SIGGRAPH2011

Per-Face Texture Mapping for Realtime Rendering

“Realtime Ptex”

Quick Note

- For CUDA/Compute folks:
  - Ptex != PTX

Goals

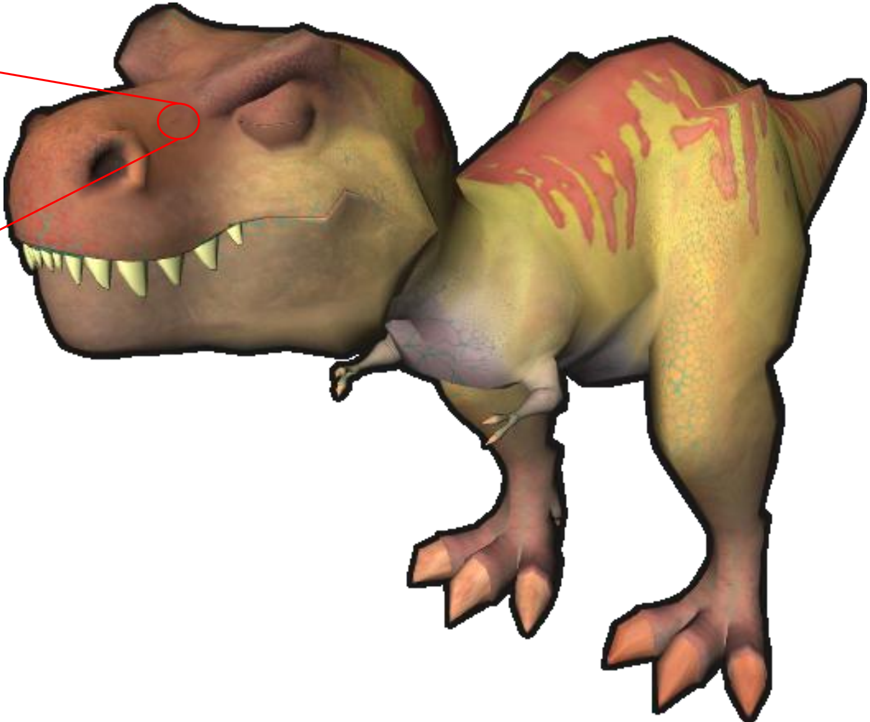
- Render native Ptex datasets in real-time on commodity hardware
- Remove texture seams from textured models
- Remove expensive, manual model unwrap step from art pipeline
- Support arbitrary resolutions on a per-face basis

Video Time!

Video Recap

Stored as 8Kx8K

Rendered as 2Kx2K

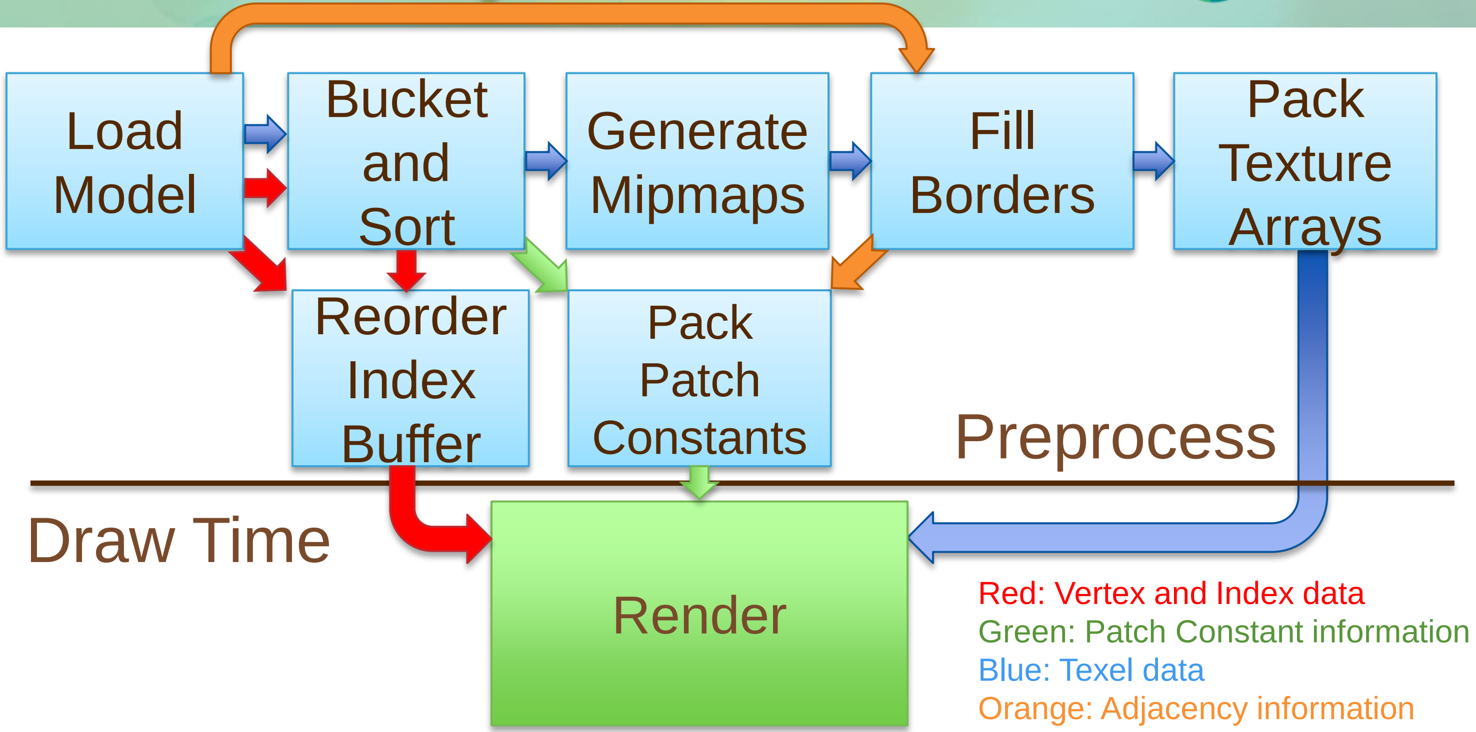


Model Statistics

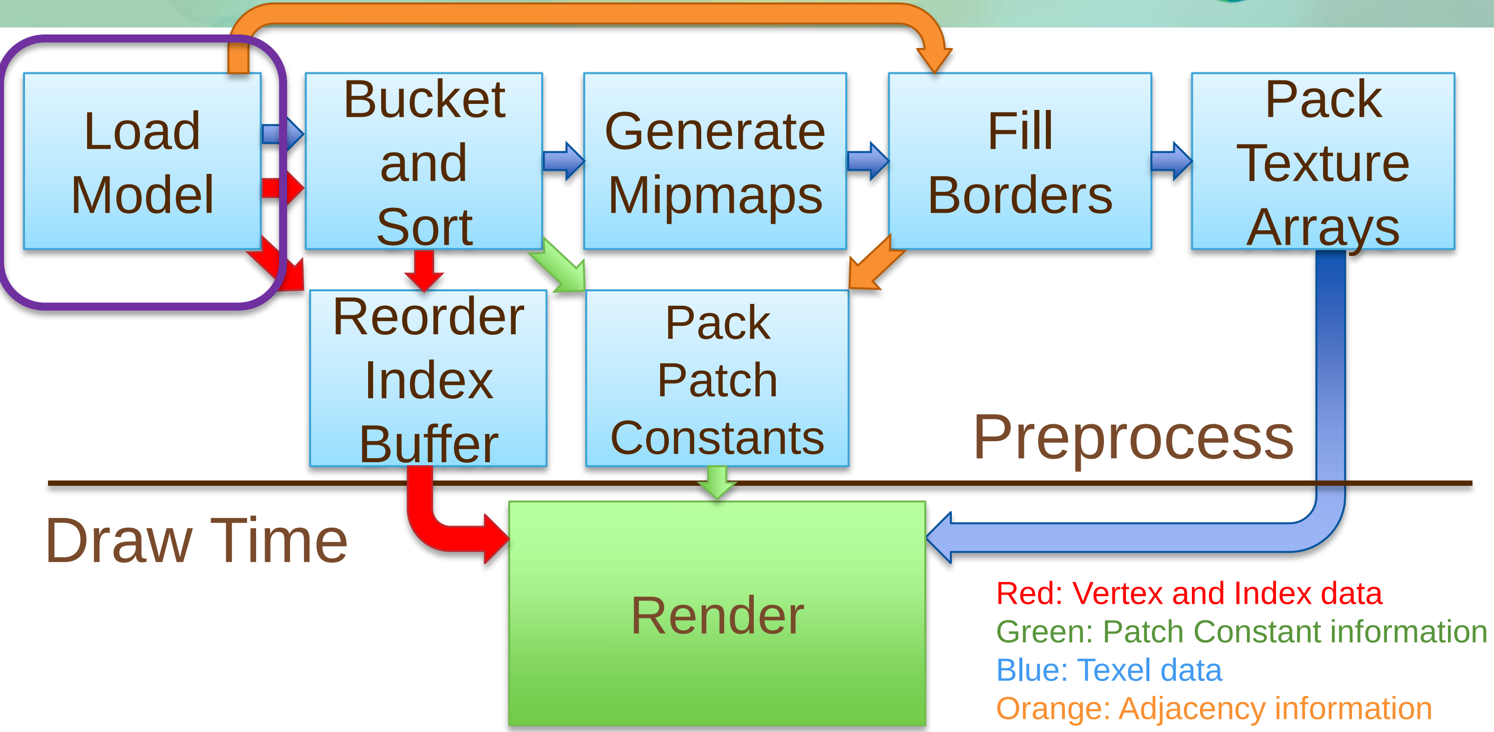
SIGGRAPH2011

- ~800 FPS on GTX 460 with no optimization
- 278M of Color Texture Data
- 5812 Patches

General Steps



Load Model



Load Model
Bucket and Sort
Generate Mipmaps
Fill Borders
Pack Texture Arrays
Reorder Index Buffer
Pack Patch Constants
Preprocess
Draw Time
Render
Red: Vertex and Index data
Green: Patch Constant information
Blue: Texel data
Orange: Adjacency information

Load Model

SIGGRAPH2011

- Vertex Data
  - Any geometry arranged as a quad-based mesh
  - Example: Wavefront OBJ
- Patch Texture
  - Power-of-two texture images
- Adjacency Information
  - 4 Neighbors of each quad patch
- Easily load with library available from http://ptex.us/

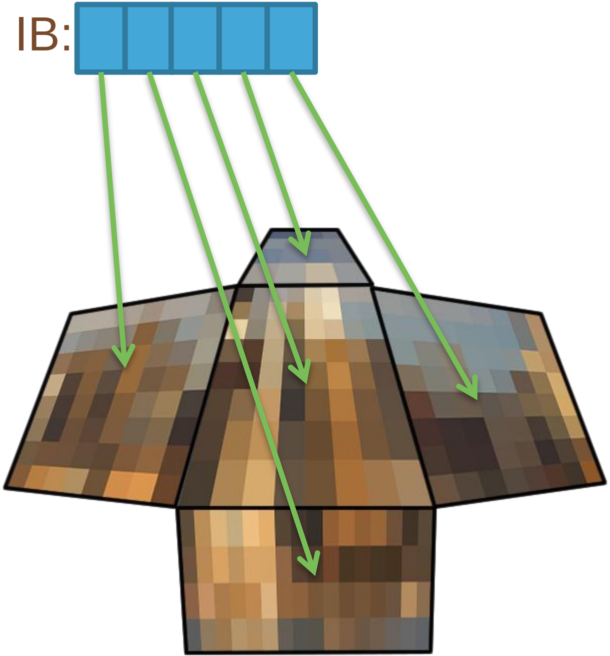
Load Model (cont'd)

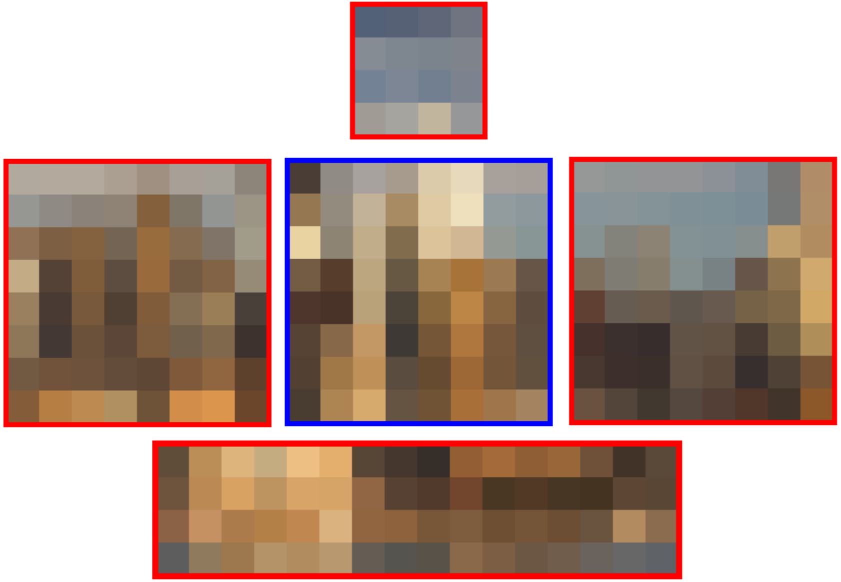
- Texel Data
  - Per face, load the largest available mipmap level from the source files
  - In memory, place the loaded texel data into a memory buffer that has a fixed-size texel border region
  - The borders must be big enough for the largest filter kernel you will support (Border size = ½ filter kernel size)
    - Bilinear (2x2 kernel): 1 pixel border
    - 16x Aniso (16x16 kernel): 8 pixel border

Load Model

VB: ...

IB:

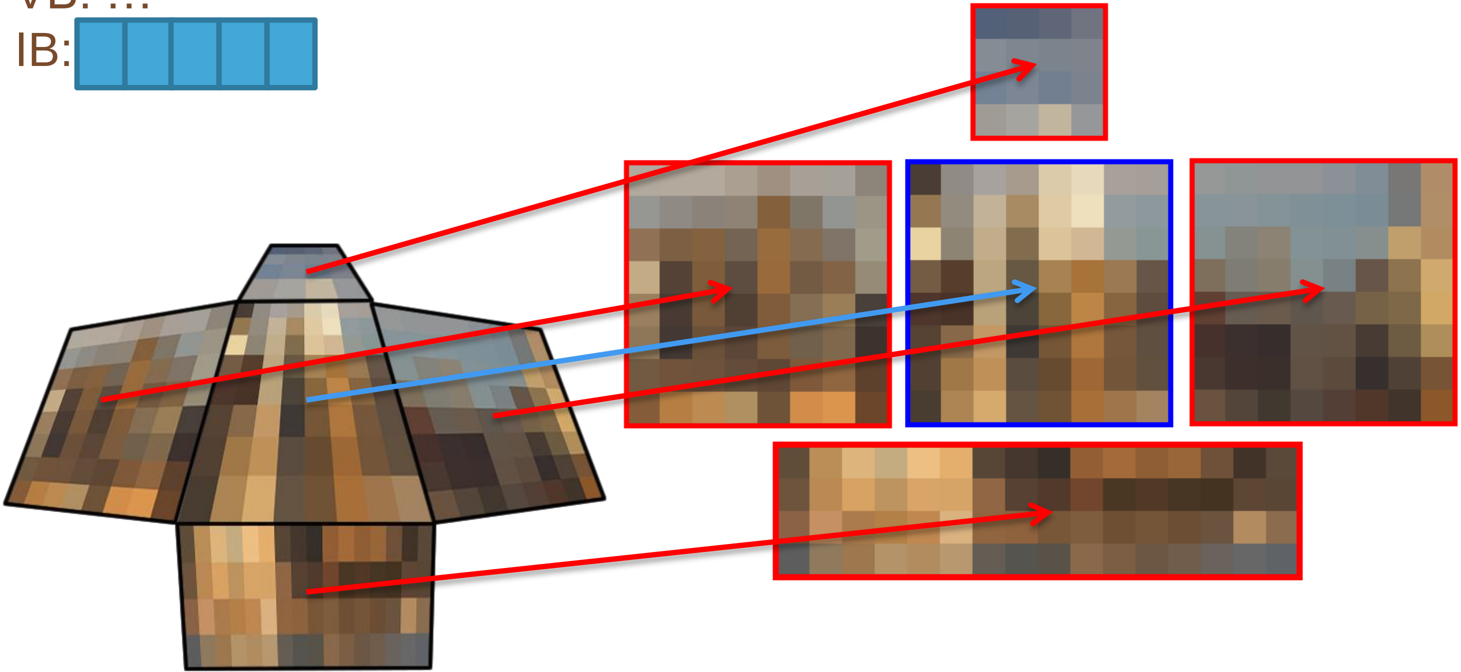




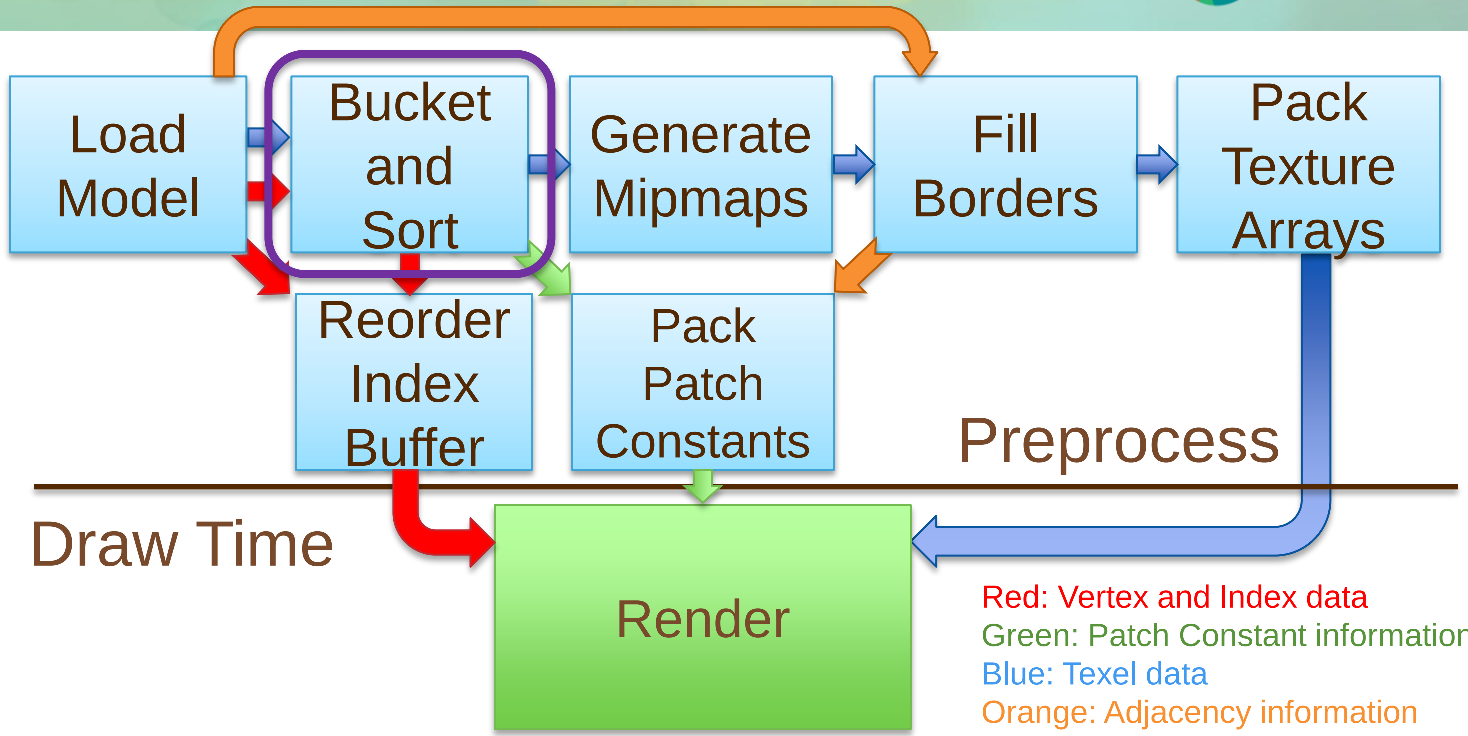
Load Model

VB: …

IB:



Bucket and Sort



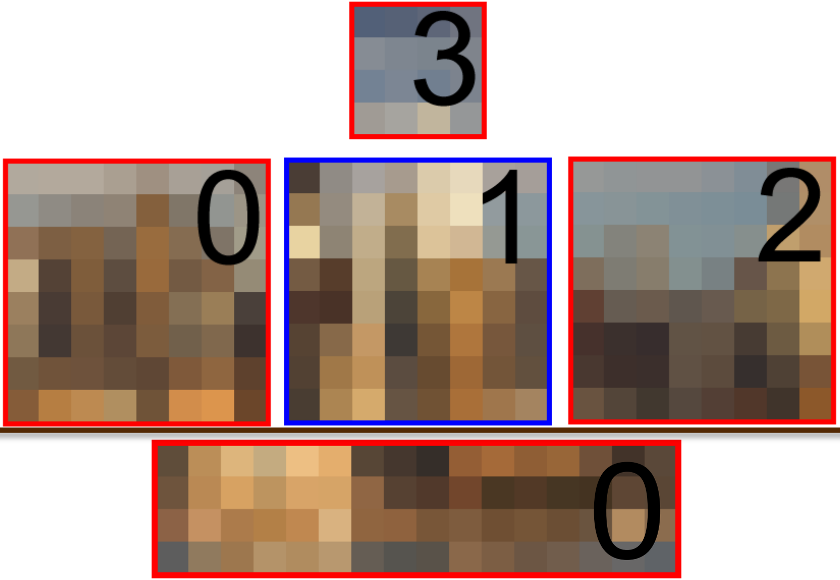
Red: Vertex and Index data
Green: Patch Constant information
Blue: Texel data
Orange: Adjacency information

Bucket and Sort

SIGGRAPH2011

- Bucket ptex surfaces into groups by Aspect Ratio
  - Each Aspect Ratio will be one bucket
  - 1:1 bucket, 2:1 bucket, 4:1 bucket, etc
- Then, within each bucket, sort by decreasing surface size and assign IDs.
  - This allows us to densely pack texture arrays, avoiding “empty” surfaces.

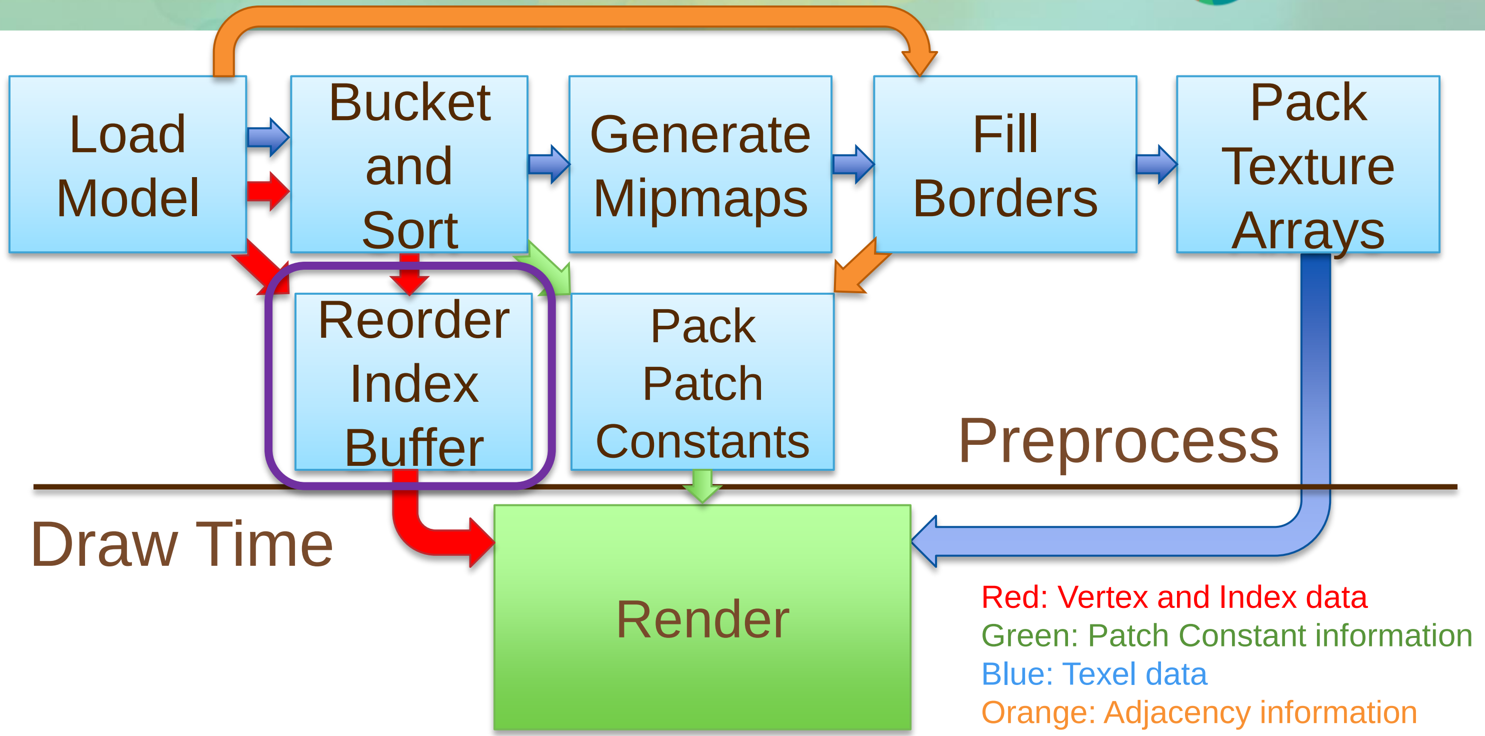
Bucket and Sort



Bucket 1:1

Bucket 4:1

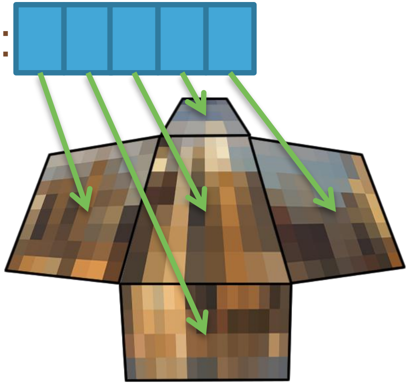
Reorder Index Buffer



Reorder Index Buffer

- Original

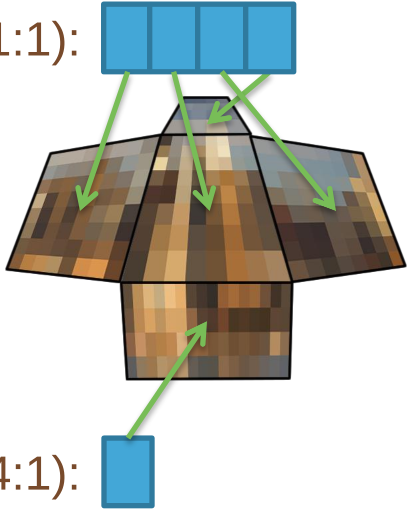
IB:



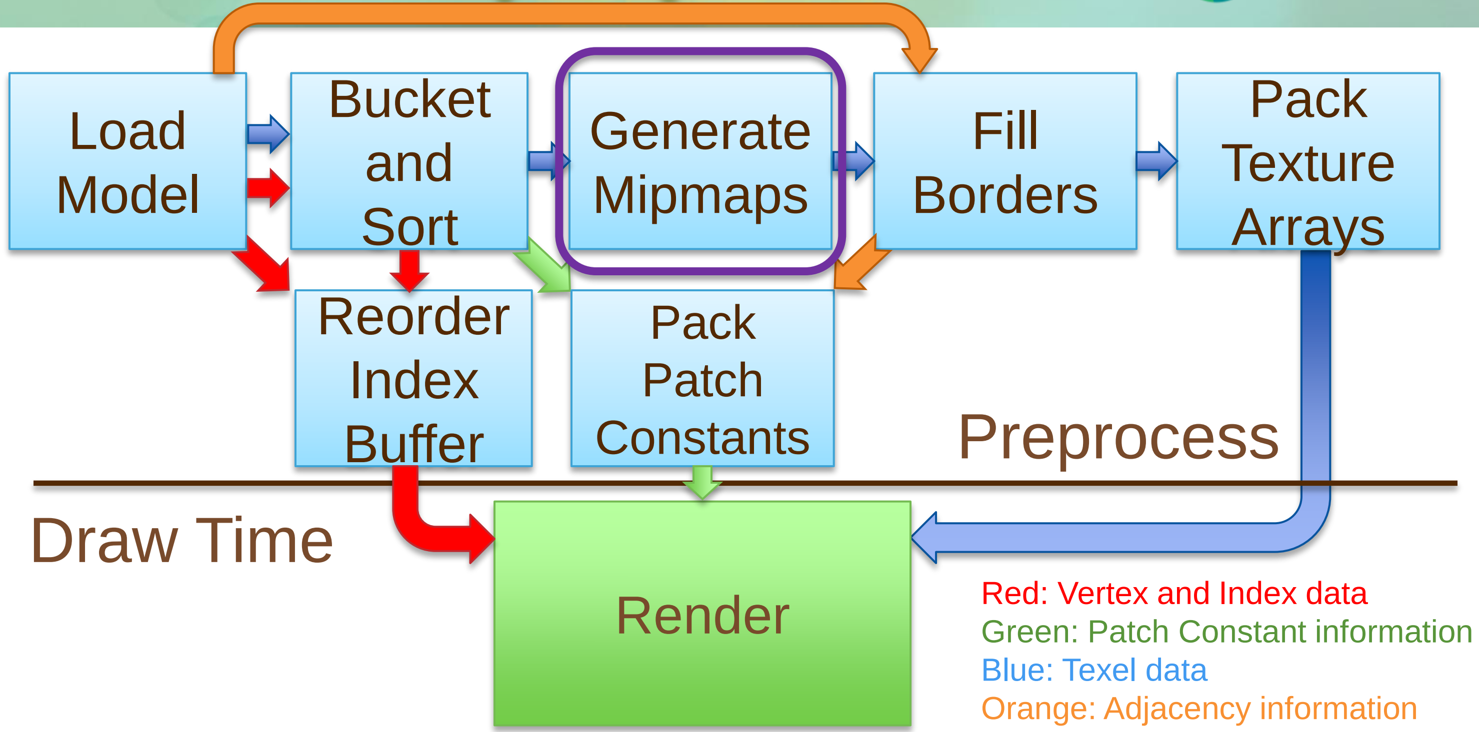
- Post Sort, we have 2!

IB(1:1):

IB(4:1):



Generate Mipmaps



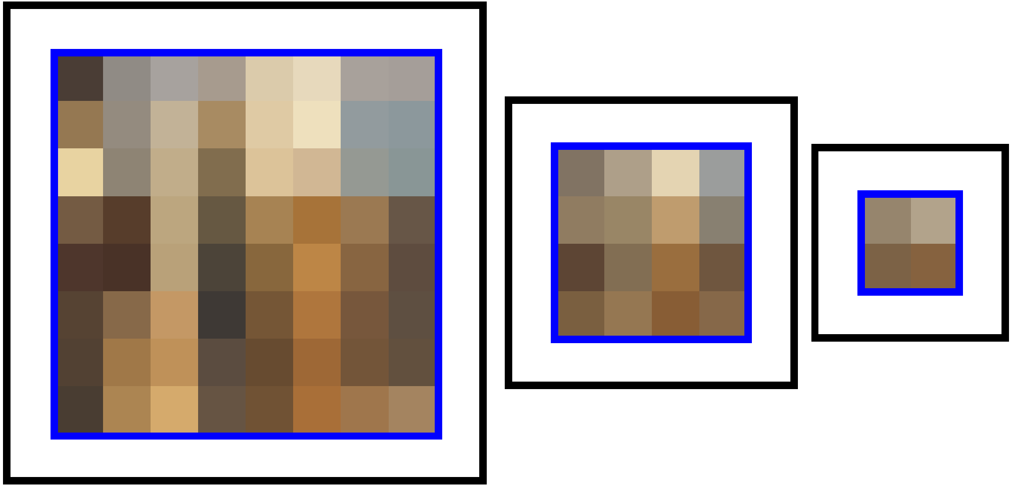
Generate Mipmaps

SIGGRAPH2011

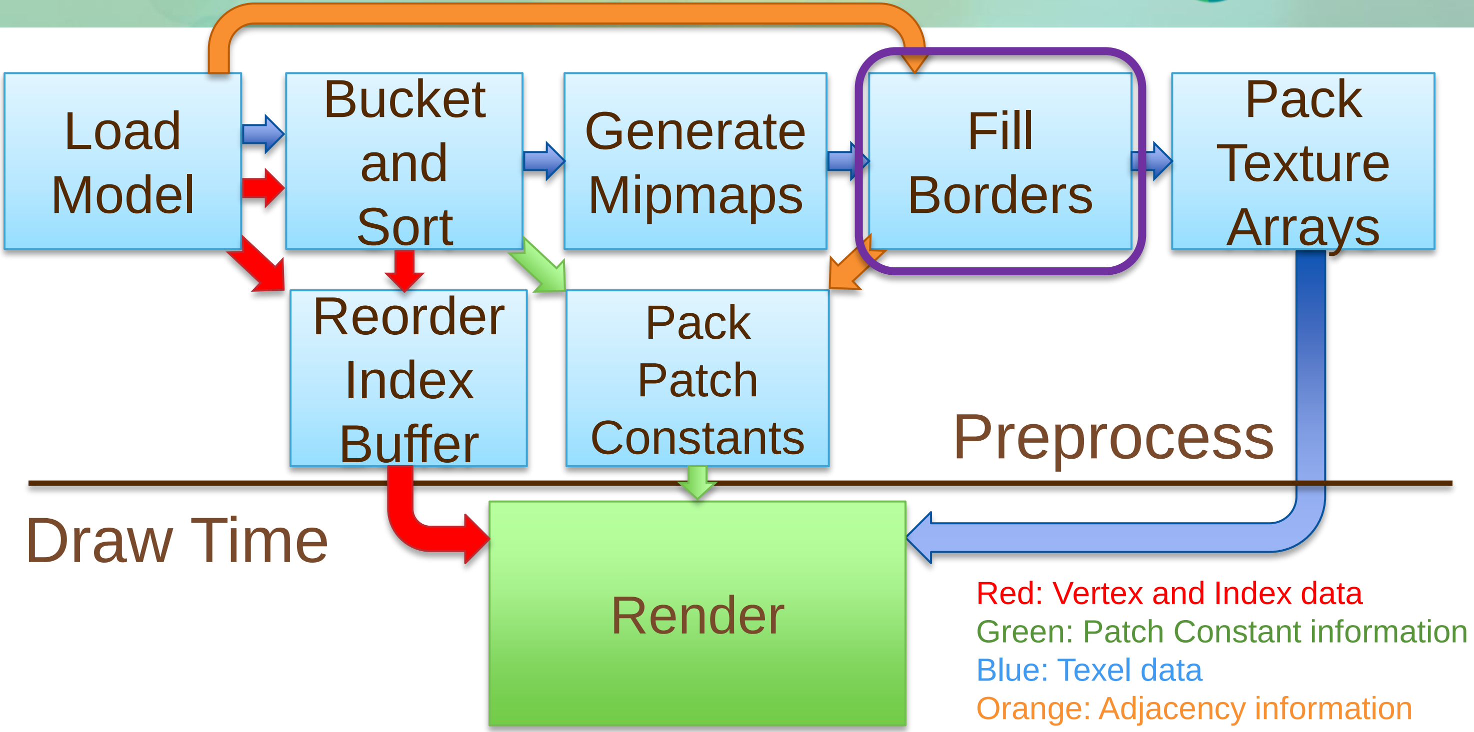
- Walk through surfaces, and generate a mipmap chain from native size to (MinFilterSize x MinFilterSize) for each surface.
  - Bilinear stops at 2x2, 8xAniso stops at 8x8
- Ptex data files do not guarantee complete mipmap chains—although the library can generate all levels for you—with pre-multiplied alpha.
- Mipmap chains stop to allow for unique pinning values in the corners

Generate Mipmaps

- Done for every surface, but only inside the surface—the border is not touched.



Fill Borders

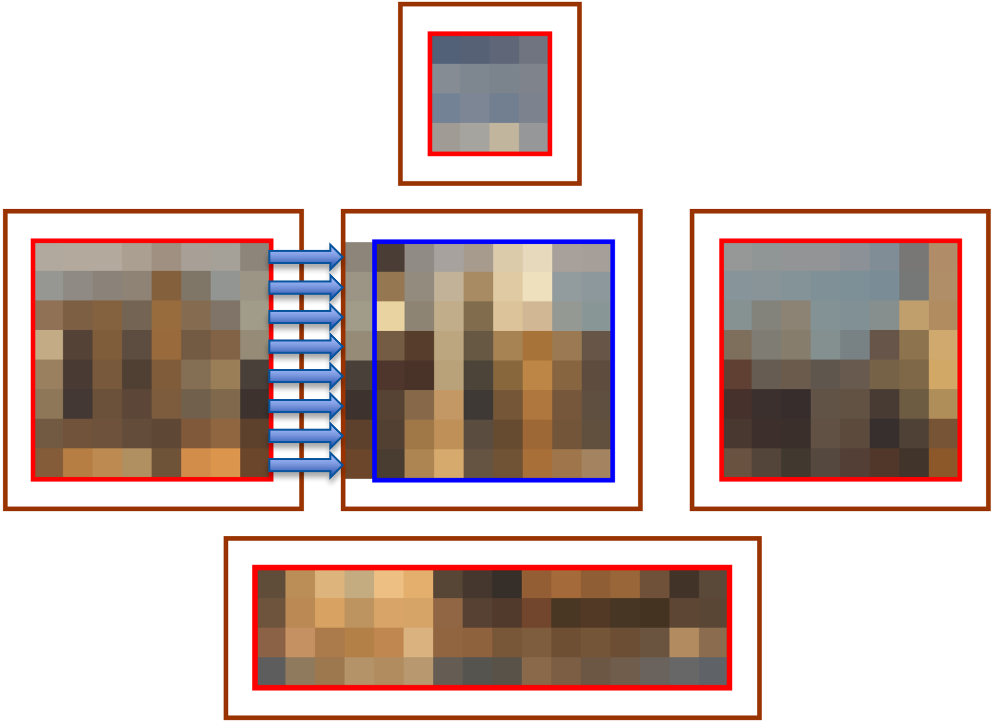


Fill Borders

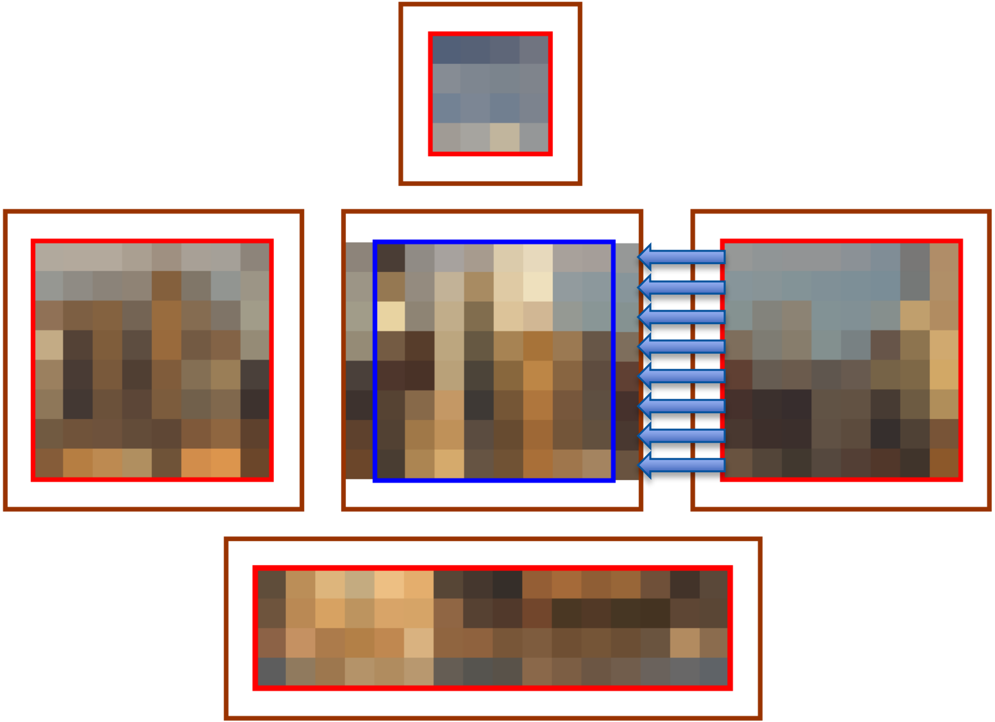
SIGGRAPH2011

- Copy neighbor texels into border area of this surface's mip level
  - Match source and destination number of pixels when possible
- Bordered textures are the heart of the logical realtime ptex solution
- Allows 1-2 texture lookups per ptex sample request
  - 1 if not performing tween-mip-level interpolation, 2 otherwise

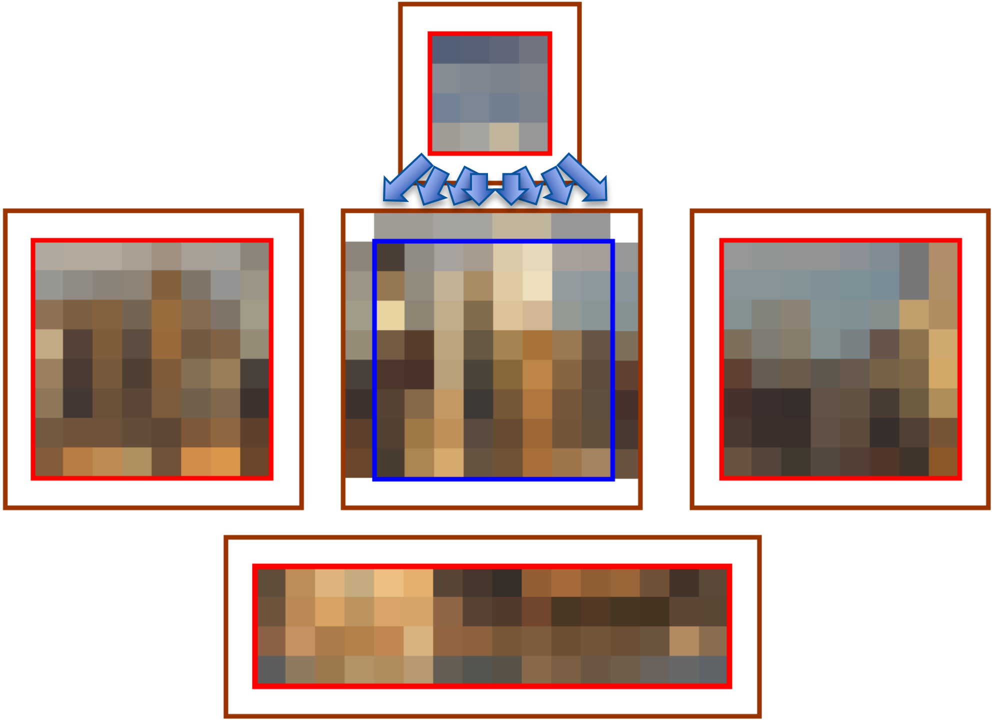
Fill Borders



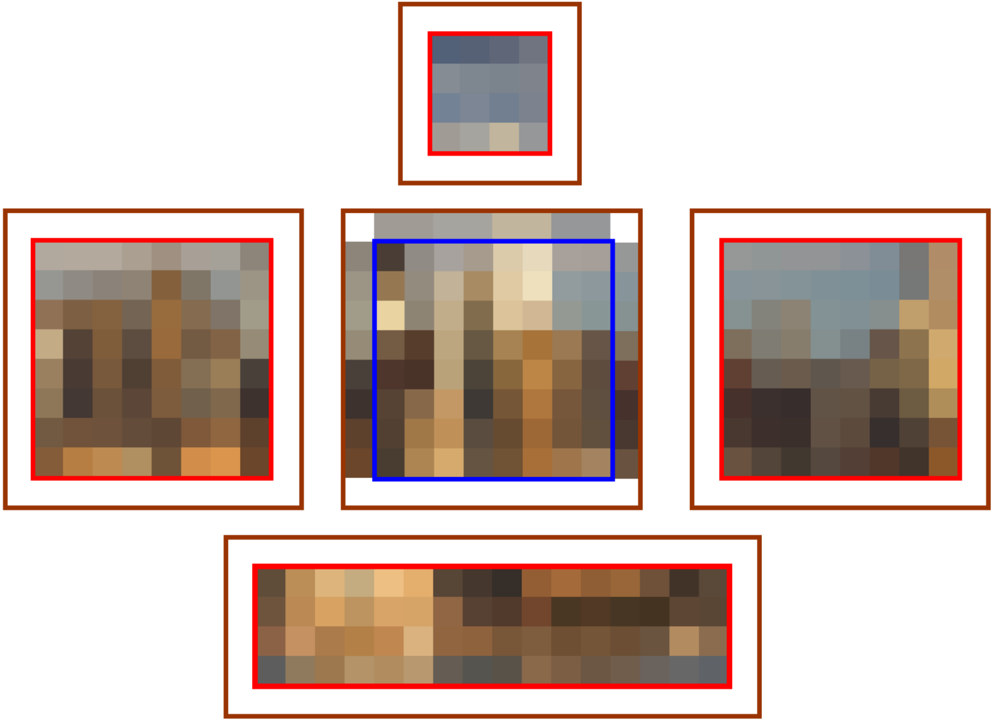
Fill Borders



Fill Borders

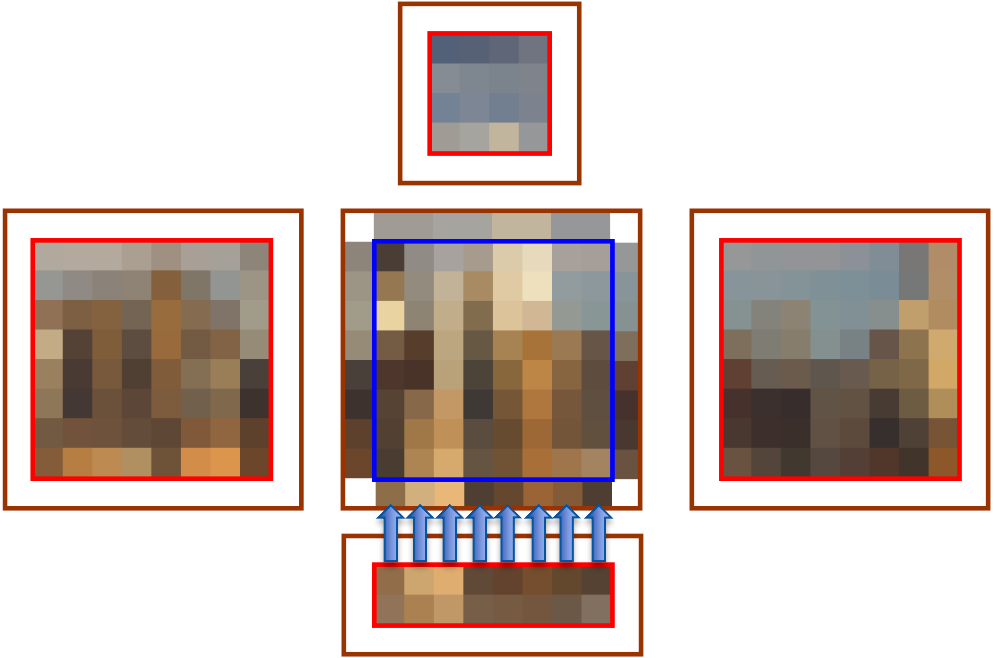


Fill Borders



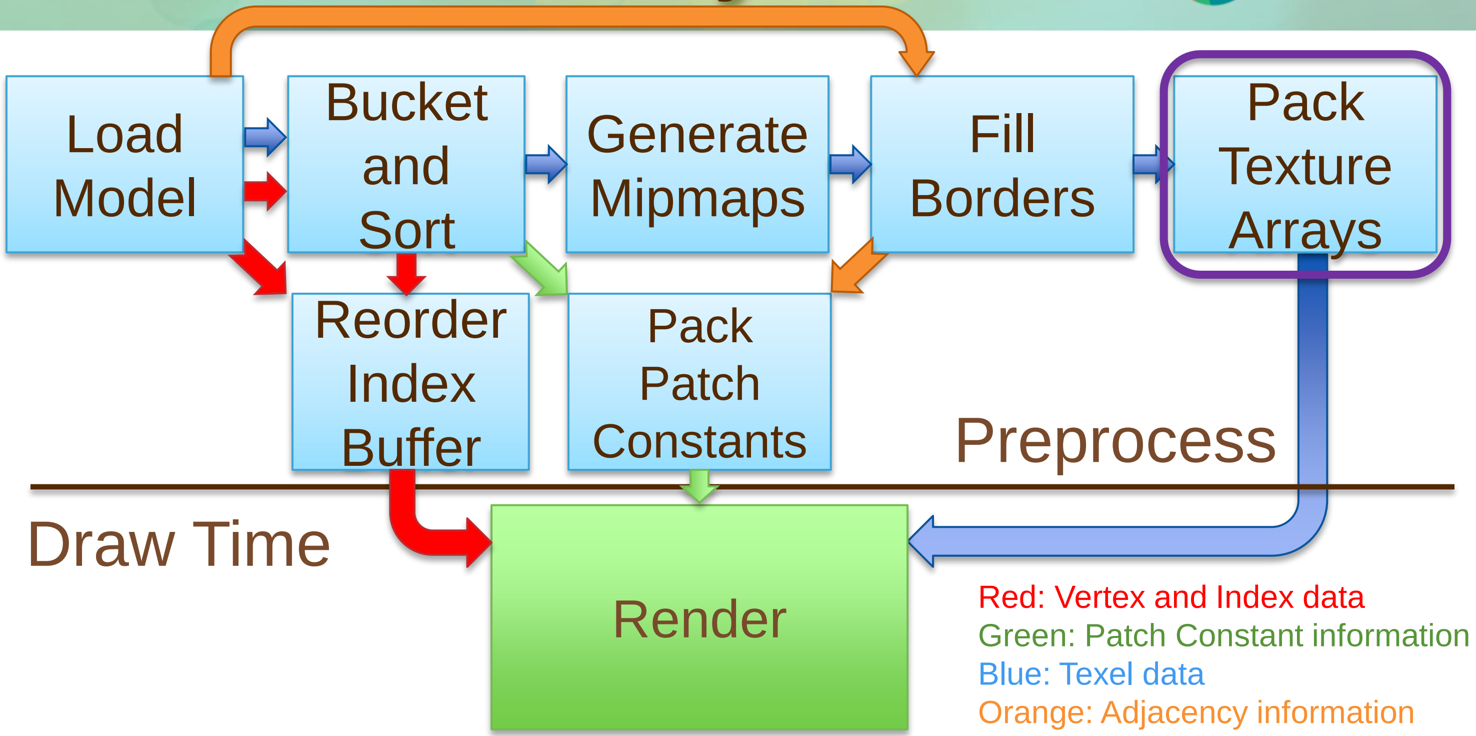
Fill Borders

SIGGRAPH2011



Pack Texture Arrays

SIGGRAPH2011



Texture Arrays

SIGGRAPH2011

- Like 3D / Volume Textures, except:
  - No filtering between 2D slices
  - Only X and Y decrease with mipmap level (Z doesn't)
  - Z indexed by integer index, not [0,1]
    - E.g. (0.5, 0.5, 4) would be (0.5, 0.5) from the 5th slice
- API Support
  - Direct3D 10+: Texture2DArray
  - OpenGL 3.0+: GL_TEXTURE_2D_ARRAY

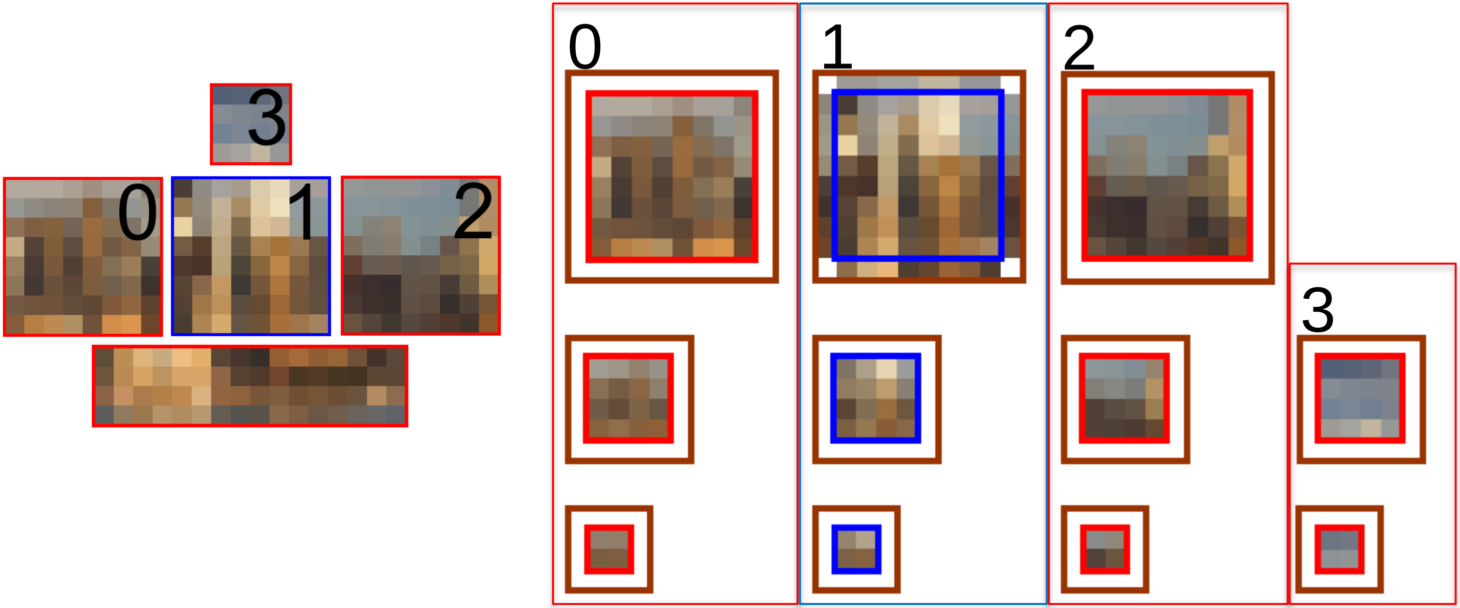
Pack Texture Arrays

SIGGRAPH2011

- Copy all generated data into Texture2DArray
- Each Texture2DArray represents a single mipmap level
  - Texture2DArrays present a view of the data that is efficient for GPU layout
  - Logical Textures cut across the same page index of every Texture2DArray

Pack Texture Arrays

Logical Texture Layout

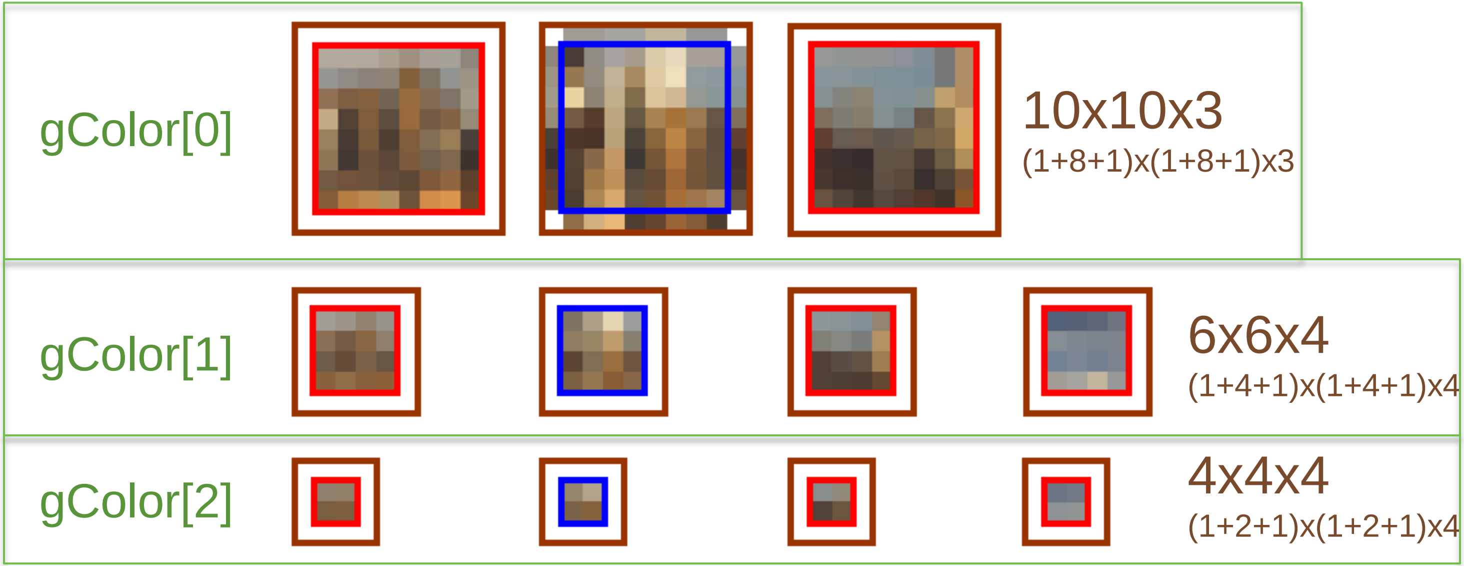


Pack Texture Arrays

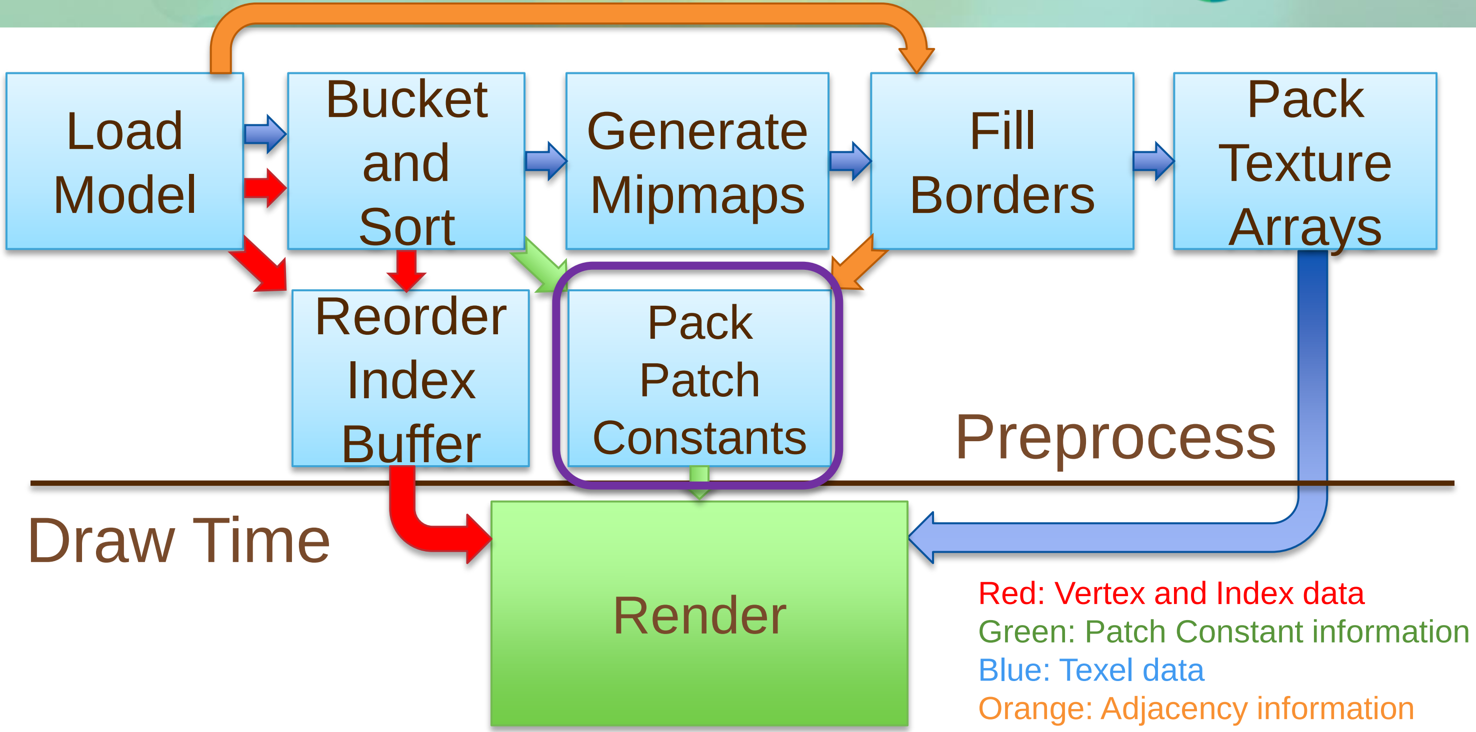
SIGGRAPH2011

GPU Layout

Texture2DArray



Pack Patch Constants

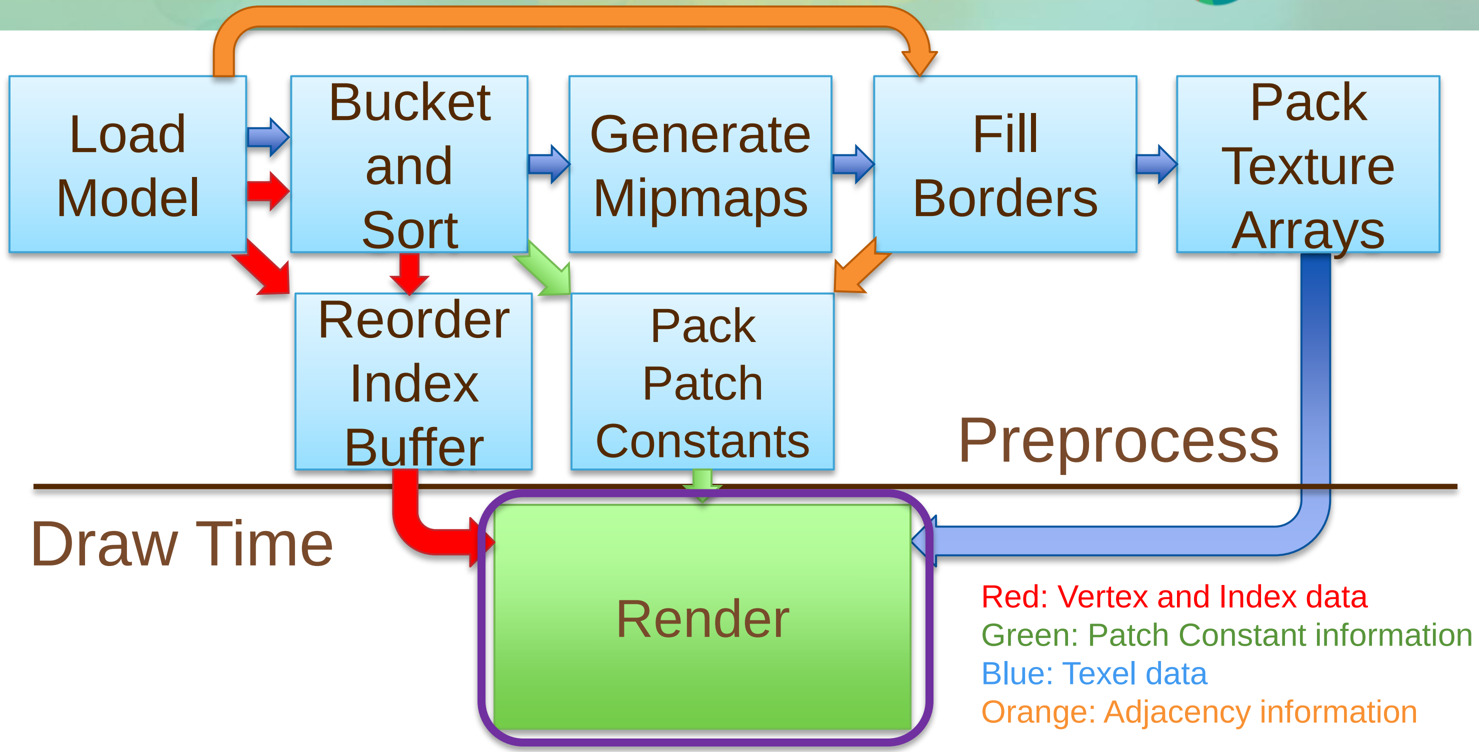


Pack Patch Constants

SIGGRAPH2011

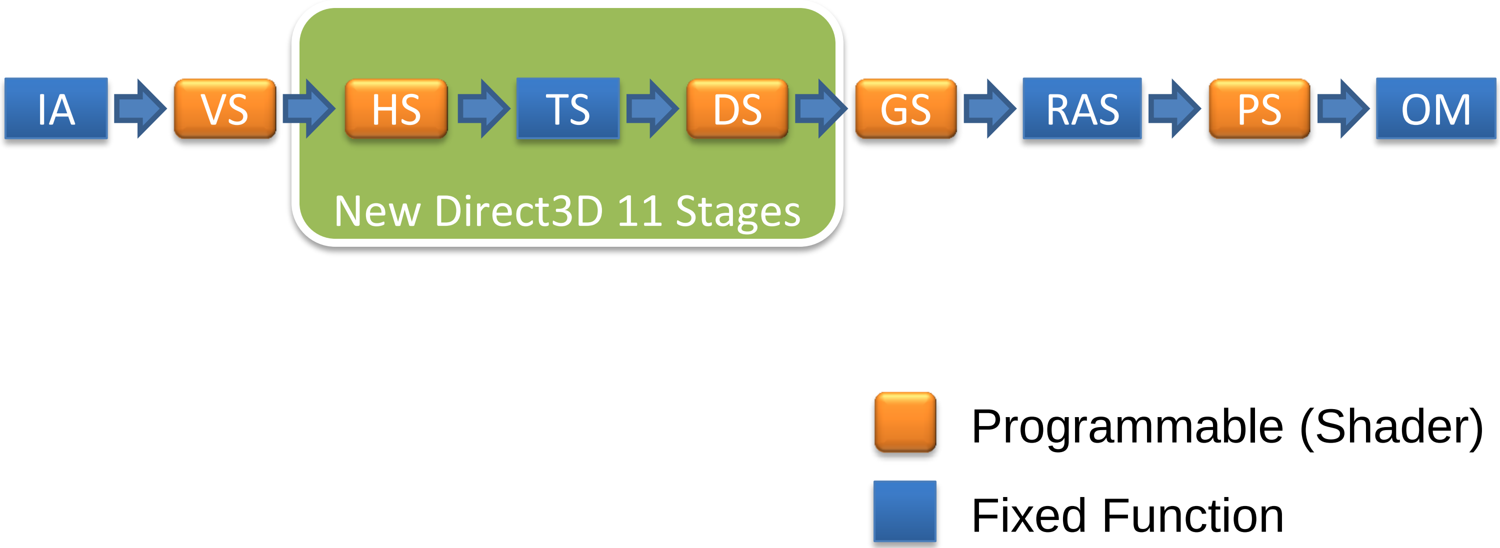
- Each primitive has a “PatchInfo” struct:
  - TextureId – which array slice contains our data
  - TopMipLevel – the index of the top-most mipmap level for this texture
  - FlipUVs – whether or not to flip UVs, allows 1:2 and 2:1 to be grouped into same bucket
  - MaxMipLevels – Maximum mipmap level for each edge

Fill Borders



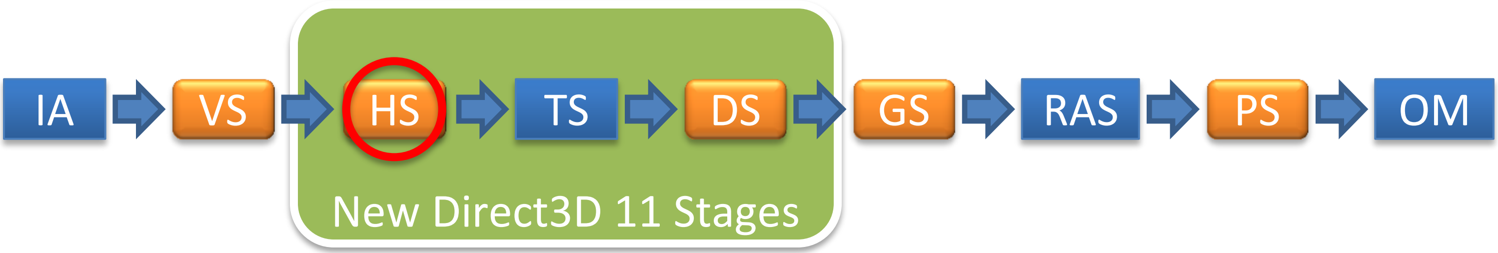
Render

- Brief Recap of D3D11 Pipe stages



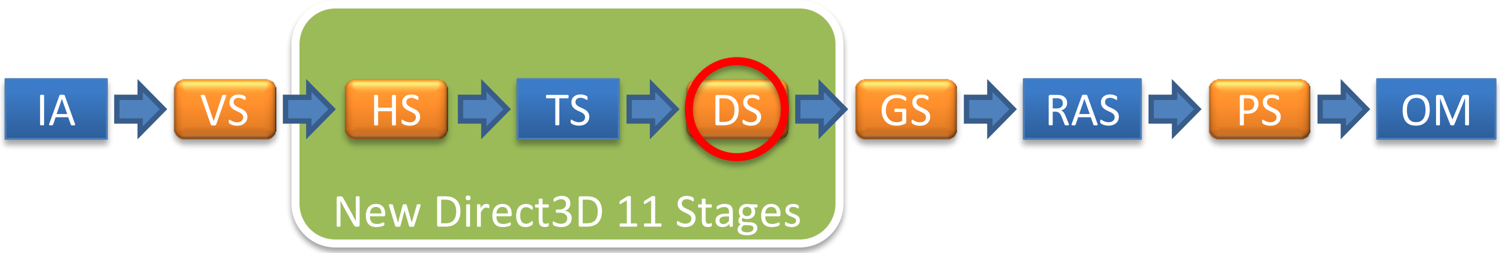
Render

- In the Hull Shader
  - Store pre-expansion PrimitiveID to output control points
  - This is used everywhere to determine which set of Patch Constants are owned by the currently running thread (in Domain, Geometry or Pixel Shaders)



Render

- In the Domain Shader
  - Vertices belonging to a quad meshes are evaluated with a domain location, which is (0,0)-(1,1) for each patch
  - Use this value to store our UV location



Render

SIGGRAPH2011

- Texture lookups in Domain or Pixel Shader are replaced with a “ptex” sample function.
  - Determines which logical texture to work from
  - Compute mipmap level(s) to access
  - Scale and bias computed (u,v) by mipmap size
  - Lookup texels, return weighted average

Texture Lookup Shader Code

SIGGRAPH2011

- Traditional (D3D11)

```
return
   gTxDiffuse.Sample(
          gSampler,
          I.fTextureUV );
```

- Ptex

```
return
  ptex( gTxDiffuse,
        gSampler,
        I.uPrimitiveId,
        I.fTextureUV );
```

Complex logic hidden in single function call

Questions?

SIGGRAPH2011

- jmcdonald at nvidia dot com
- Brent dot Burley at disneyanimation dot com
- http://ptex.us/
- http://groups.google.com/group/ptex
- Or visit our studio session
  - Monday, 8 August @ 4:30 PM – 5:00 PM
  - The Studio / West Building, Ballroom A

Additional Considerations

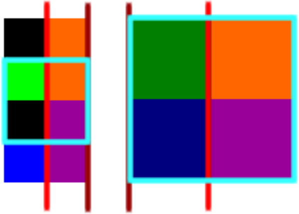
SIGGRAPH2011

- Filtering across edges with differing numbers of pixels
- Filtering across corners

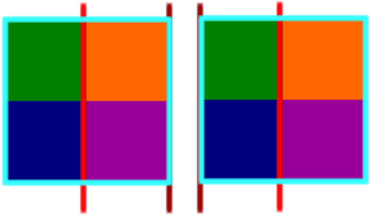
Filtering across edges

SIGGRAPH2011

Filtering across edges

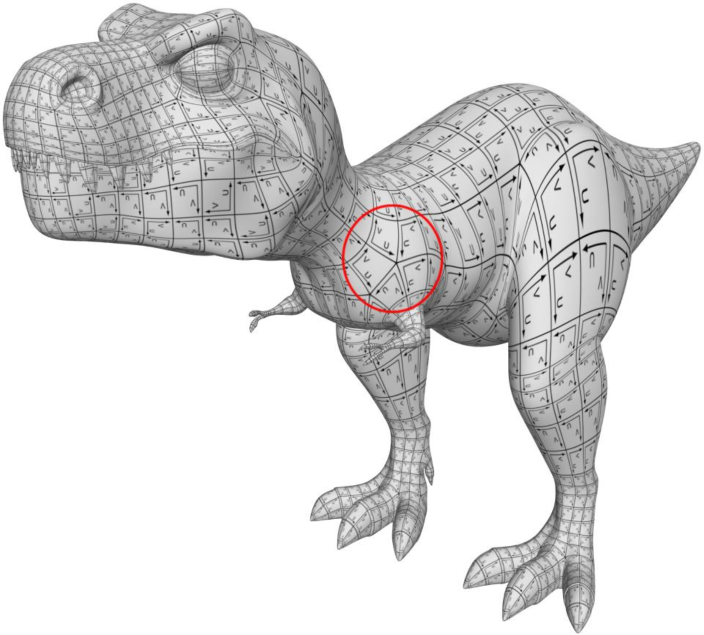


Filtering across edges



Filtering across corners

- Corners with valence > 4 cannot be exactly matched with a bilerp (4 samples)



Filtering across corners

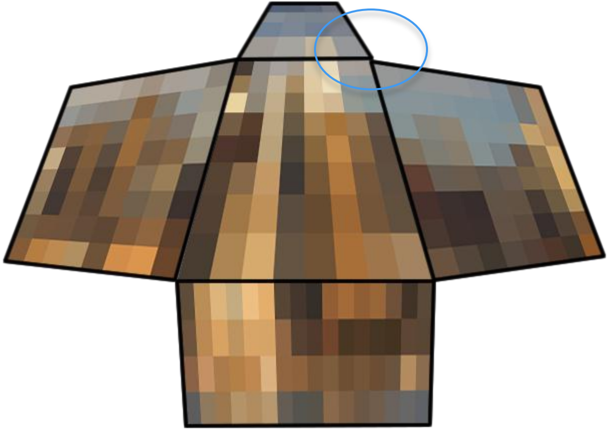
SIGGRAPH2011

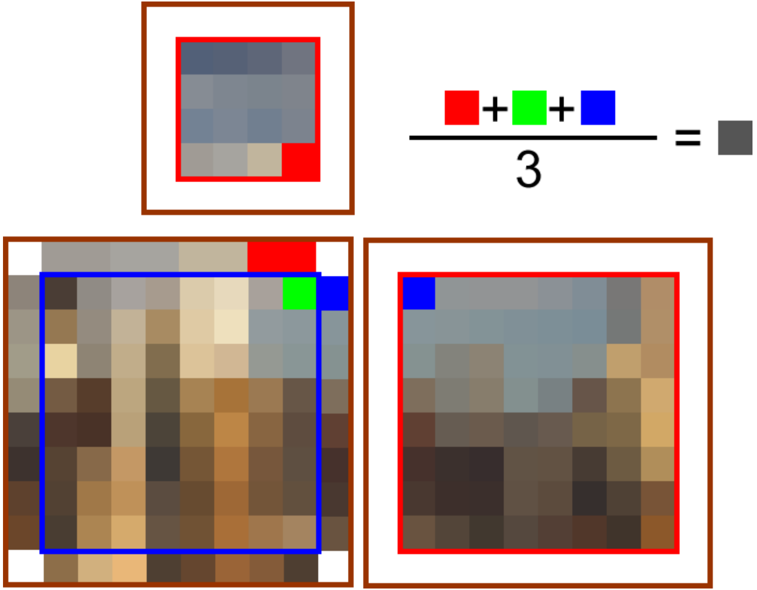
- The solution is a bit more involved.
  - First, walk the mesh and determine which corners are shared
  - For each shared group, determine the correct value for when we’re exactly at that corner. E.g. Simple Average.
  - Then, modify every mipmap level of every surface of that group s.t. the shared corner has the same value
  - When you’re in the corner, everyone will perform the same lookup—regardless of mipmap level—and continuity prevails

Filtering across corners

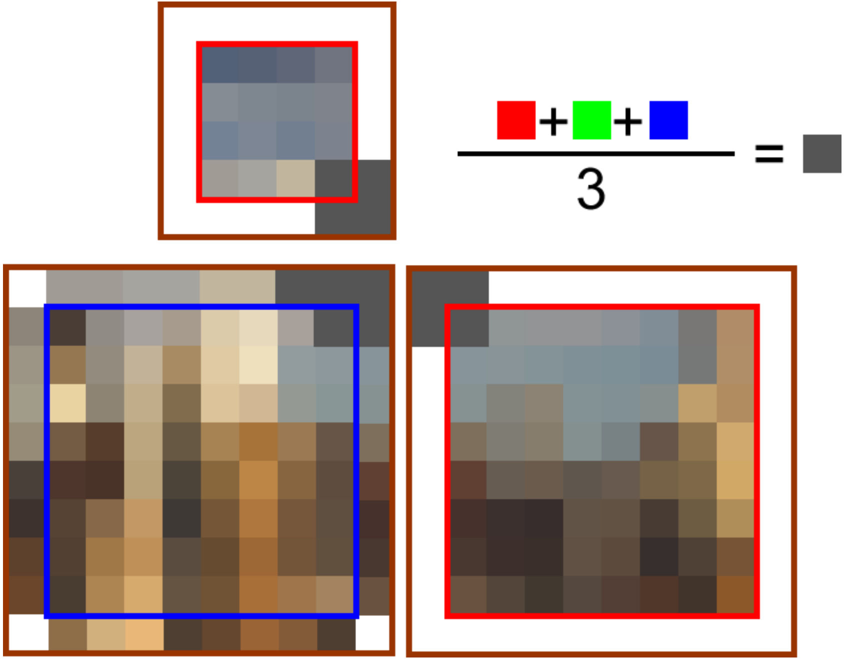
SIGGRAPH2011

- For Realtime Ptex, we apply pinning to all corners, regardless of valence.





Pinned Corners



Questions redux?

- jmcdonald at nvidia dot com
- Brent dot Burley at disneyanimation dot com
- http://ptex.us/
- http://groups.google.com/group/ptex
- Or visit our studio session
  - Monday, 8 August @ 4:30 PM – 5:00 PM
  - The Studio / West Building, Ballroom A