Per-Face Texture Mapping for Real-time Rendering

John McDonald, Jr
NVIDIA Corporation
jmcdonald@nvidia.com

Brent Burley
Walt Disney Animation Studios
Brent.Burley@disney.com

1. Introduction

Texture unwrapping is a nearly ubiquitous step in texture mapping, and serves two important purposes: (1) it indicates where to sample texture across each face and (2) it allows artistic freedom to "trade" texels from low importance areas to high importance areas. Unfortunately, texture unwrapping is time-consuming and automated tools generally produce inferior results to those of a skilled artist. Additionally, even the best unwraps suffer from seams, which cause lighting discontinuities at best and surface discontinuities at worst.

Ptex, proposed in [Burley and Lacewell 2008], assigns a separate texture per face, skipping the texture unwrap entirely. The technique uses adjacency data to filter across faces, removing visible seams. We propose a technique to render Ptex datasets—intended for offline rendering—at interactive framerates using commodity Direct3D 11 capable hardware.

2. Algorithm

As with offline Ptex, meshes must be quad-based, textures must be power-of-two sized (though not necessarily square), and patch adjacency data must be provided. The proposed real-time Ptex algorithm involves a preprocess step as well as straightforward shader modifications to texture lookups.

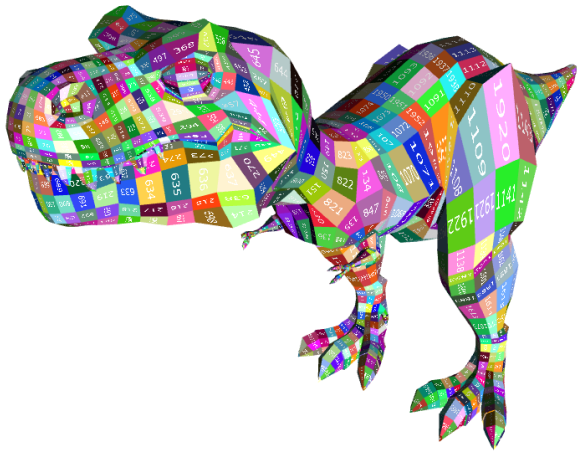
Textures are first grouped by aspect ratio. Within each group, textures are sorted by decreasing size and assigned monotonically increasing IDs in this sorted order. Textures are stored in a manual mip-chain using one Texture2DArray per unique resolution, and each array is indexed by texture ID. Data from adjacent patches are sampled into a fixed size border around each texture. The border size can be chosen according to desired filter quality.

A "patch constant" structure stores information about each patch's texture data: base array ID, texture ID, and maximum mip-level.

During shading, we:

1. Use *SV_PrimitiveID* [Microsoft 2011] to index into an array of patch constant structures to determine the ptex parameter set of the current patch.
2. Compute the desired mip-level and select the corresponding texture array.
3. Adjust *SV_DomainLocation* [Microsoft 2011] to account for the fixed size border. The scale and offset are pre-computed per unique texture resolution and are accessed from the shader.
4. Sample the specified texture array entry using desired hardware texture filter.
5. Repeat steps 2-4 once more if mip-map interpolation is desired.

Figure 1. Dinosaur with each face textured from unique textures.



3. Advantages

The texture unwrap step is skipped entirely. Texture seams will be invisible along edges and at vertices with a valence of four. Additionally, each face can have a texture size completely independent of its neighbors—and this can be changed at any point in a model's lifetime without having to go through another expensive unwrap operation. Unlike most unwrapping schemes, no texture space is wasted, and arbitrary filter widths may be used without sourcing invalid texture data.

4. Results

We've applied this method of rendering to several test meshes, including the dinosaur pictured above. The dinosaur has 5,812 faces—each individually textured. The working set of 80 mega-texels renders in less than one half of one millisecond on a GTX 460 1 Gb.

References

BURLEY, B., AND LACEWELL, D. 2008. Ptex: Per-Face Texture Mapping for Production Rendering. In *Eurographics Symposium on Rendering 2008*, 1155–1164.

MICROSOFT, INC. 2011. Semantics (DirectX HLSL), http://msdn.microsoft.com/en-us/library/bb509647%28v=vs.85%29.aspx

SIGGRAPH 2011, Vancouver, British Columbia, Canada, August 7 – 11, 2011.
ISBN 978-1-4503-0921-9/11/0008