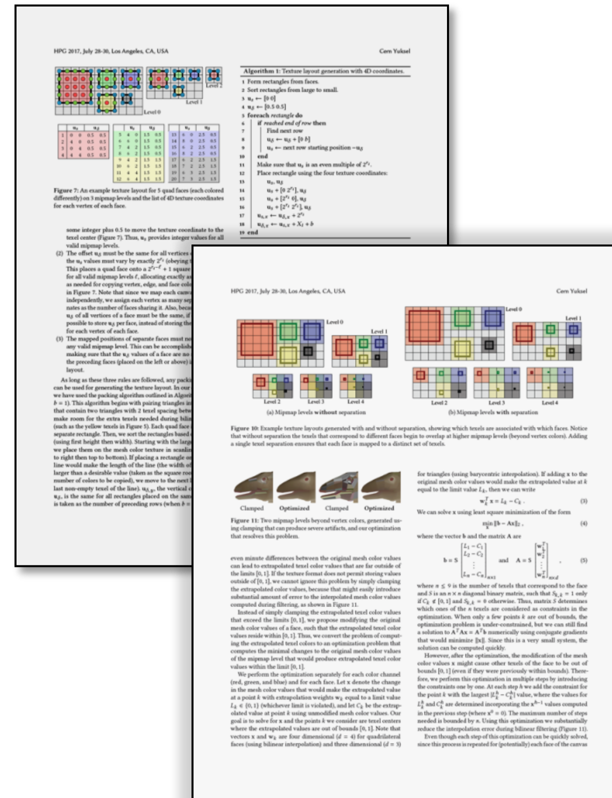
2D constant offset

4D

  - Packing problem
  - Intuitively,
    - $\mathbf{u}_s$: # of preceding edge colors (+1 per face)
    - $\mathbf{u}_\delta$: # 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**

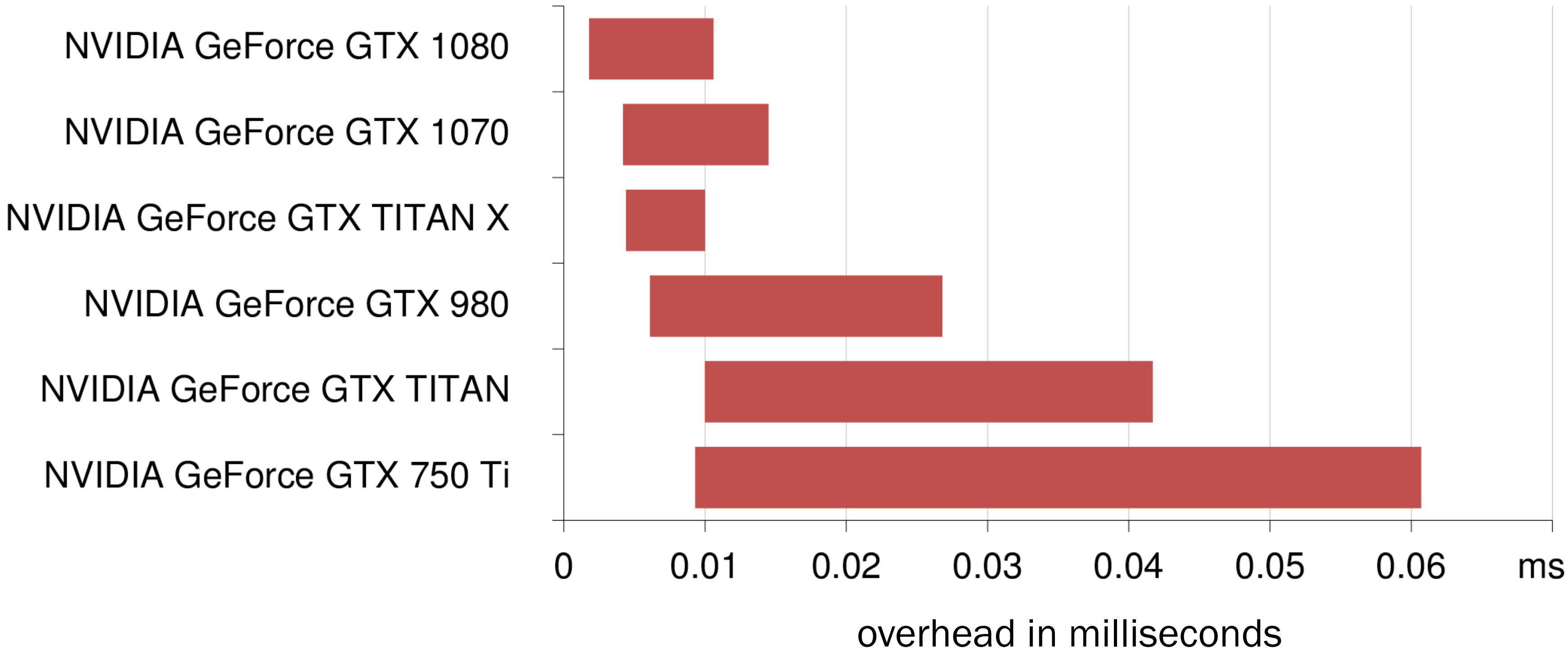


# Fragment Shader for Mip-map Filtering

```
#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



# Mesh Color Textures

- Wasted texture space due to packing

| | | | | |
|---|---|---|---|---|
| Standard 2D texture | 26% | 39% | 19% | 17% |
| Mesh Color Texture | 15% | 15% | 15% | 16% |

# 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

# Conclusion

- Advantages of Mesh Colors
  - Improved authoring
  - Improved rendering
- Speed of 2D textures

*Mesh Color Textures bring the advantages of Mesh Colors to real-time rendering.*

# Acknowledgements


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

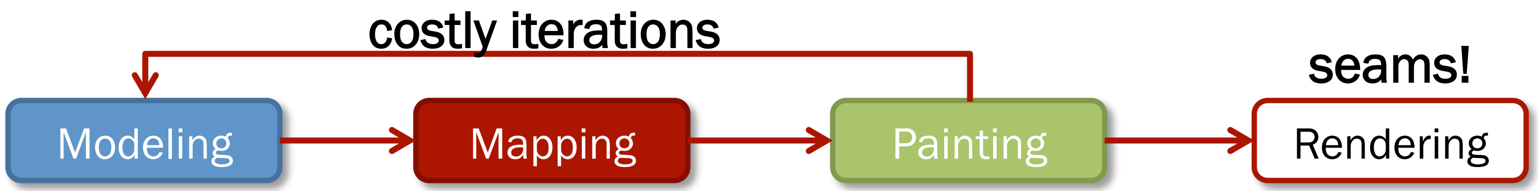
# Questions?

Mesh Color Textures

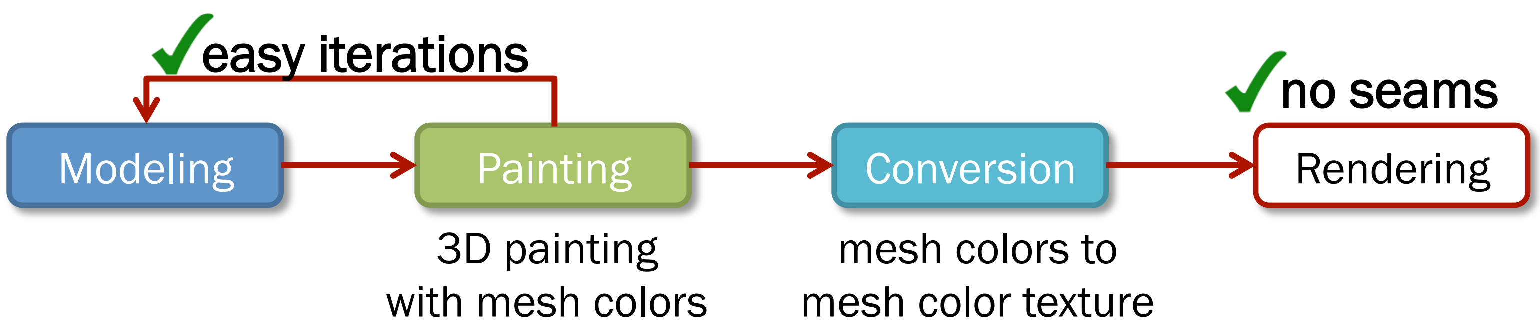
Cem Yuksel
("Jem Youksell")
*University of Utah*

# Pipeline

- Standard 2D textures

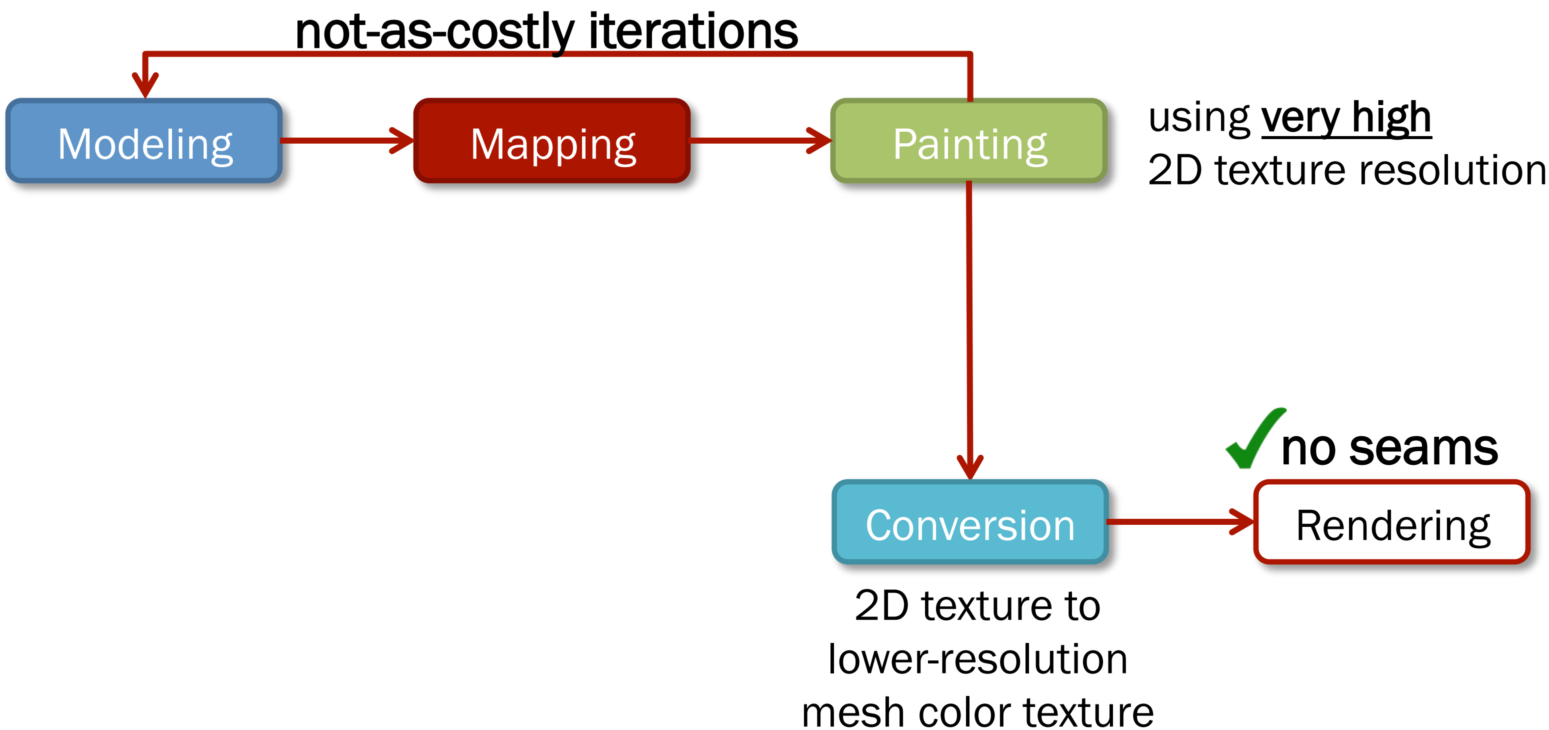



- Mesh Color Textures



# Pipeline

- Hybrid (not ideal, but easier to implement)



# Pipeline

- Hybrid (not ideal, but easier to implement)