PATCH TEXTURES:
HARDWARE IMPLEMENTATION OF MESH COLORS

Ian Mallett  Larry Seiler  Cem Yuksel

Utah Graphics  hpg 2019
TEXTURES . . . HAVE PROBLEMS
TEXTURES . . . HAVE PROBLEMS

Artist must make mapping
TEXTURES . . . HAVE PROBLEMS

Distortion

Seams
Textures... have problems

Wasted space
Textures... have problems

Incorrect mipmap reduction
Textures . . . Have Problems

Incorrect anisotropic filters
TEXTURES . . . HAVE PROBLEMS

Local resolution adjustment means remaking texture and mapping
TEXTURES . . . HAVE PROBLEMS

Artist must make mapping
L O T S O F D I F F E R E N T A P P R O A C H E S ... What does industry actually use?

(Figure [Yuksel et al. 2019])
TEXTURE DATA LIVING ON SURFACES!

- Ptex and Mesh Colors
TEXTURE DATA LIVING ON SURFACES!

• Film uses texture data directly on surfaces
  • Much easier to model!
  • Get all the rendering benefits too

• Ptex widely (sometimes exclusively) used
• Mesh Colors has also been used in production
**Mesh Colors and Ptex**

Every model has a list of patches, each with its own separate texture.
Mesh Colors and Ptex

(Figure modified from [Burley and Lacewell 2008] and [Yuksel et al. 2019])
Mesh Colors or Ptex . . . In Realtime?

- Possible [McDonald 2013], [Yuksel et al. 2010], ...

- Still less-practical than 2D textures due to SW (or SW+HW) implementation:
  - Software is much-slower than hardware!
  - Complicated to emulate filtering functionality
  - Implementation limitations
Mesh Color Textures [Yuksel 2017]

• Implements Mesh Colors using standard 2D textures
  • Almost as fast as 2D textures for simple filtering
  • Added shader complexity
  • Complicated implementation
  • Anisotropic filtering problematic

• What if the required HW changes to implement directly were small?
OUR CONTRIBUTION

• We show that implementing Mesh Colors on existing GPUs would require minimal HW changes

• Introduce “patch texture” representation—this is what makes the required HW changes minor!

Can leverage existing storage!
Can leverage existing filtering HW!
Edge-crossing unnecessary
Similar performance expected
Unlike in Mesh Colors, in a Patch Texture, edge and corner data is duplicated, not shared with neighbors.
Patch Textures

Standard 2D Texture

Set of Patch Textures

Individual Patch Texture
No restriction on resolution is imposed, but if $2^n+1$ on a side, smooth filtering between neighbors is possible.
Patch Textures

(u,v) and (s,t) coordinates in a patch texture
Patch Textures

2D Texture \( (u,v) \) and \( (s,t) \) coordinates in a patch texture.
Patch Texture Storage

• Very similar to standard 2D textures
Patch Texture Storage (Mipmaps)

• Very similar to standard 2D textures

• Most GPUs store in e.g. 4×4 tiles
  • If mesh textures are $2^n+1$ on a side, requires padding
Power-of-two 2D textures fit nicely in memory.
$2^n + 1$ patch textures require padding (a few %)
Patch Texture Storage (Triangular Patches)

- Triangular patch textures do not map nicely to 2D storage
- Not necessarily a problem
  - Quad-dominant meshes standard
  - Patch textures typically small
- Clever workarounds exist (though require more HW changes)
Patch Texture Storage (Triangular Patches)
**Patch Texture Storage (Triangular Patches)**

Filtering can require values outside triangle to be defined. Extra defined texels.

Ref: [Yuksel 2017]

Slice
FILTERING (QUADRILATERAL PATCHES)

• Exactly the same as for 2D textures!
  
  • (Except we don’t need the half-texel shift of 2D textures for \((s,t)\mapsto(u,v)\) conversion.)
Filtering (Triangular Patches)

• Triangular patches use barycentric filtering

• *Many* possible ways to tweak existing logic so that it can implement this

• Mostly, just pass 0 in some places (see paper)
ANISOTROPIC FILTERING

• Same process as for 2D textures

• However, we now have the chance to detect patch boundaries!
Anisotropic Filtering (Patch Boundaries)

What to do about out-of-patch samples?
Anisotropic Filtering (Patch Boundaries)

"Right" thing is to traverse to neighbor patch

This is expensive (an indirection)
Anisotropic Filtering (Patch Boundaries)

Clamping the points to the edge changes the filter shape.
Anisotropic Filtering (Patch Boundaries)

Clipping the filter is a simple alternative.
If multisampling is used, the on-screen result can be similar to the correct filter (see [Yuksel et al. 2010], [Toth 2013])
ANISOTROPIC FILTERING COMPARISON

• Any approach is acceptable
  • None of the methods reveal the edges
  • Ground truth not expected without edge-crossing

• This is because not filtering across edges, like in 2D textures
Anisotropic Filtering Comparison

Patch Edges

Ground Truth
Anisotropic Filtering Comparison

Patch Edges

Clamped
ANISOTROPIC FILTERING COMPARISON

Patch Edges

Clipped
ANISOTROPIC FILTERING COMPARISON

Patch Edges

Ground Truth
Anisotropic Filtering Comparison

Patch Edges

Clamped+MSAA
ANISOTROPIC FILTERING COMPARISON

Patch Edges  Clipped+MSAA
Proof of Concept

• Implemented all algorithms in GPU renderer
Proof of Concept

Lizard (1751 patches)

Nyra (15124 patches)

Alien (5488 patches)

Head (9094 patches)
CONCLUSION

• Implementing Mesh Colors requires only minimal changes to existing GPU hardware

• Takeaway for vendors: go implement it already! 😊
Questions
Why is it Harder with Ptex?

• Ptex and Mesh Colors have minor theoretical differences (they are duals of each-other), but this leads to significant difficulty in-practice.

• Main problem is that edge-crossing is required for correct filtering in Ptex.
Why is it harder with Ptex?

Data from neighbor required for correct bilinear filter near edge.
Why is it harder with Ptex?

Unclear how to correctly filter here. Different solutions, all involving edge-crossing. This means substantial HW changes!
Duplicating edge data avoids crossing, but is inconsistent.
FILTERING (QUADRILATERAL PATCHES)
FILTERING (TRIANGULAR PATCHES)