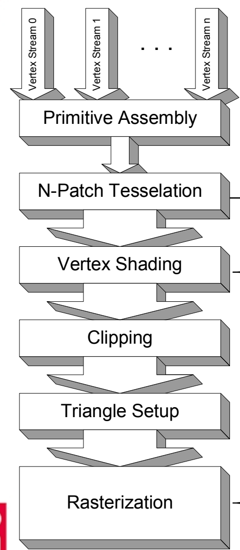
Unified Pixel Shading in the ATI R200

Mark Fowler
ATI 3D Hardware, Marlboro MA
(markf) @ati.com

Outline

- R200
  - N-Patch higher-order surface
  - Programmable geometry
  - Programmable pixel shaders
    - Unified instruction set
    - 6 textures and 16 blenders

R200 3D Pipeline



- Jittered FSAA & Multisample Buffers
- Fast Z clear, Z Compression, Hierarchical Z
- N-Patch Higher Order Surface
- Vertex Shaders
- Fixed-Function Transform
- Indexed Vertex Blending
- Point Sprites
- Pixel Shaders
- Orthogonal 3D Textures
- Quad 3D pipes
- Up to 16:1 Anisotropic Filtering
- Cubic Environment Mapping

N-Patches on R200

Triangular Bezier Patch

- Simple & Easy to use
- Compatible with existing data structures
- Extremely low impact to API
- Minimal effort to adapt existing 3D models
- Models compatible with HW without surface support
- Absolutely no software (driver) setup
- Fast in hardware

N-patches Buy Bandwidth

- In the past, we have used pixel and texel caching and compression schemes to minimize need to access memory.
- N-Patches are a way to do something similar for geometry.
- Geometry compression has been explored in the literature (see Deering), but the techniques usually rely on vertex-to-vertex coherence and thus require a lot of on-chip storage for decompression, not to mention the issue of settling on a standard way to do the compression. Textures were easy by comparison
- N-Patches take geometry in an already-consumable triangle form and "smoothify geometry automagically."

Motivated by Characters

- The major application of N-Patches is character rendering
- Majority of polygons in modern games are in the characters
- A single character instance can be considered to be at a single LOD
- Surface tessellation can be used in combination with skinning, tweening, etc.

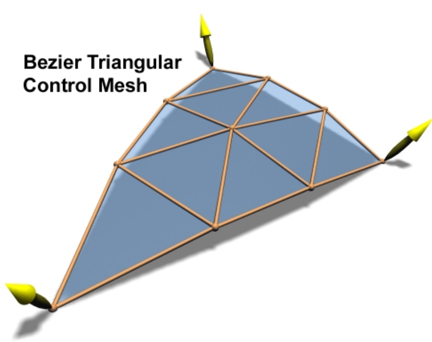
OpenGL N-Patch API

- PN Triangles Extension:
  - Subdivision Level
    - Takes an integer *n*
    - *n* new points are added along each edge of triangle
  - Normal Interpolation Type
    - LINEAR or QUADRATIC
- Normals must be provided in vertices
- All existing triangle drawing commands are still valid

Control Mesh

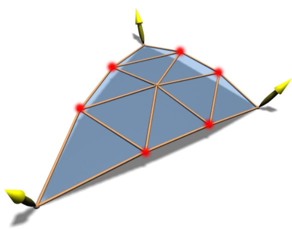
N-Patch is an interpolating triangular cubic Bezier surface

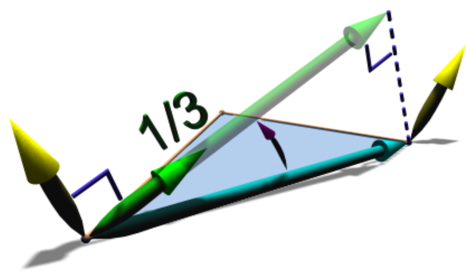
- A 10-point control mesh is needed to tessellate this surface.
- The control mesh is derived from 3 point/normal pairs (a triangle).



Boundary Points

- 3 original vertices *are* 3 of the control points
- 6 points, called **boundary points** (don't necessarily lie on boundary), are derived:
  - Computed by projecting the edge vector into the plane defined by the normal
  - The vector is then scaled by 1/3



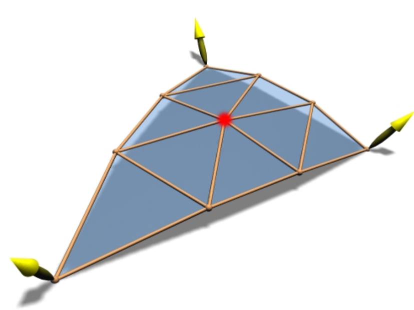


3D Application Research Group - ATI Research 9

Interior Control Point

The **interior** point is then calculated from the other 9 control points (original 3 vertices and 6 border points):

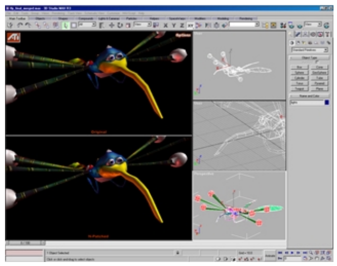
Interior = (SumOfBorderPoints/4.0) - (SumOfOriginalVertices/6.0)

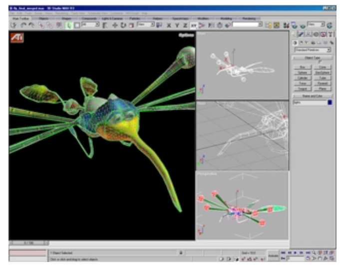


3D Application Research Group - ATI Research 10

3D Studio Max Plug-In

- Allows artists to stay "in tool" to preview the look of N-Patches as they model
- Tessellates in real-time on R200.





N-Patch Demo

Indexed Vertex Blending

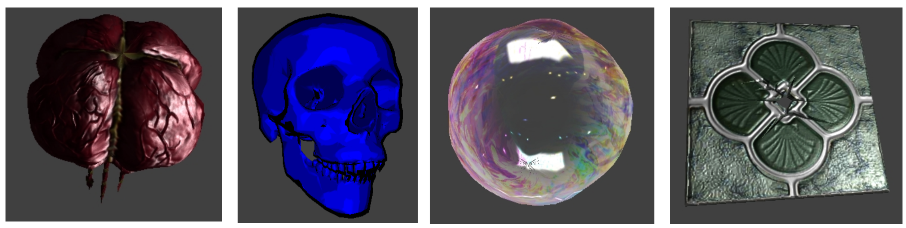
- Also known as "Matrix Palette Skinning"
- More Matrices in Fixed Function than you could express in a Vertex Shader
- On R200 you get 29 Matrices

R200 Programmable Geometry

- Full DirectX 8.0 Vertex Shading (v 1.1)
  - Programs up to 128 instructions
  - 96 vectors of constant store
  - 12 temporary data registers
  - Indexed access to constant store
  - Full precision reciprocal and reciprocal square root

Other Shaders

- Custom environment mapping
- BRDFs
- Anisotropic lighting models
- Procedural displacement
- Non-photorealistic Rendering
- Anything you can dream up!



Vertex Shader Demo

R200 Pixel Shading

- Unified Instruction Set
- 6 textures and 16 instructions
- High-precision internal representation
- Will be fully exposed in DX8.1 (Shader Version 1.4)

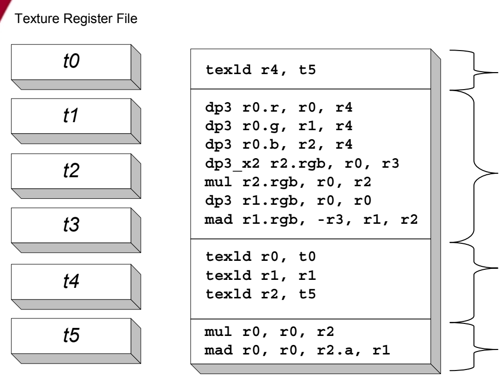
3D Application Research Group - ATI Research

Pixel Shader Goals

- Shared syntax with vertex shaders
- Simple but powerful instruction set
- Extensible for future improvements

3D Application Research Group - ATI Research 18

Pixel Shader Structure



- Texture registers (t_n) are pre-initialized according to `texture` state.
- Optional Sampling
- Address Shader
  - Up to 8 instructions
- Optional Sampling
  - aka “dependent reads”
- Color Shader
  - Up to 8 instructions

3D Application Research Group - ATI Research 19

R200 Pixel Shader Instructions

```
• add   d, s0, s1        // sum
• sub   d, s0, s1        // difference
• mul   d, s0, s1        // modulate
• mad   d, s0, s1, s2    // s0 + s1*s2
• lrp   d, s0, s1, s2    // s2 + s0*(s1-s2)
• mov   d, s0            // d = s0
• cnd   d, s0, s1, s2    // d = (s2 > 0.5) ? s0 : s1
• cmp   d, s0, s1, s2    // d = (s2 > 0) ? s0 : s1
• dp3   d, s0, s1        // s0 dot s1 replicated to rgba
• dp4   d, s0, s1        // s0 dot s1 replicated to rgba
• d2add d, s0, s1, s2    // s0.r*s1.r + s0.g*s1.g + s2.b
```

3D Application Research Group - ATI Research

Pixel Shading Sample 1

- Per-pixel N·L for four lights

```
ps.1.4
texld r0, t0              ; Sample the bump map
texld r1, t1              ; Sample the base map
texld r2, t2              ; Normalize L0
texld r3, t3              ; Normalize L1
texld r4, t4              ; Normalize L2
texld r5, t5              ; Normalize L3
; ----------------------- end of free instructions
dp3 r2, r0, r2            ; N.L0
dp3 r3, r0, r3            ; N.L1
dp3 r4, r0, r4            ; N.L2
dp3 r5, r0, r5            ; N.L3
; ----------------------- end of address shader
phase
; ----------------------- don't do any dependent reads
mul r0, r2, r1            ; N.L0 * base
mad_sat r0, r0, r3, r1    ; (N.L0 + N.L1) * base
mad_sat r0, r0, r4, r1    ; (N.L0 + N.L1 + N.L2) * base
mad_sat r0, r0, r5, r1    ; (N.L0 + N.L1 + N.L2 + N.L3) * base
```


Pixel Shading Sample 2

- Per-pixel N·H used to index into an exponential map to do per-pixel $(N\cdot H)^k$ for four lights

```
ps.1.4
texld r0, t0              ; Sample the bump map
texld r2, t2              ; Normalize H0
texld r3, t3              ; Normalize H1
texld r4, t4              ; Normalize H2
texld r5, t5              ; Normalize H3
; ----------------------- end of free instructions
dp3 r2.r, r0, r2          ; N.H0
dp3 r3.r, r0, r3          ; N.H1
dp3 r4.r, r0, r4          ; N.H2
dp3 r5.r, r0, r5          ; N.H3
; ----------------------- end of address shader
phase
texld r1, t1              ; Sample the base map
texld r2, r2              ; (N.H0)^k
texld r3, r3              ; (N.H1)^k
texld r4, r4              ; (N.H2)^k
texld r5, r5              ; (N.H3)^k
; ----------------------- and of dependent reads to raise N.Hn to a power
mul     r0, r2, r1.a      ; ((N.H0)^k) * gloss
mad_sat r0, r0, r3, r1.a ; ((N.H0)^k + (N.H1)^k) * gloss
mad_sat r0, r0, r4, r1.a ; ((N.H0)^k + (N.H1)^k + (N.H2)^k) * gloss
mad_sat r0, r0, r5, r1.a ; ((N.H0)^k + (N.H1)^k + (N.H2)^k + (N.H3)^k) * gloss
```

Pixel Shading Sample 3

- Interpolation of 3x3 basis matrix for per-pixel bumped reflection indexing into a cubic environment mapping

```
; Tangent space to cube map space transformation computations
texcrd  r0, t0            ; 1st row of 3x3 basis matrix
texcrd  r1, t1            ; 2nd row of 3x3 basis matrix
texcrd  r2, t2            ; 3rd row of 3x3 basis matrix
texcrd  r3, t3            ; Eye vector
texld   r4, t5            ; sample normal map
; ------------------------ end of free instructions
dp3 r0.r, r0, r4          ; 1st row of matrix multiply
dp3 r0.g, r1, r4          ; 2nd row of matrix multiply
dp3 r0.b, r2, r4          ; 3rd row of matrix multiply
dp3_x2 r2.rgb, r0, r3     ; 2 * (N dot Eye)
mul r2.rgb, r0, r2        ; 2 * N * (N dot Eye)
dp3 r1.rgb, r0, r0        ; N dot N
mad r1.rgb, -r3, r1, r2   ; 2 * N * (N dot Eye) - Eye * (N dot N)
; -------------- dependent reads
phase
texld r0, r0              ; sample diffuse cubic env map (m1)
texld r1, r1              ; sample specular cubic env map
texld r2, t5              ; sample the base map (shininess in alpha)
; -------------- color shader
mul r0, r0, r2            ; diffuse * base
mad r0, r0, r2.a, t1      ; (diffuse * base) + (spec * gloss)
```

3D Application Research Group - ATI Research

Pixel Shader Demo

3D Application Research Group - ATI Research

24