Per-Face Textures

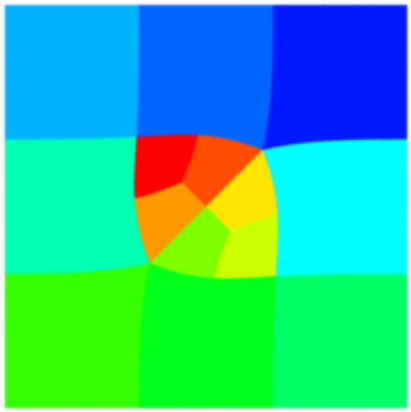
Introduction

RenderMan Pro Server supports per-face texture mapping, also known as Ptex. Ptex is a texture mapping system that uses the intrinsic per-face UVs of a polygonal mesh; no explicit texture parameterization is required. Ptex textures are stored along with adjacency data in a single file per primitive. Ptex uses the adjacency data to perform seamless anisotropic filtering of multi-resolution textures across surfaces of arbitrary topology. Additional technical information about Ptex textures and filtering can be found at http://ptex.us/overview.html

To help kick start the process of authoring Ptex files, this appnote provides details of shaders and utility programs that can be used to bake shaded results into Ptex textures. A parameter, __faceindex, which labels each face on the geometry, is required by theptexture() shading function and we provide a RIF called RifSubdivFaceIndex that will label geometry with the appropiate index. We also provide a program called ptxmake that will take a pointcloud baked in a specific way and convert the data to a Ptex file.

Geometric Primitive Support

One of the required parameters for the ptexture() call is __faceindex. This index is a facevarying, postive integer at each quad face of the surface, usually starting at 0. For Catmull-Clark surfaces, non-quad faces are numbered at each vertex, and the first level of subdivision of the non-quad faces will generate N-subfaces, where N is the number of vertices. For quad faces, the same faceindex` value is used on each vertex. An example of the faceindex layout is shown below. Note that the __faceindex follows special subdivision rules that will ensure the value is constant across quad-faces and that the shader parameter will be of *varying* detail. For Loop surfaces, only triangular faces are supported; each vertex of the face should be numbered with the same __faceindex value. A RIF called RifSubdivFaceIndex.so is included for convenience that will take a subdivision surface without a faceindex and create the appropriate __faceindex values at each vertex.



Here is the RIB file example utilizing Ptex:

```
##RenderMan RIBversion 3.03

Projection "perspective" "fov" [45]
Format 256 256 1.0
Option "ribparse" "string varsubst" ["$"]

Display "test.tif" "tiff" "rgba"
WorldBegin
Attribute "visibility" "camera" [1]
Translate 0 0 3

Pattern "PxrPtexture" "ptex" "string filename" "./plateface.ptx"
Bxdf "PxrDiffuse" "mydiffuse" "reference color diffuseColor" ["ptex:resultRGB"]
SubdivisionMesh "catmull-clark"
[4 4 4 4 3 3 4 4 4 4]
[4 5 1 0 5 6 2 1 6 7
 3 2 8 9 5 4 9 10 5
10  6 5 10 11 7 6
12 13 9 8 13 14 10 9
14 15 11 10]

["interpolateboundary" "smoothtriangles"] [0 0 1 0] [1] []
"P" [-1.0 -1.0 0
    -0.3333333333333 -1.0 0
    0.3333333333333 -1.0 0
    1.0 -1.0 0
    -1.0 -0.3333333333333 0
    -0.3333333333333 -0.3333333333333 0
     0.3333333333333 -0.3333333333333 0
     1.0 -0.3333333333333 0
     -1.0 0.3333333333333 0
     -0.3333333333333 0.3333333333333 0
      0.3333333333333 0.3333333333333 0
     1.0 0.3333333333333 0
     -1.0 1.0 0
     -0.3333333333333 1.0 0
     0.3333333333333 1.0 0
     1.0 1.0 0]

"facevarying float __faceindex" [6 6 6 6
7 7 7 7
8 8 8 8
9 9 9 9
0 1 2
3 4 5
10 10 10 10
11 11 11 11
12 12 12 12
13 13 13 13]
"constant string __handleid" ["${RMANREGRESS_TESTOBJDIR}/plate"]

WorldEnd
```

---

ptxmake

A new utility program called ptxmake has been created that will take data baked into a point cloud and convert it into a Ptex texture file. The usage is as follows:

```
ptxmake [options] ptcfile channelname ptxfile
```

ptxmake supports the following options:

-fedfile (fedfilename)	provide a face edge data file
-geometry (quad|tri)	provide the geometry of the baked data quad
-depth (byte|short|half|float)	bit depth of output image, the default is half

-splat (none|diffusion|smooth|area)	splattering mode, the default is diffusion
-newer (0|1)	convert if (and only if) the ptcfile is newer than the ptxfile
-verbose	enables verbose output
-outofcore	work out of core for large data
-v|-version	output version string

One of the key components of a Ptex file is the face and edge connectivity data of the geometry being textured. A RIF, calledRifFaceEdgeData, has been provided that takes a surface that has been tagged with __handleid and generates a file called:__handleid.fed. This fed file can then be passed to the ptxmake program. Note, a limitation of the RifFaceEdgeData plugin is that it can only be used while rendering with prman. It will not function with catrib or prman with the -catrib option. In addition to the connectivity data, the fedfile also encodes the geometry type, so the -geom flag does not need to be used with ptxmake, except in cases where no fedfile is present.

Using ptxmake

The ptxmake program is used to convert the point cloud data into a Ptex file. The usage for our dragon test case is:

```
ptxmake -depth half -fedfile dragon.fed dragon.ptc occlusion dragon.ptx
```

This takes the .fed file and the .ptc file generated during the bake pass and renders each face in the point cloud into a per-face texture in the .ptx file. This file can then be used to re-render the baked occlusion data. Notice, Ptex files support the half-float data format.

The utility program ptxinfo can be used to query data about the ptx file. In our test case ptxinfo -f dragon.ptx gives the following output:

```
meshType: quad
dataType: float16
numChannels: 1
alphaChannel: (none)
numFaces: 895
  hasFaceEdits: no
numMetaKeys: 0
face 0:  res: 7 7 (128 x 128)  adjface: 85 1 6 5  adjedge: 3 3 0 1  flags: (none)
face 1:  res: 7 7 (128 x 128)  adjface: 88 2 7 0  adjedge: 3 3 0 1  flags: (none)
face 2:  res: 7 7 (128 x 128)  adjface: 91 3 8 1  adjedge: 3 3 0 1  flags: (none)
face 3:  res: 7 7 (128 x 128)  adjface: 94 4 9 2  adjedge: 3 3 0 1  flags: constant
face 4:  res: 7 7 (128 x 128)  adjface: 97 5 10 3  adjedge: 3 3 0 1  flags: constant
face 5:  res: 7 7 (128 x 128)  adjface: 100 0 11 4  adjedge: 3 3 0 1  flags: (none)
face 6:  res: 7 7 (128 x 128)  adjface: 0 7 12 11  adjedge: 2 3 0 1  flags: (none)
...
```