SAN ANTONIO
SIGGRAPH
2002

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

SAN ANTONIO SIGGRAPH 2002

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

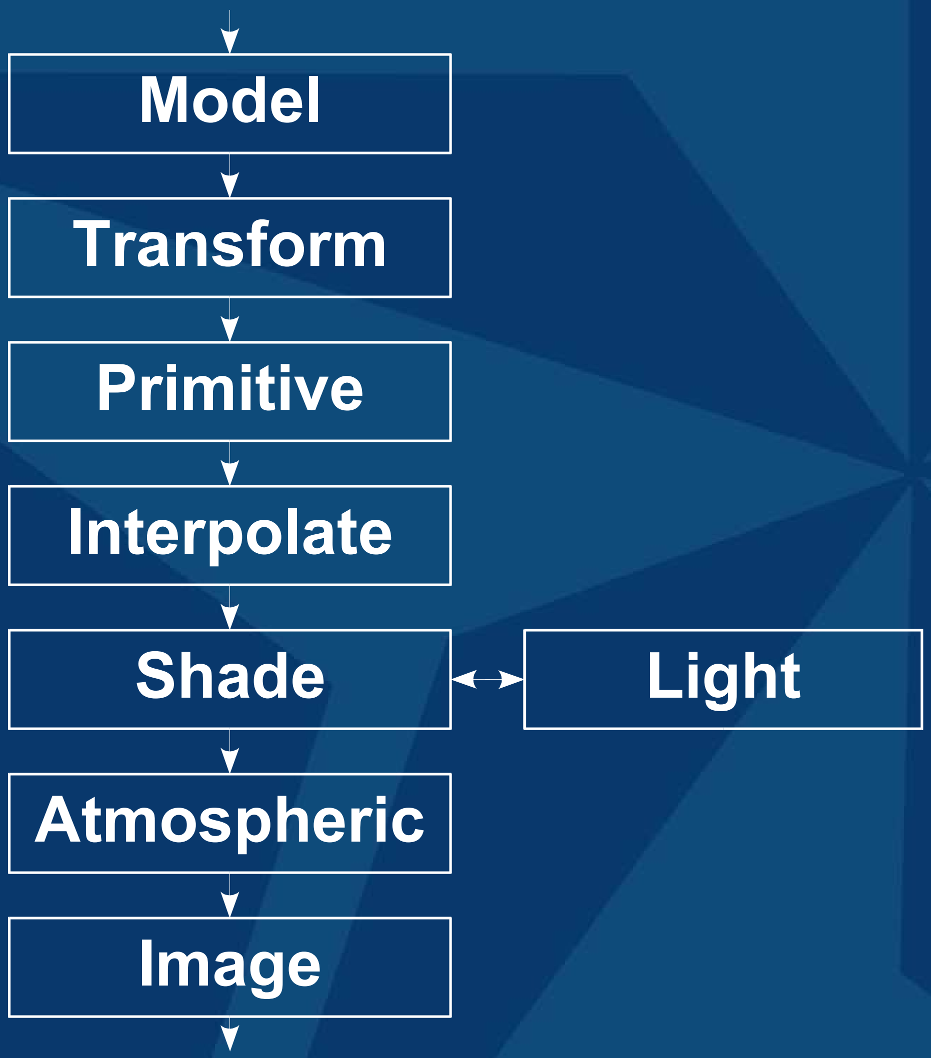
User does not want to know

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

Answer:

- Simple model of machine
- High-level language for procedures

Abstract Pipeline



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

Model

Model

Transform

Primitive

Interpolate

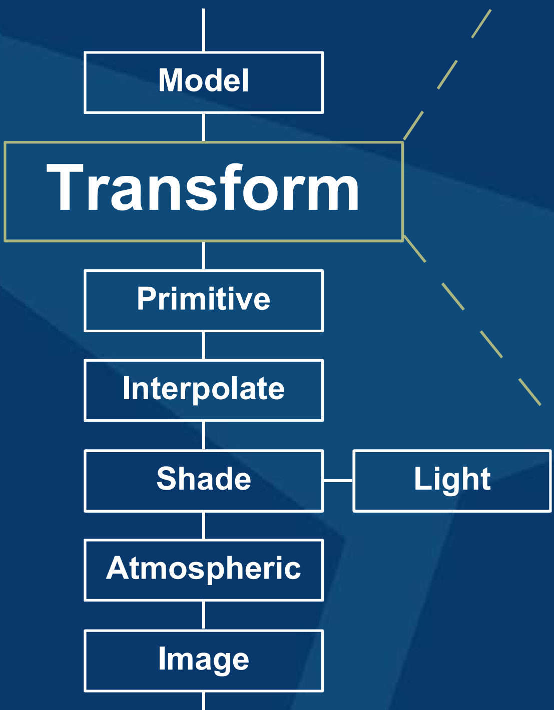
Shade — Light

Atmospheric

Image

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

Transformation



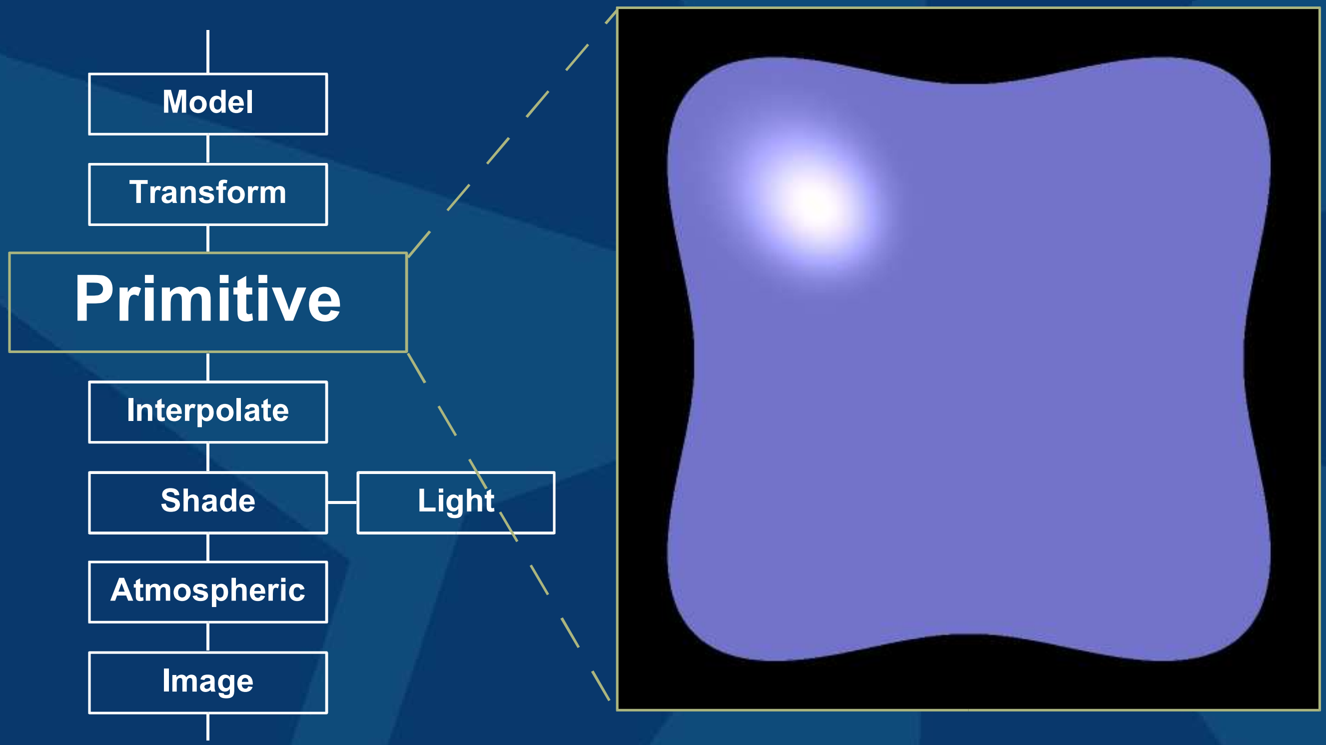
Procedures

- Fleischer 88
- Upstill 90

Techniques

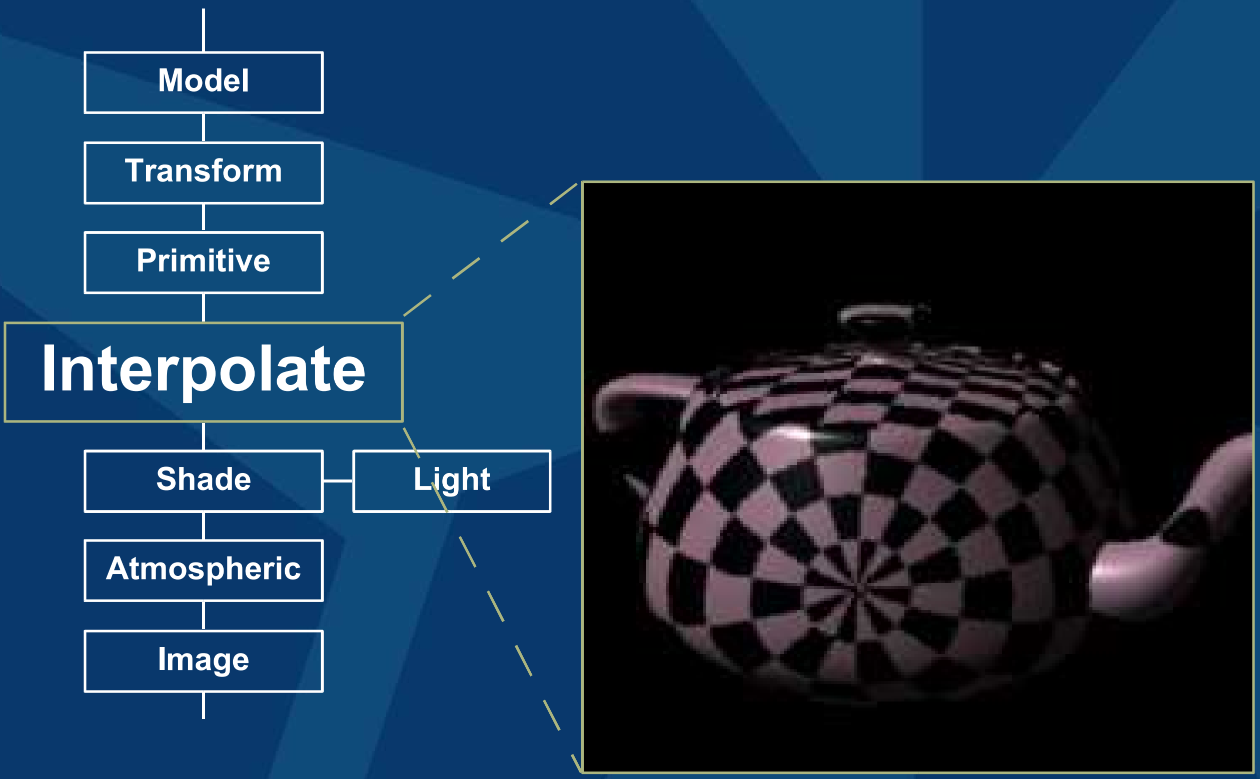
- Barr 84
- Sederberg 86

Primitive



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

Interpolation



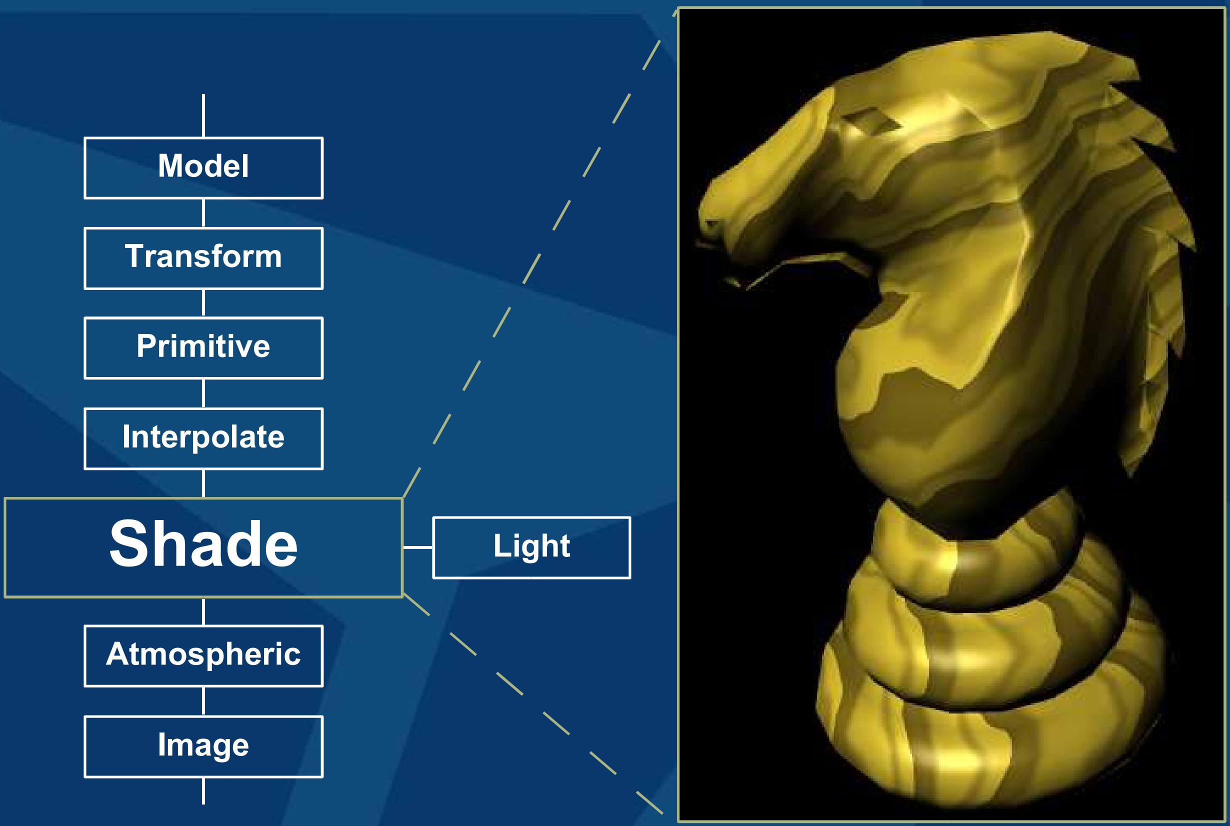
Procedures

- Ebert 94

Techniques

- Neider 93

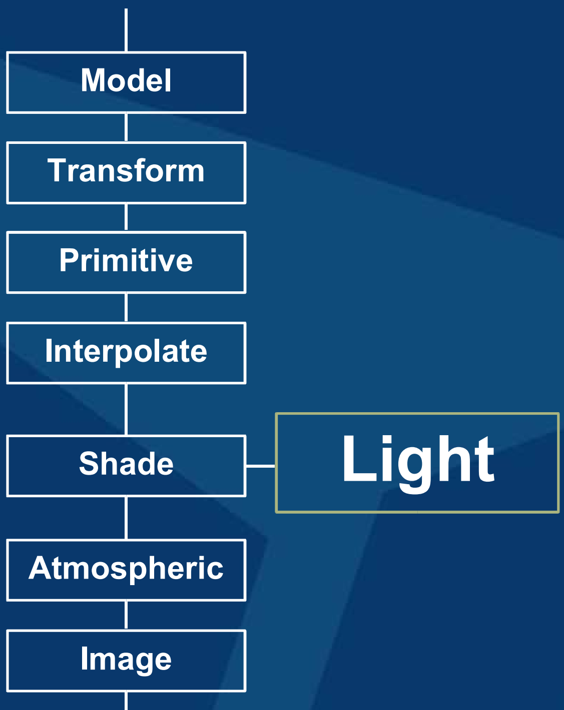
Shading



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

Lighting

- Cook 84
- Hanrahan 90
- Slusallek 98



Atmospheric Effects

- Cook 84
- Hanrahan 90
- Ebert 94

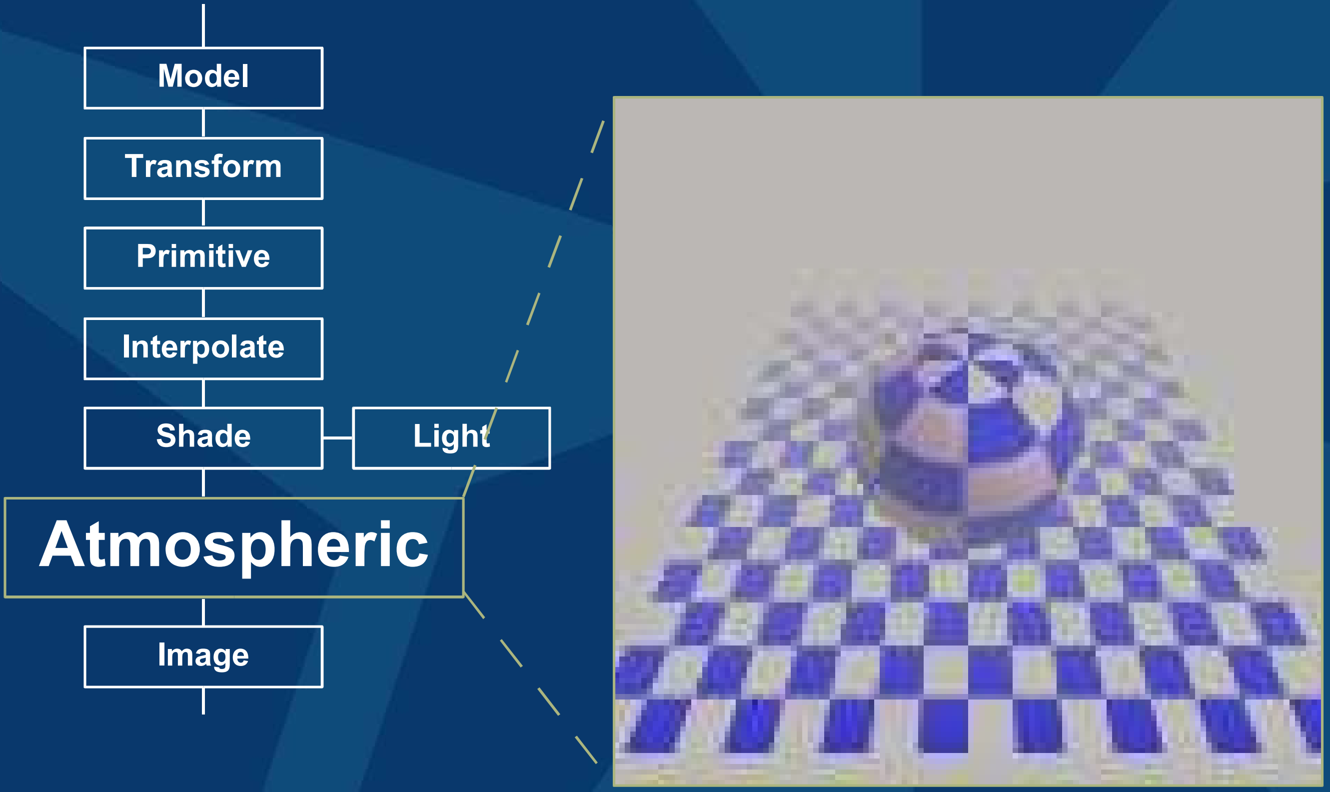
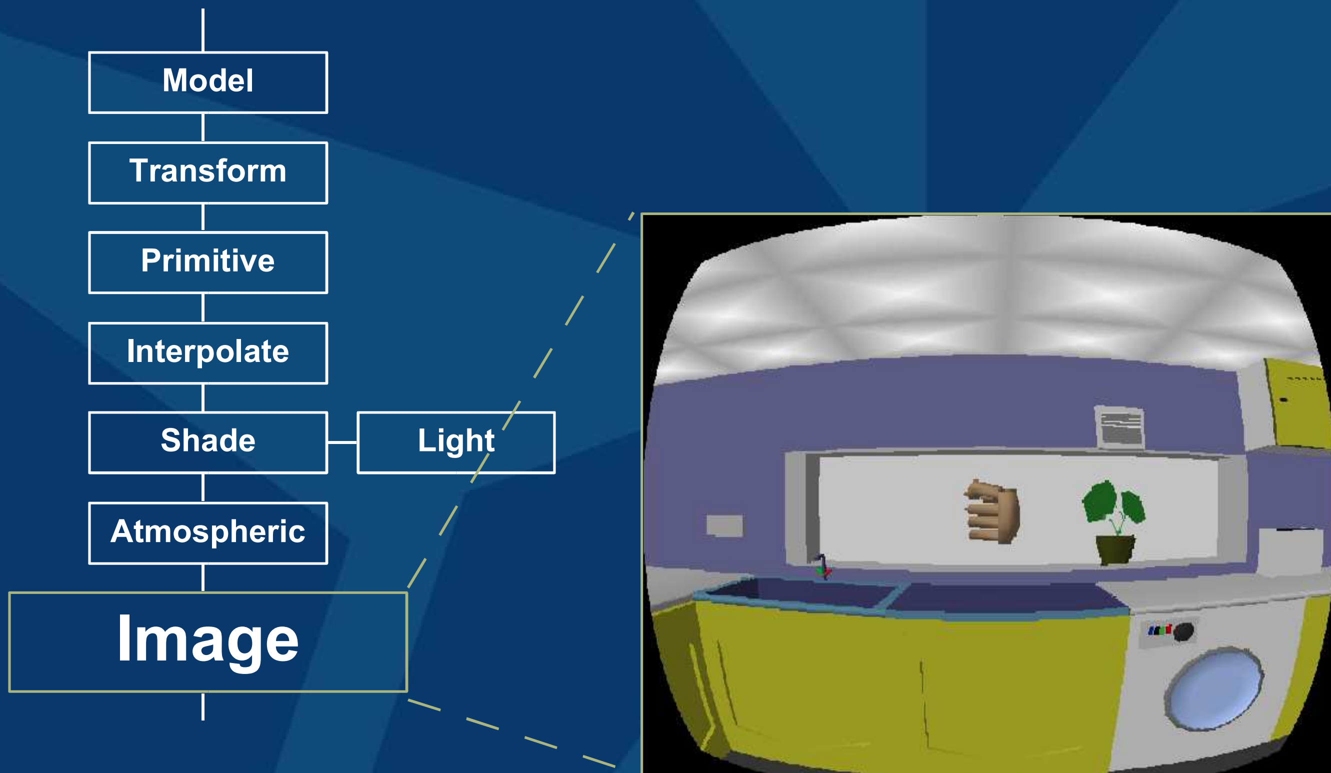


Image Effects

SAN ANTONIO SIGGRAPH 2002



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

Organization

Introduction

Abstract Pipeline

pfman

Hardware and real-life

Conclusions

pfman Language

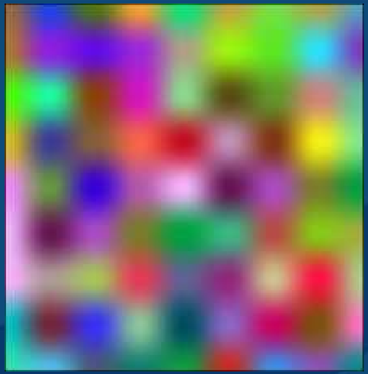
Like RenderMan

Easy to learn

Allows optimizations

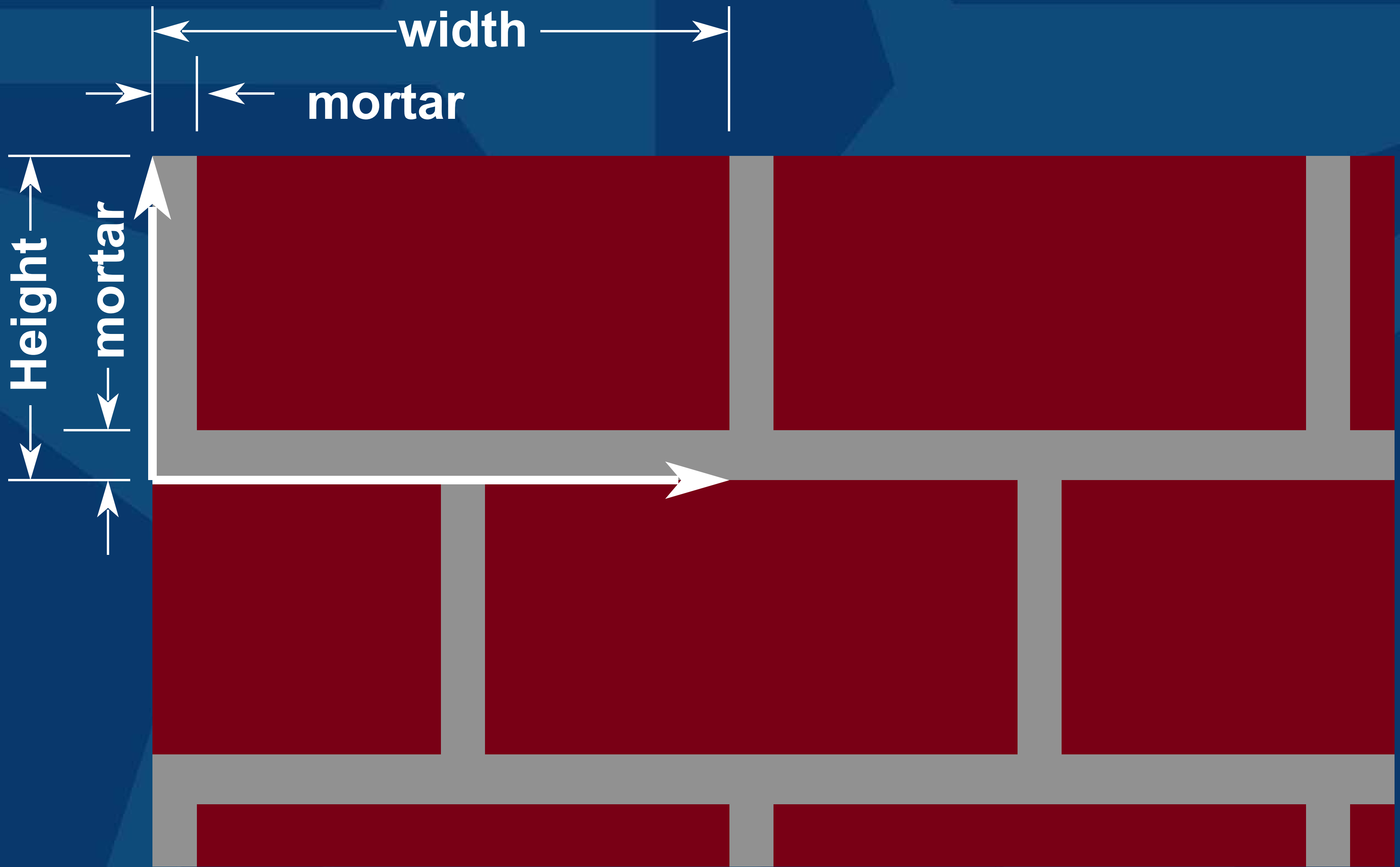
SAN ANTONIO SIGGRAPH 2002

Simple shader

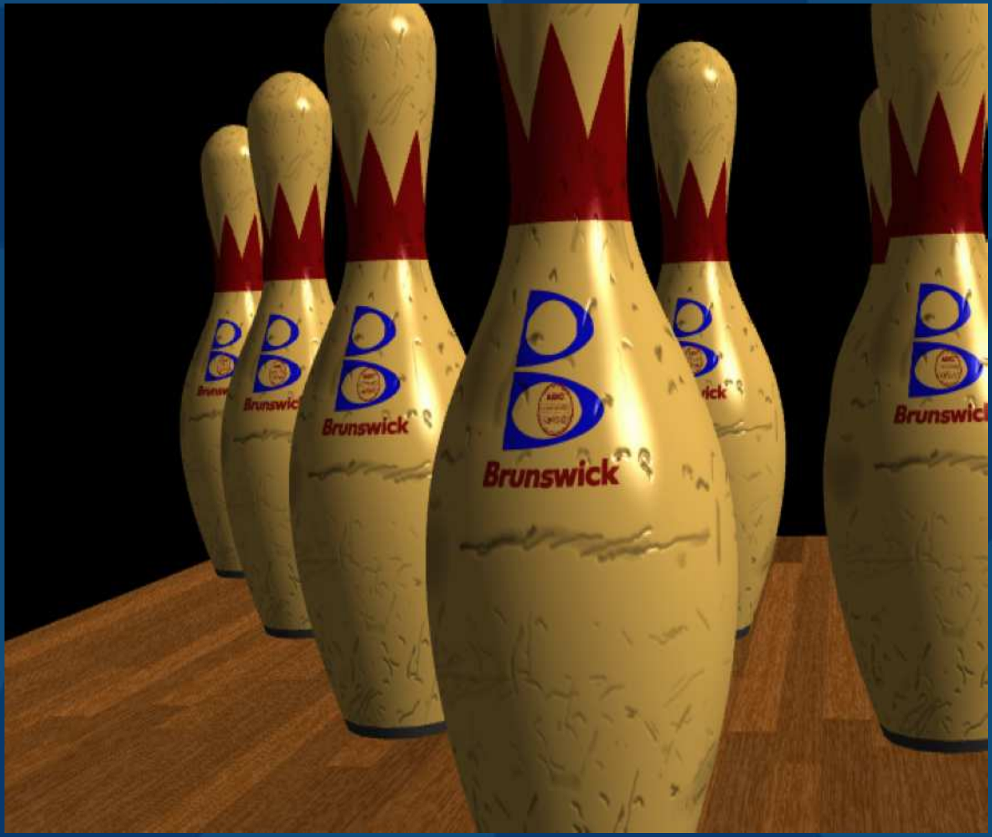


```
#include <pfman.h>
surface noisy(
    output varying Color px_rc_co[3],
    varying unsigned fixed<16,16>
            transform_as_texture px_shader_texcoord[2],
    float px_material_diffuse[3] = {0.999, 0.999, 0.999})
{
    float ncoord[2] = 8 * px_shader_texcoord;
    float color[3] = px_material_diffuse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

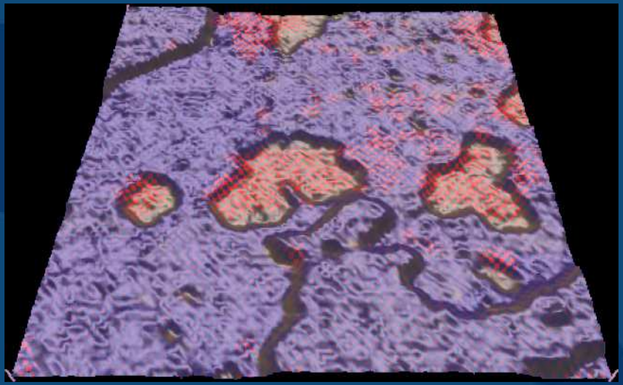
Brick Video

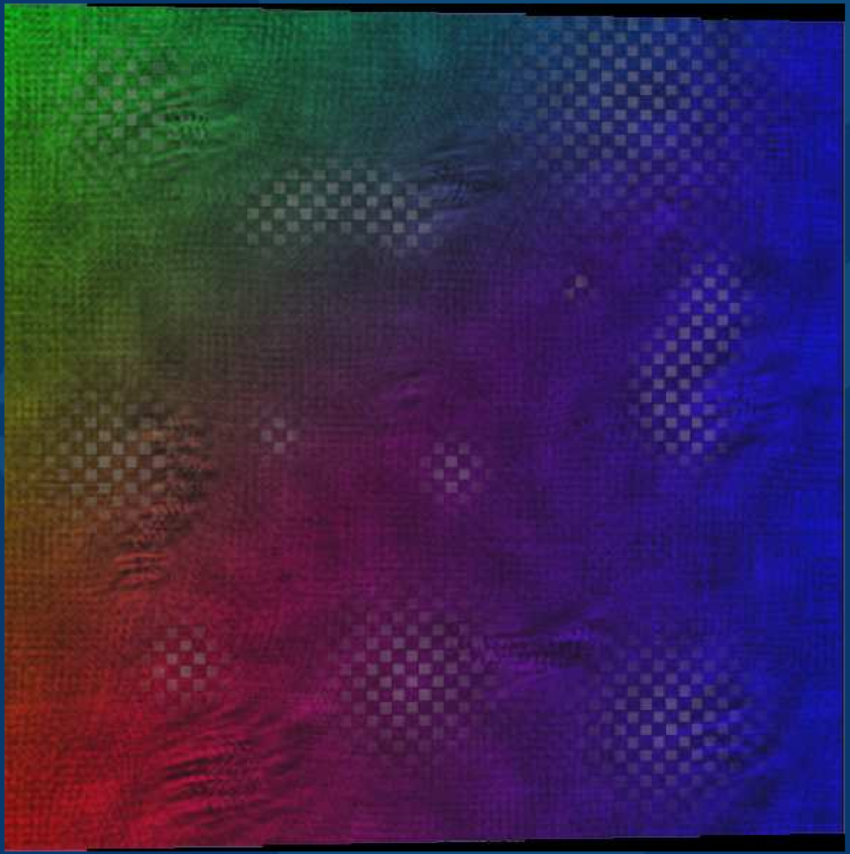


Bowling



nanoManipulator





Organization

Introduction

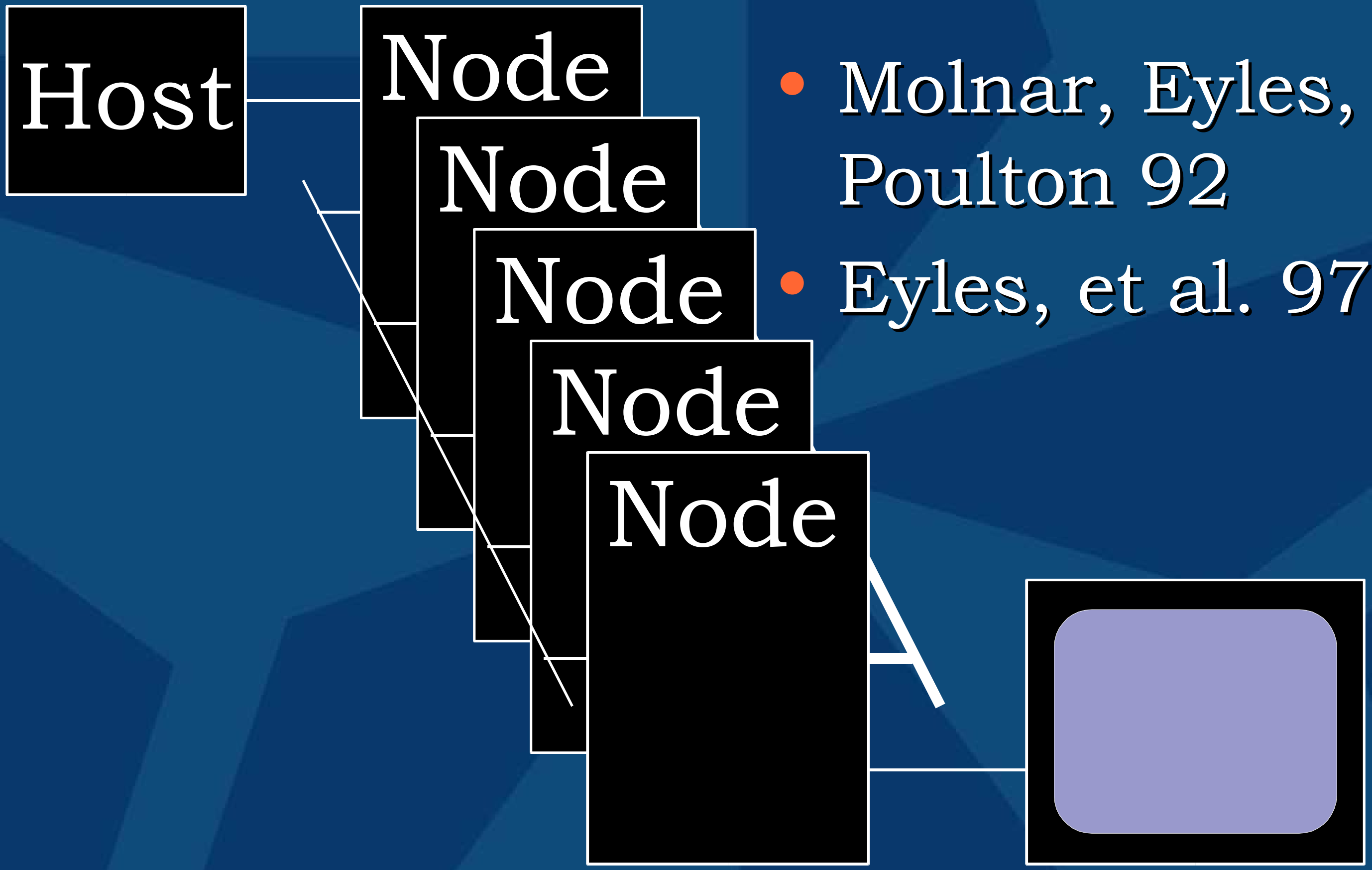
Abstract Pipeline

pfman

Hardware and real-life

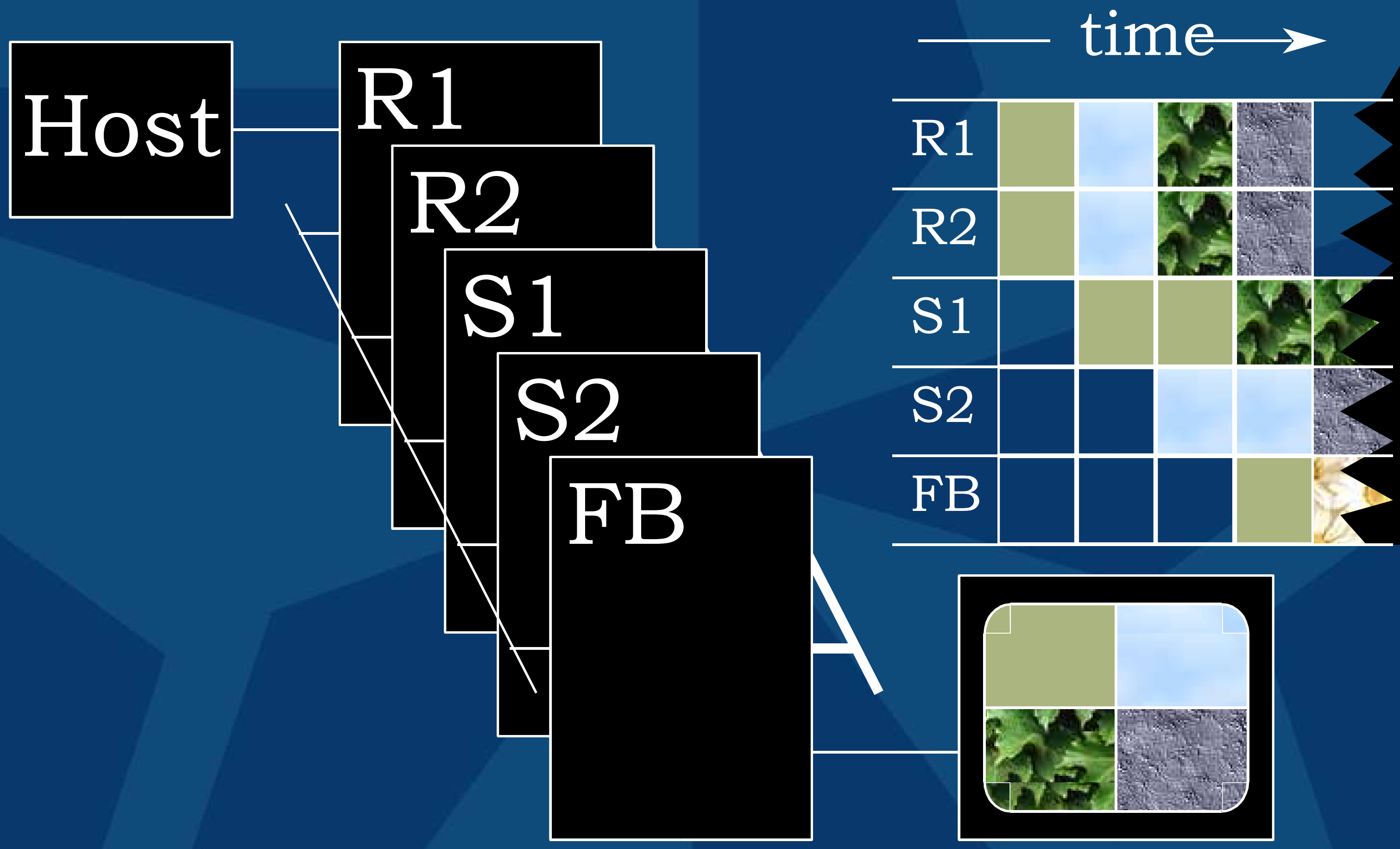
Conclusions

PixelFlow machine

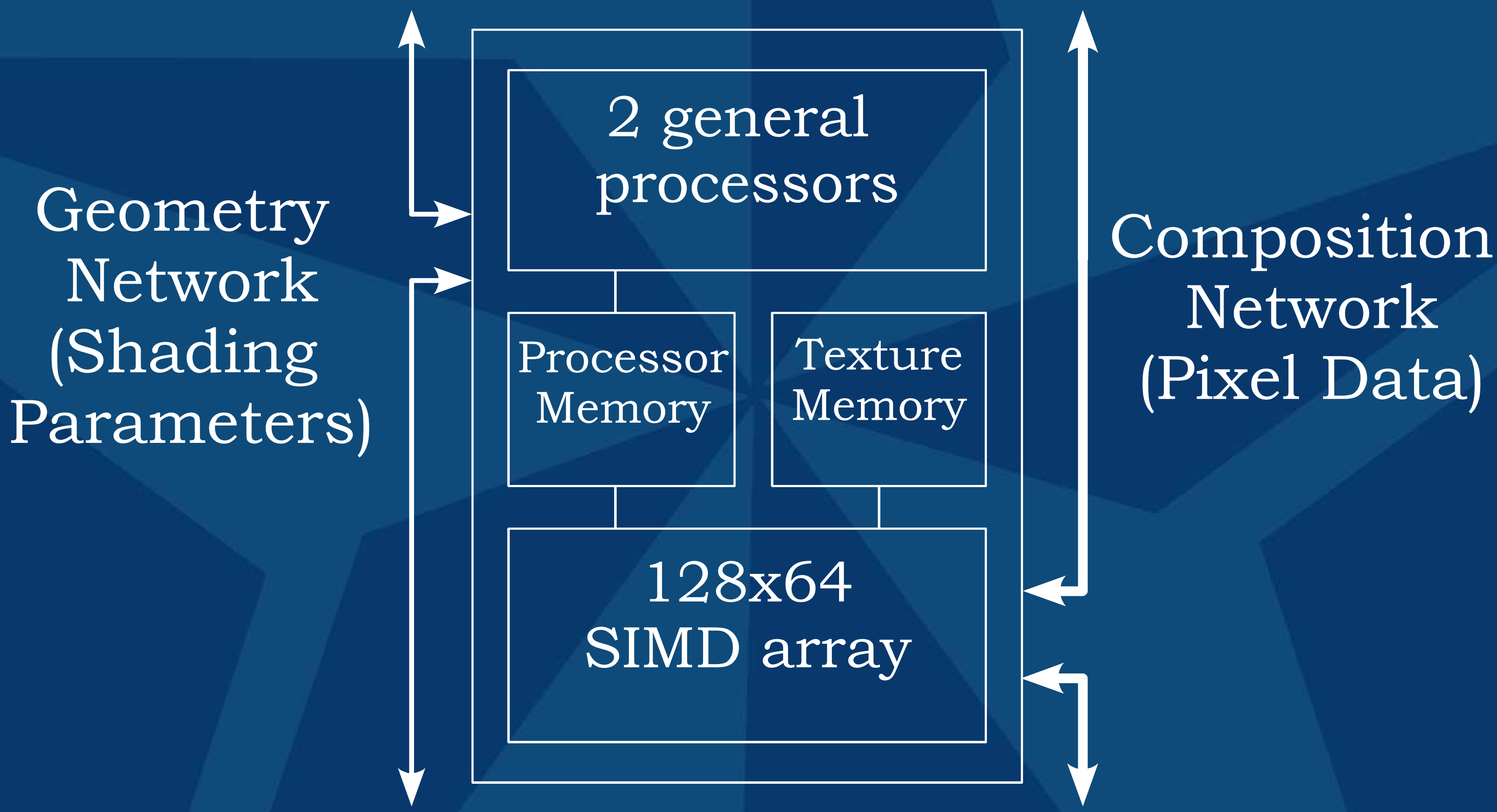


- Molnar, Eyles, Poulton 92
- Eyles, et al. 97

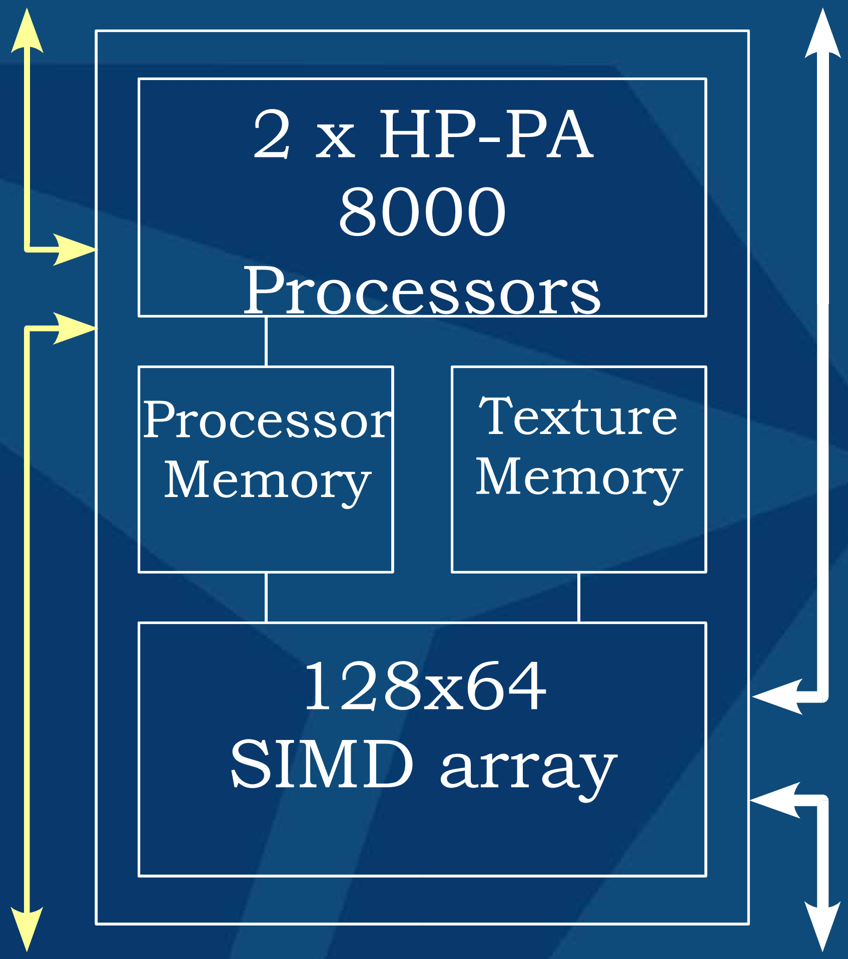
PixelFlow rendering



PixelFlow node



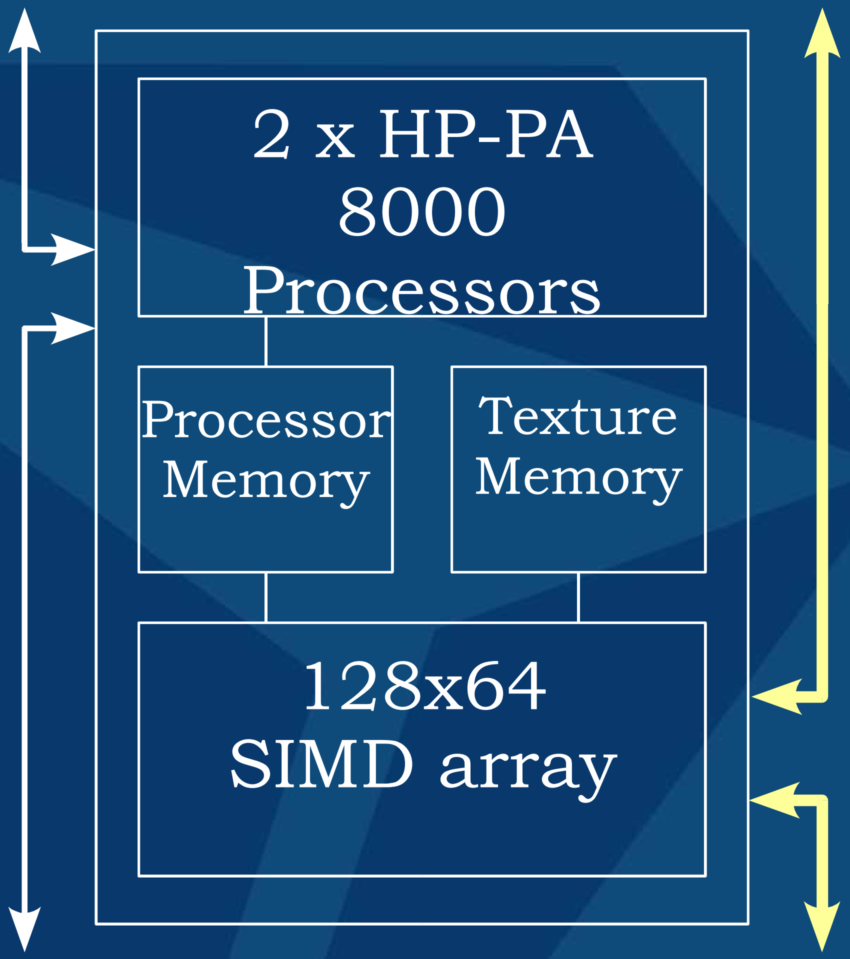
Bandwidth



Geometry network

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

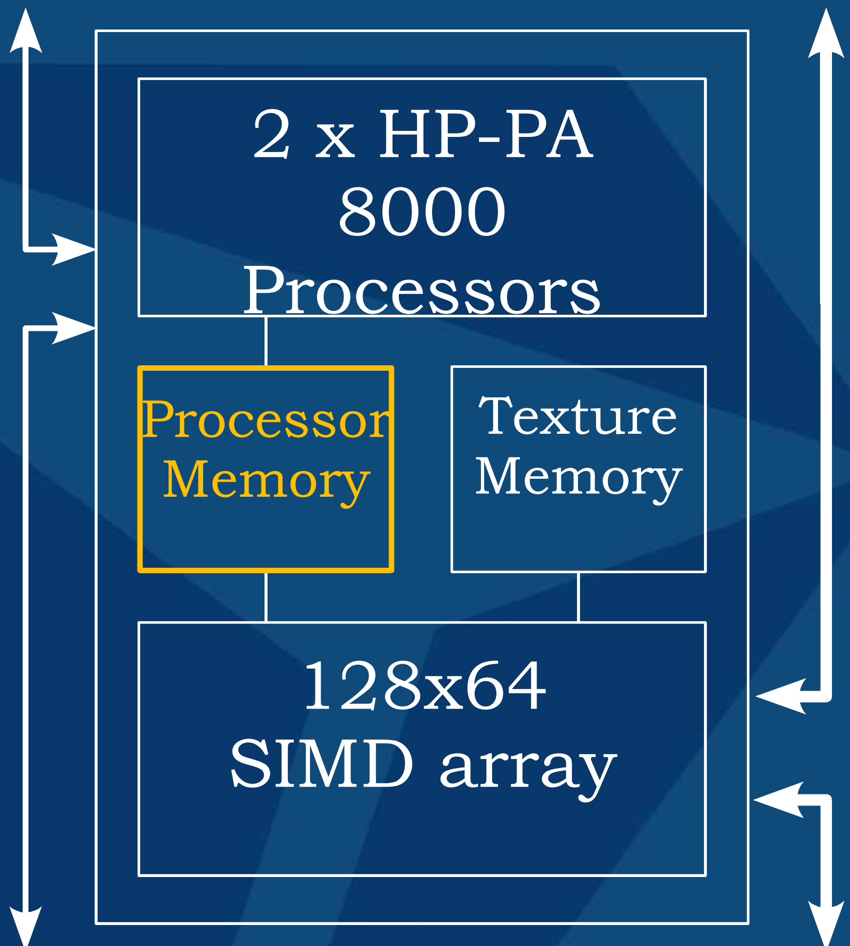
Bandwidth



Composition network

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

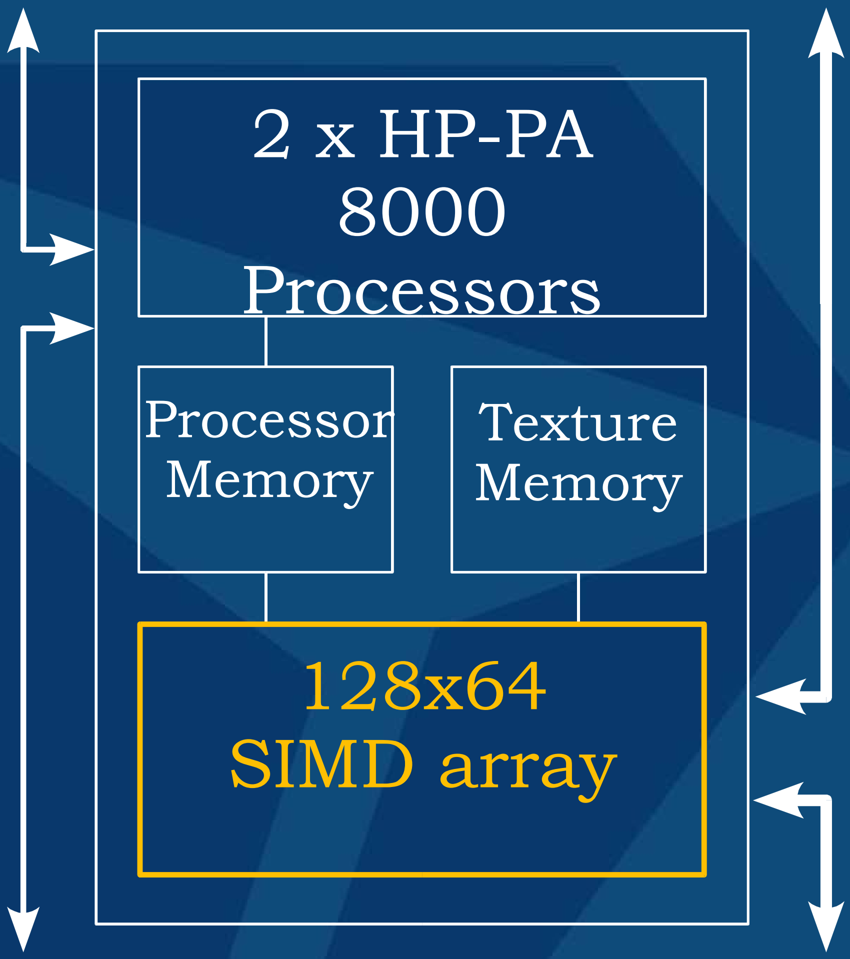
Memory



Microprocessor memory

- Render node: geometry
- Shading node: *uniform* variables
- Instructions for SIMD array
- 128 MB

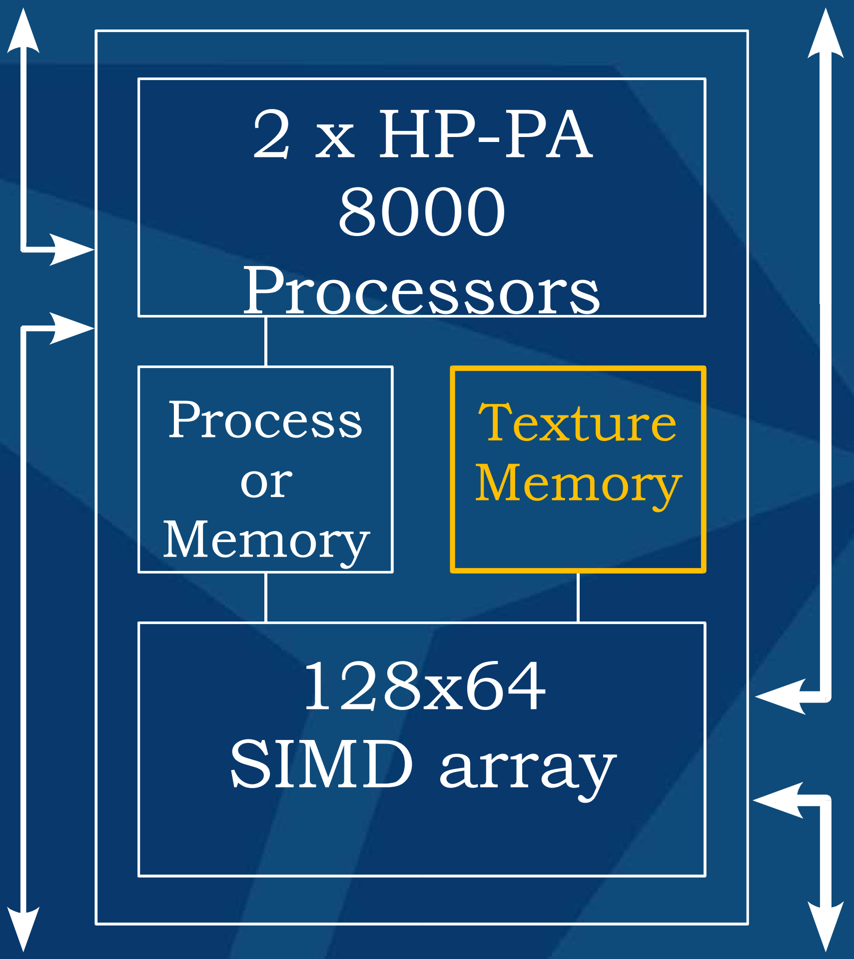
Memory



Per-pixel memory

- Varying variables
- 256 bytes

Memory



Texture memory

- Texture maps
- 64 MB

Time

Target

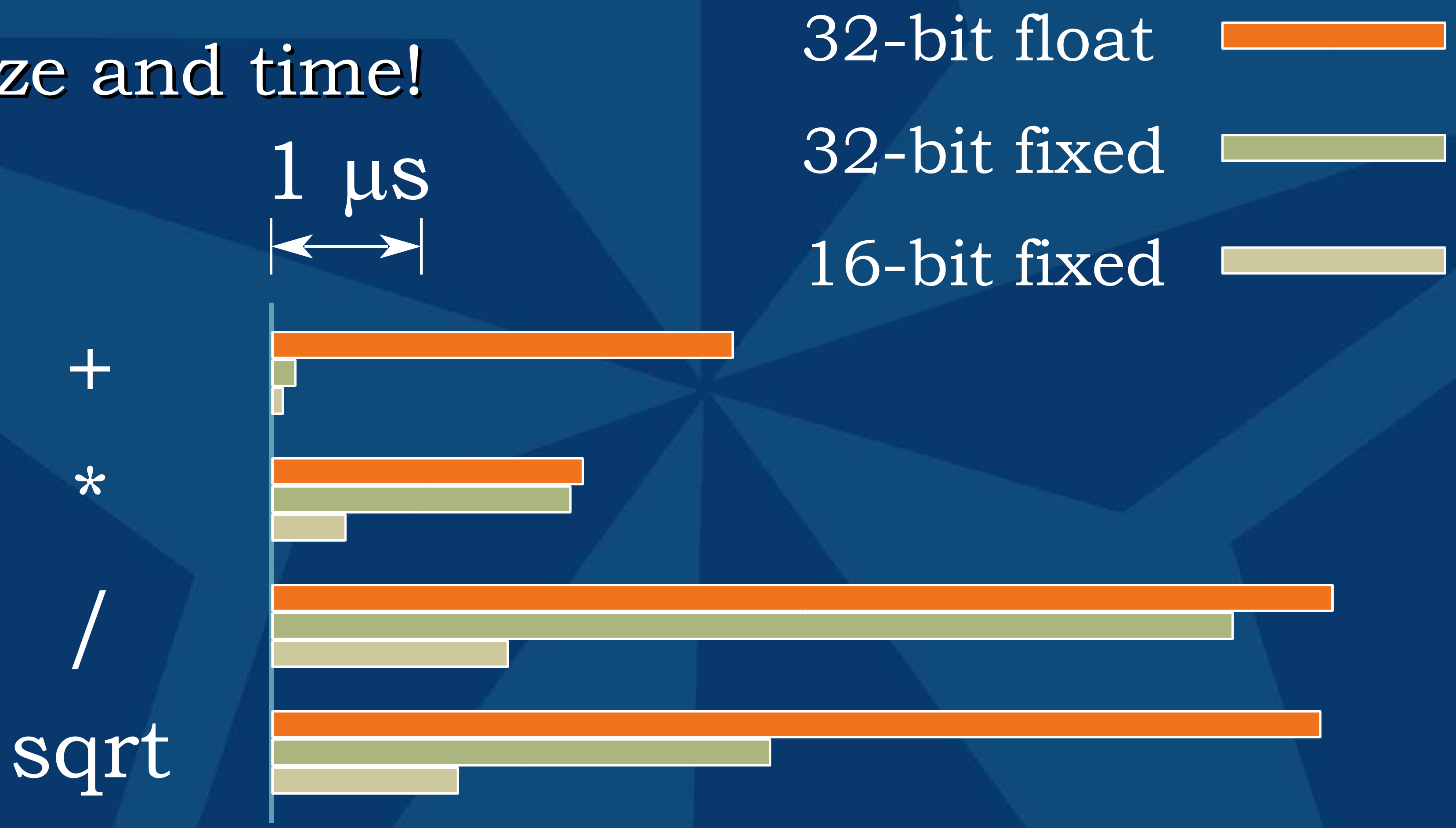
- 640 x 512 / 4 sample (160-128x64 regions)
- 30 frames/second

Time for shading

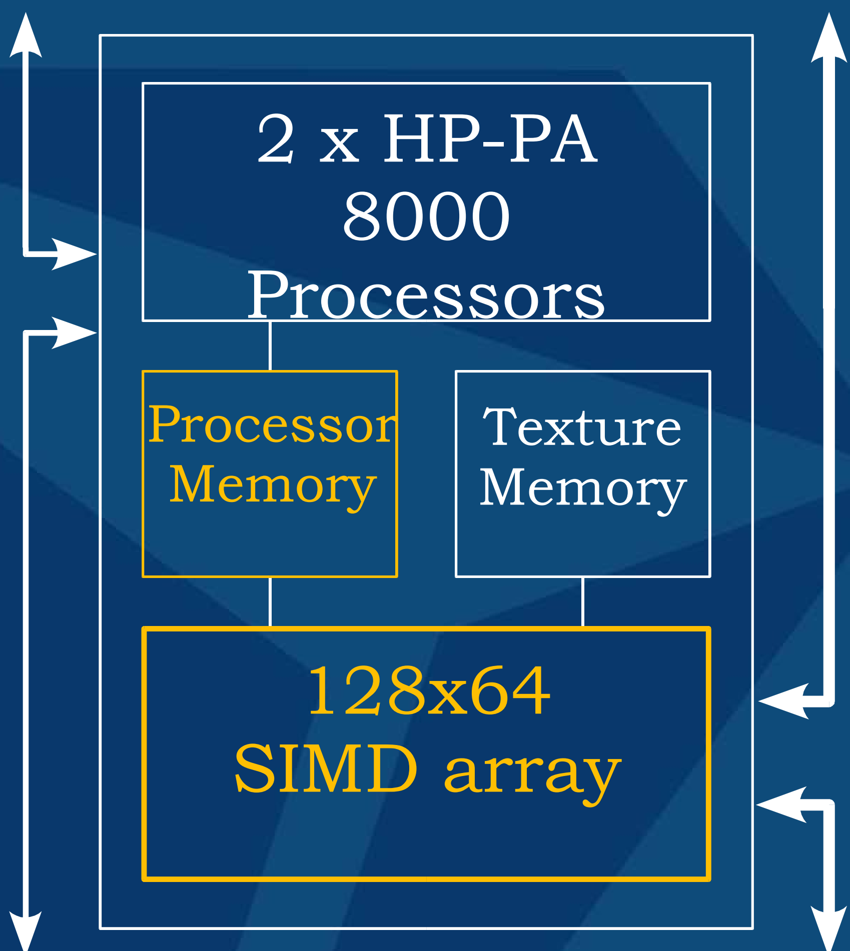
- Shading nodes / (regions * frame rate)
- 4 nodes = 830 μs
- 8 nodes = 1.7 ms
- 32 nodes = 6.6 ms

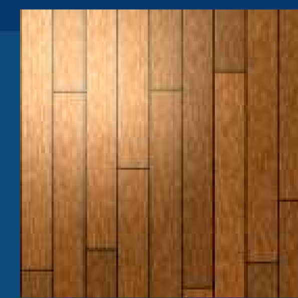
Float vs. fixed

Size and time!



Memory optimization





total bytes	239	341	216
varying	175	193	152
optimized	101	137	97

Math functions

Often enough to just look right

Floating point sine

- Accurate: 81.36 µs
- Fast: 45.64 µs

Results

Shader	bytes free	time
brick	46	613 µs
ripple	59	1058 µs
planks	105	532 µs
bowling pin	86	402 µs
nanoManipulator 1	75	568 µs
nanoManipulator 2	1	2041 µs
nanoManipulator 3	51	1639 µs

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

SAN ANTONIO
SIGGRAPH 2002

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

SAN ANTONIO

SIGGRAPH 2002