PixelFlow
Marc Olano
SGI
Shading

Compute base color and light interaction

Special-purpose shading language

- Shade Trees
  [Cook 84]
- Pixel Stream Editor
  [Perlin 85]
- RenderMan
  [Hanrahan and Lawson 90, Upstill 90]
Interactive Shading

Pixel-Planes 5 [Fuchs, et al. 1985]
- Low-level language, hard to use [Rhoades, et al. 92]

PixelFlow [Molnar, et al. 91]
- pfman
  - High-level language, based on RenderMan [Olano 98]
- API based on OpenGL [Leech 98]

Others covered later...
Hardware Requirements

Programmability
Memory
Tons of arithmetic
Organization

Introduction

Abstract Pipeline

pfman

Hardware and real-life

Conclusions
Machine complexity

Graphics machines are complex
User does not want to know

• How machine does what it does
• Dozens of machine-specific differences

Answer:
• Simple model of machine
• High-level language for procedures
Abstract Pipeline

- Model
- Transform
- Primitive
- Interpolate
- Shade
- Atmospheric
- Image

User’s mental model
Hide details
Device-independent
Procedural stages
Allow partial coverage
Model

- Newell 75
- Hedelman 84
- Amburn 86
- Cook 87
- Green 88
Transformation

Procedures
- Fleischer 88
- Upstill 90

Techniques
- Barr 84
- Sederberg 86
Primitive

- Whitted 82
- Cook 87
- Fleischer 87
- Perlin 89
- Upstill 90
Interpolation

Interpolate

Procedures
- Ebert 94
- Neider 93
Shading

- Model
- Transform
- Primitive
- Interpolate

Shade

- Light
- Atmospheric
- Image

- Whitted 82
- Cook 84
- Perlin 85
- Hanrahan 90
- Rhoades 92
Lighting

- Model
- Transform
- Primitive
- Interpolate
- Shade
- Atmospheric
- Image

Light

- Cook 84
- Hanrahan 90
- Slusallek 98
Atmospheric Effects

- Cook 84
- Hanrahan 90
- Ebert 94
Image Effects

- Perlin 85
- Knoll 90
- Slusallek 95
- Kylander 97
Organization

Introduction

Abstract Pipeline

pfman

Hardware and real-life

Conclusions
pfman Language

Like RenderMan
Easy to learn
Allows optimizations
#include <pfman.h>

surface noisy(
    output varying Color px_rc_co[3],
    varying unsigned fixed<16,16>
        transform_as_texture px_shader_texcoord[2],
    float px_material_diffuse[3] = {0.999, 0.999, 0.999})
{
    float ncoord[2] = 8 * px_shader_texcoord;
    float color[3] = px_material_diffuse - noise3d(ncoord);
    px_rc_co = clamp(color, 0, 0.999);
}
Brick Video

width

mortar

Height mortars
Bowling
nanoManipulator
PixelFlow machine

- Molnar, Eyles, Poulton 92
- Eyles, et al. 97
PixelFlow rendering
PixelFlow node

Geometry Network (Shading Parameters) → 2 general processors

Processor Memory → Texture Memory

128x64 SIMD array → Composition Network (Pixel Data)
Bandwidth

- 2 x HP-PA 8000 Processors
- Processor Memory
- Texture Memory
- 128x64 SIMD array

Geometry network
- Geometry data
- Shading control parameters
- 160 MB/sec each direction
Bandwidth

Composition network

- Varying shading parameters
- 6.4 GB/sec each direction
- 142 bytes/pixel
Memory

2 x HP-PA 8000 Processors

- Processor Memory
- Texture Memory
- 128x64 SIMD array

Microprocessor memory
- Render node: geometry
- Shading node: uniform variables
- Instructions for SIMD array
- 128 MB
Memory

2 x HP-PA 8000 Processors

Processor Memory
Texture Memory

128x64 SIMD array

Per-pixel memory
- Varying variables
- 256 bytes
Memory

- 2 x HP-PA 8000 Processors
- 128x64 SIMD array
- Texture maps
- 64 MB
Time

Target
• 640 x 512 / 4 sample (160-128x64 regions)
• 30 frames/second

Time for shading
• Shading nodes / (regions * frame rate)
• 4 nodes = 830 μs
• 8 nodes = 1.7 ms
• 32 nodes = 6.6 ms
Float vs. fixed

Size and time!

1 μs

32-bit float

32-bit fixed

16-bit fixed

+ 

* 

/ 

sqrt
Memory optimization

- 2 x HP-PA 8000 Processors
- 128x64 SIMD array
- Processor Memory
- Texture Memory

<table>
<thead>
<tr>
<th></th>
<th>total bytes</th>
<th>varying</th>
<th>optimized</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>239</td>
<td>175</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td>341</td>
<td>193</td>
<td>137</td>
</tr>
<tr>
<td></td>
<td>216</td>
<td>152</td>
<td>97</td>
</tr>
</tbody>
</table>
Math functions

Often enough to just look right

Floating point sine

• Accurate: 81.36 µs
• Fast: 45.64 µs
<table>
<thead>
<tr>
<th>Shader</th>
<th>bytes free</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>brick</td>
<td>46</td>
<td>613 μs</td>
</tr>
<tr>
<td>ripple</td>
<td>59</td>
<td>1058 μs</td>
</tr>
<tr>
<td>planks</td>
<td>105</td>
<td>532 μs</td>
</tr>
<tr>
<td>bowling pin</td>
<td>86</td>
<td>402 μs</td>
</tr>
<tr>
<td>nanoManipulator 1</td>
<td>75</td>
<td>568 μs</td>
</tr>
<tr>
<td>nanoManipulator 2</td>
<td>1</td>
<td>2041 μs</td>
</tr>
<tr>
<td>nanoManipulator 3</td>
<td>51</td>
<td>1639 μs</td>
</tr>
</tbody>
</table>
Organization

Introduction
Abstract Pipeline
pfman
Hardware and real-life
Conclusions
Conclusions

Interactive shading language is possible

A shading language

• Makes it easier to write shaders
• Makes it easier to optimize
• Can hide hardware details and ideosyncracies
Acknowledgments

UNC and HP PixelFlow team

Users
- Arthur Gregory, Alexandra Bokinsky, Chun-Fa Chang, Aron Helser, Sang-Uok Kum, Renee Maheshwari, Chris Wynn

Funding agencies
- DARPA order numbers A410 and E278
- NSF grant numbers MIP-9305208 and MIP-9612643