UNC High-Performance Rendering Hardware

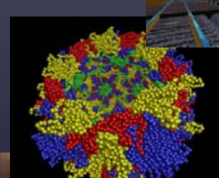
Anselmo Lastra

October 2001

UNC Graphics Hardware

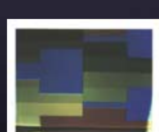
WarpEngine
???
image-based

PixelFlow
1997
scalable
shading HLL



Pixel-Planes 5
1991
fully parallel
prog. shading

Pixel-Planes 4
1986
proc. per pixel
512 x 512
shadows, etc.

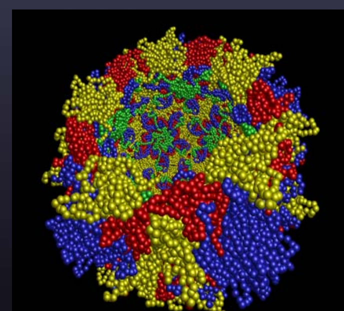


Pixel-Planes
1, 2, 3
1980-85

Anselmo Lastra
October 2001

Why Is CG Difficult?

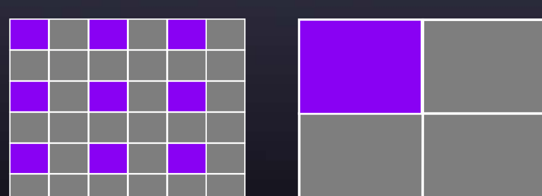
- Floating Point Requirements
  – Approximately 400 MFlops for 1M triangles
  – For example: Infinite Reality: Eight geometry engines at 480 MFlops each for 10 million triangles per second peak
- Memory Bandwidth
  – Approximately 250 million frame buffer accesses to rasterize 1 million 100-pixel triangles



Lastra, Oct. 2001 Slide 3

Rasterization

- Memory size increase 3-4 orders of magnitude
- Speeds have not kept up!
- Number of pins remained relatively constant.
- Distributed frame buffers were proposed to address the problem [Fuchs77][Parke80].

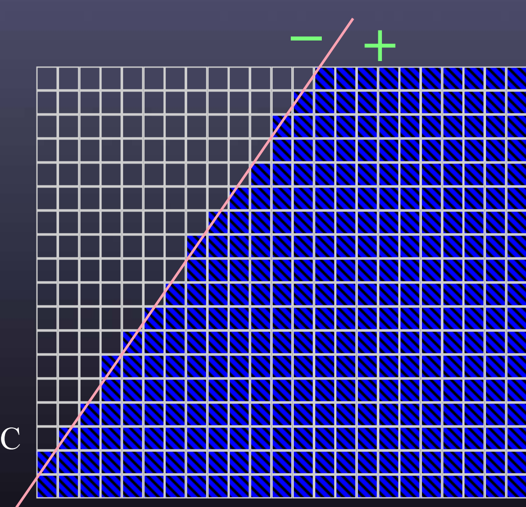


Lastra, Oct. 2001 Slide 4

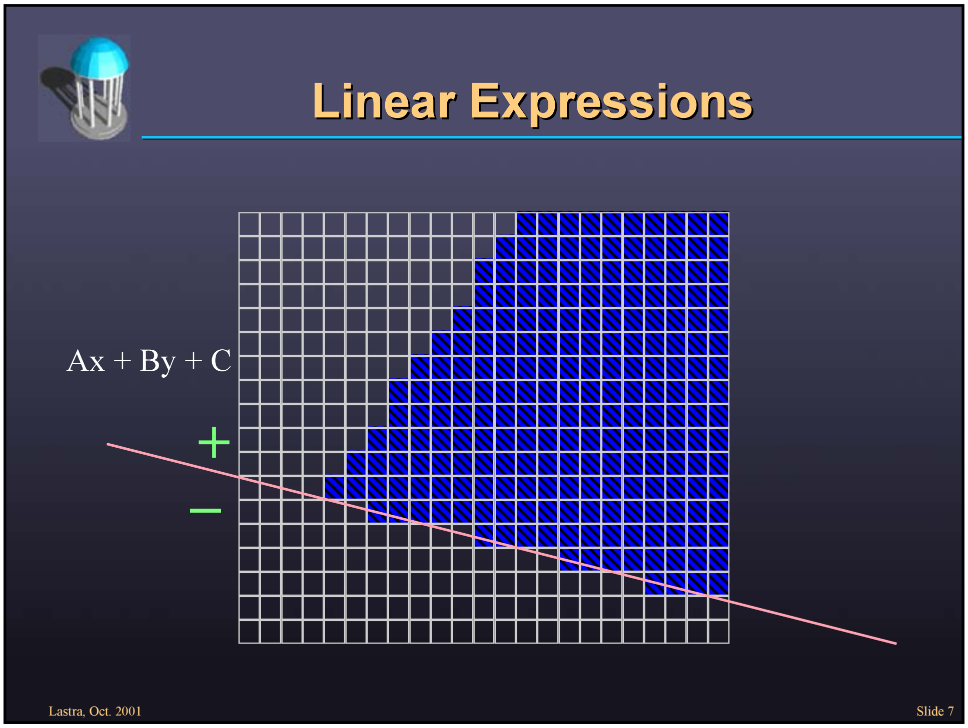
Pixel-Planes

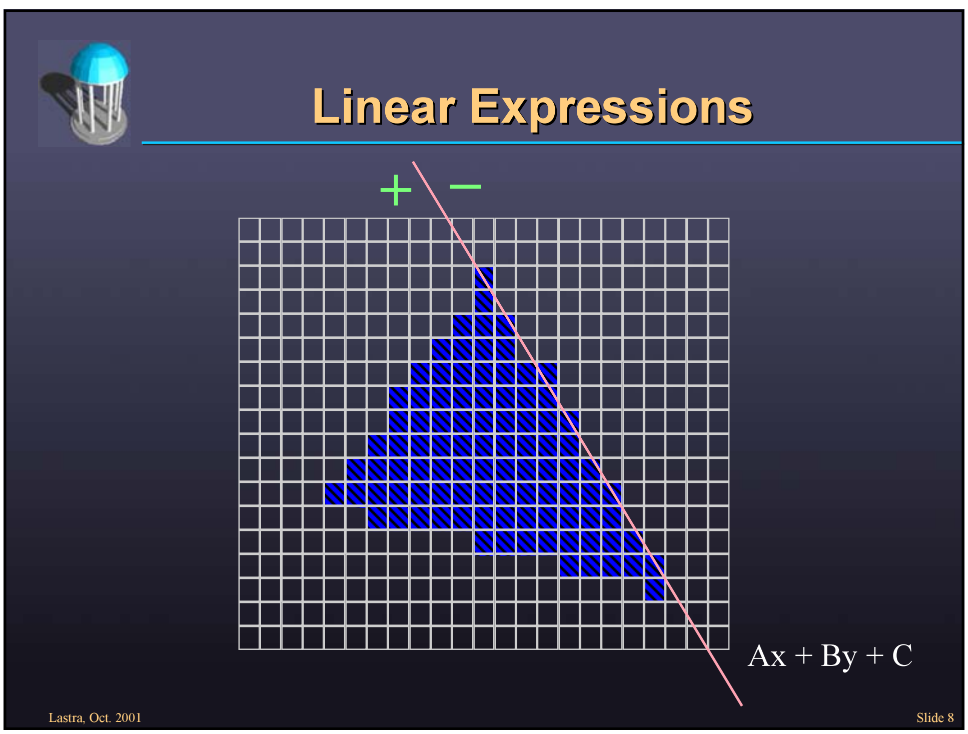
- Henry Fuchs' Idea: build processing into the frame buffer, a *processor per pixel.*
  - UNC designs are called *enhanced memories*,
  - not *SIMD processors*
- Enabler was linear expression tree…

Linear Expressions



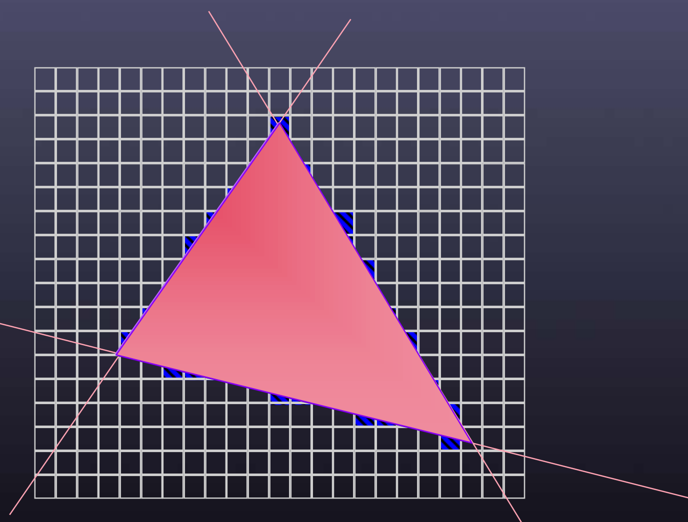
Ax + By + C


Linear Expressions
Ax + By + C
+
–
Lastra, Oct. 2001
Slide 7


Linear Expressions
+
–
Ax + By + C
Lastra, Oct. 2001
Slide 8

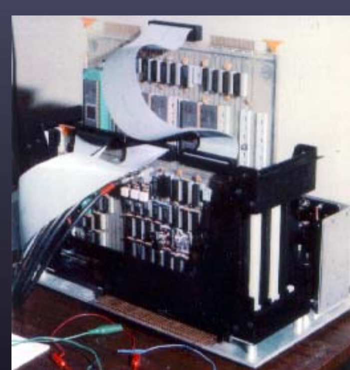
Depth and Color Interpolation

z = F(x, y) and color = F(x, y) for each of red, green, and blue



Pixel-Planes 1, 2, 3

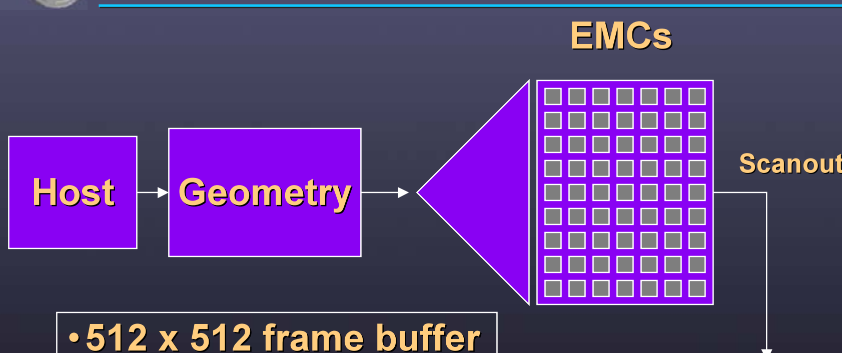
- Pixel-Planes 1 - 4 processors
- Pixel-Planes 2 - 4 by 64
- Pixel-Planes 3 - 64 by 64



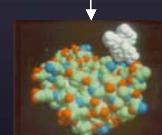
Pixel-Planes 4 (1986)

- Full-size (512 by 512 pixel) prototype
- 2048 enhanced memory ICs
- One Geometry Processor
- 72 bits memory per pixel

Pxpl4 Block Diagram



- 512 x 512 frame buffer
- 128 processors/chip
- 2048 EMC chips
- 32 boards
- Clock speed: ~ 10MHz



Performance

- 35K triangles per second
- Spheres as a primitive
- CSG
- Shadows

Video

Lastra, Oct. 2001 Slide 13

References

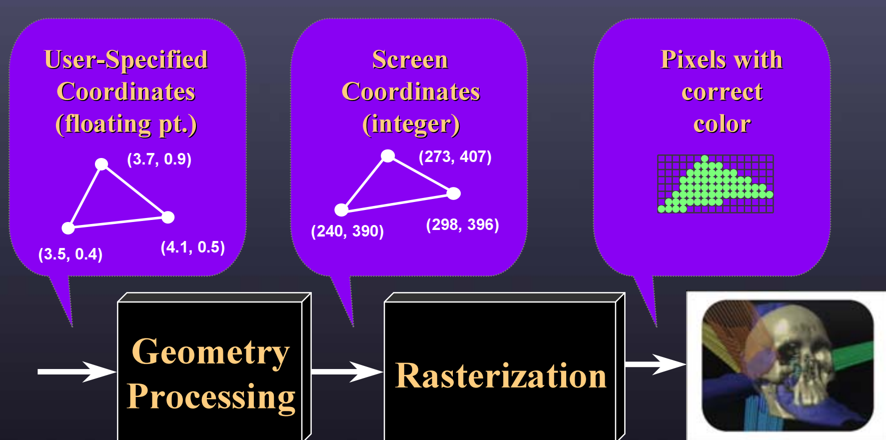
- The first reference describes the algorithms, while the second describes the machine as-built.
- Fuchs, H., Goldfeather, J., Hultquist, J., Spach, S., Austin, J., Brooks, F., Eyles, J., and Poulton, J., "Fast Spheres, Shadows, Textures, Transparencies, and Image Enhancements in Pixel-Planes," SIGGRAPH '85 Conference Proceedings, Vol. 19, No. 3, July, 1985, pp 111-120.
- John Eyles, John Austin, Henry Fuchs, Trey Greer and John Poulton, “Pixel-Planes 4: A Summary”, Eurographics Workshop on Graphics Hardware, 183-207, 1987.
- Web pages at http://www.cs.unc.edu/~pxpl

Lastra, Oct. 2001 Slide 14

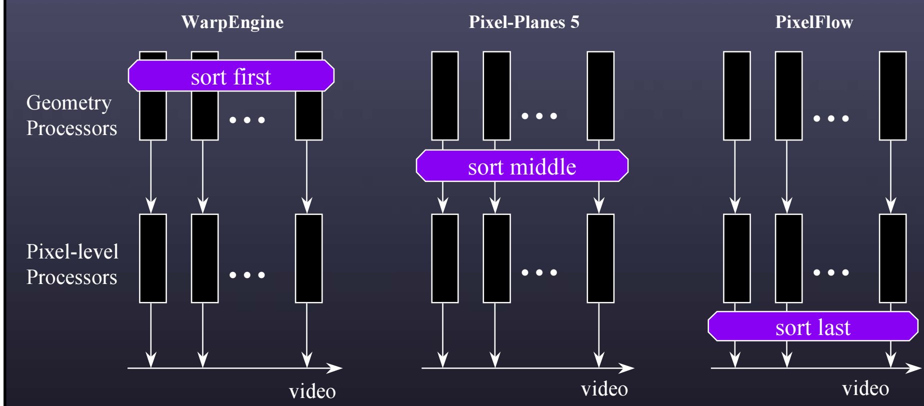
Lessons Learned from Pxpl4

- Programmability useful!
- More pixel memory required
- Tris small,
  - many processors unused
- Must extend parallelism to geometry processing → **fully parallel pipeline**

Graphics Pipeline



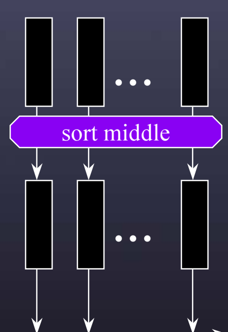
Sorting Classification



Lastra, Oct. 2001 Slide 17

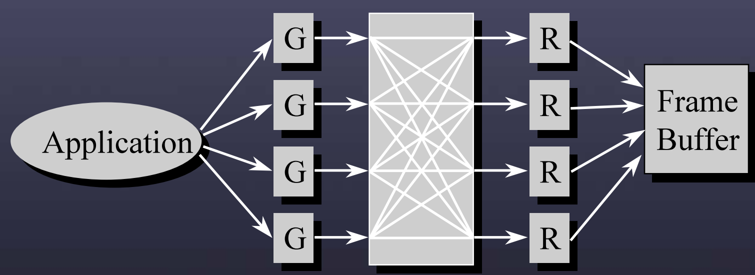
Sort Middle

- Straightforward and well known.
- Network limits scalability.
- Somewhat scalable in display size.



Lastra, Oct. 2001 Slide 18

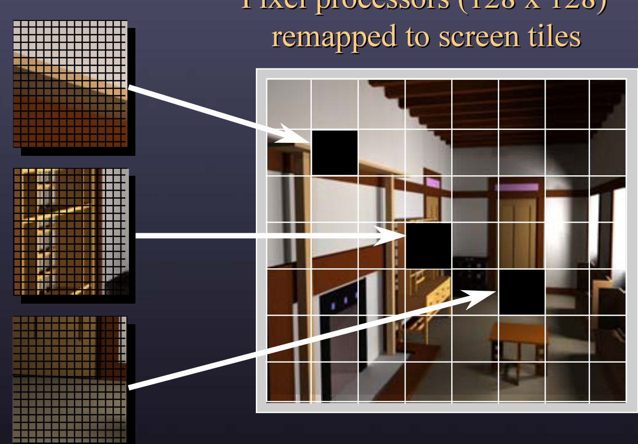
Sort Middle (Pixel-Planes 5)



- Perform geometry processing in parallel by primitive. Sort each transformed primitive to determine where on screen it belongs.
- Route transformed primitive to renderer responsible for appropriate screen regions.

Screen-Space Subdivision

Pixel processors (128 x 128) remapped to screen tiles



Problem: Polygons must be sorted!

Virtual Frame Buffer

- Must sort all primitives before scan conversion
- Pipelined two frames,
  - –one in geometry stage,
  - –another in scan conversion
- Penalty: memory & increase in latency

Lastra, Oct. 2001 Slide 21

Load Balancing on pxpl5

- Greedy algorithm
- Master collects info on # of prims / region
- Starts token, which flows GP to GP

Rasterization order	1	2	3	3	2	1	2	3

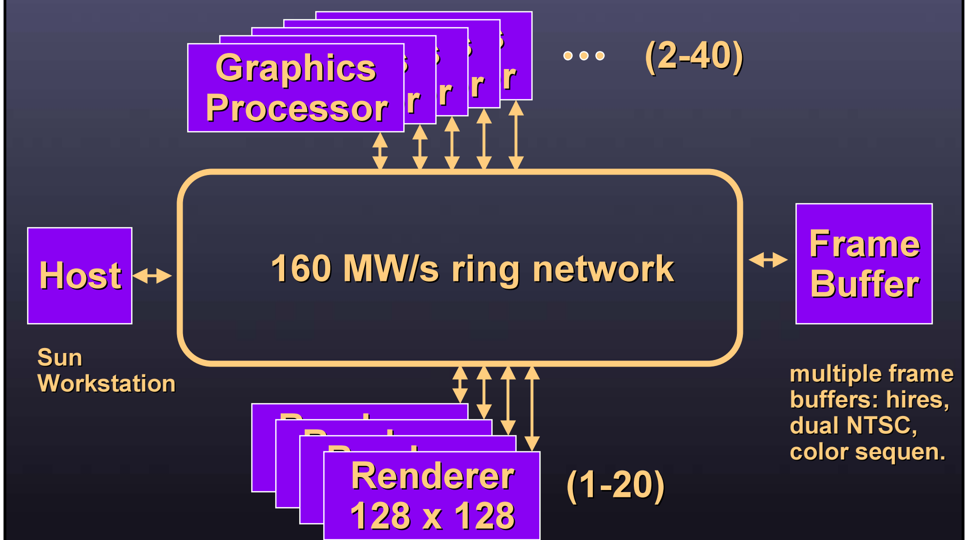
Size of box indicates # of primitives

- Last GP sends token back to master

When does this fail?

Lastra, Oct. 2001 Slide 22

Implementation



Technology

- 1-bit ALU
- Quadratic evaluator

  $Ax^2+By^2+Cx+Dy+Exy+F$
- 208 bits/pixel
- 4K backing store/pixel
- 40 MHz clock speeds

Shading

- Finally enough bits to do cool shading!
- Language was assembler with macros (sqrt, norm, etc). Word length variable.

Video

Image courtesy Division – from Pixel-Planes 6

Problem: too hard to write code!

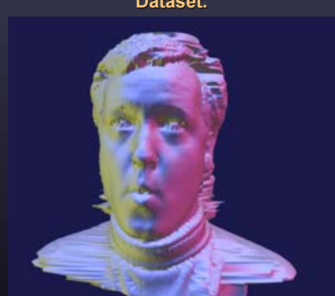
Lastra, Oct. 2001 Slide 25

Performance

- Triangles small, but

the size doesn't matter as long as overlap factor does not increase much!

Record Performance on GPC "head" Dataset.



Model courtesy GPC committee (now SPEC)

Dataset courtesy Herman Towles and Sun Microsystems

Lastra, Oct. 2001 Slide 26

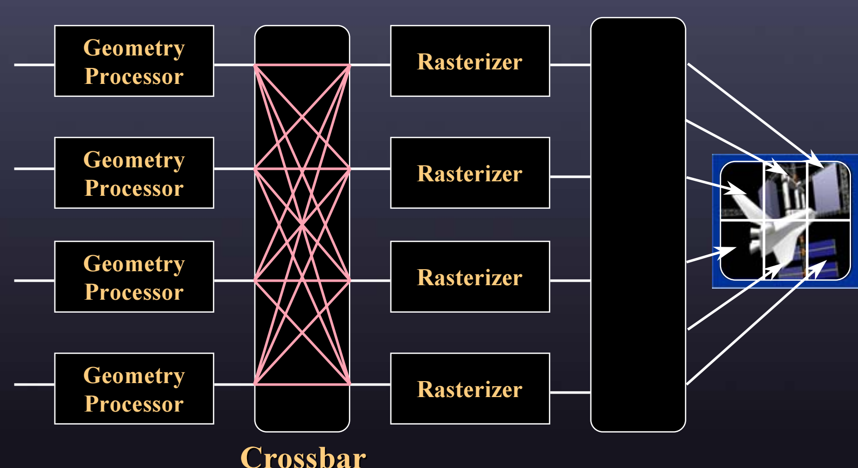
References

- Fuchs, H., Poulton, J., Eyles, J., Greer, T., Goldfeather, J., Ellsworth, D., Molnar, S., Turk, G., and Israel, L., "A Heterogeneous Multiprocessor Graphics System Using Processor-Enhanced Memories," Computer Graphics (Proc. of SIGGRAPH '89), Vol. 23, No. 3, pp 79-88.
- John Rhoades, Greg Turk, Andrew Bell, Andrei State, Ulrich Neumann and Amitabh Varshney, "Real-time procedural textures", Proceedings of the 1992 Symposium on Interactive 3D Graphics (Cambridge, Massachusetts, March 29--April 1, 1992). In Computer Graphics special issue. ACM SIGGRAPH, March 1992. pp. 95--100.

Lastra, Oct. 2001 Slide 27

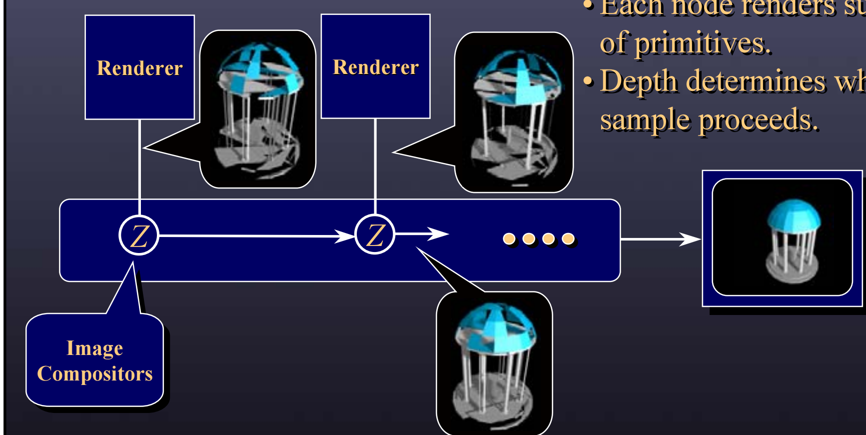
Limits to Scalability

Crossbar bandwidth must increase with number of primitives



Lastra, Oct. 2001 Slide 28

Image Composition

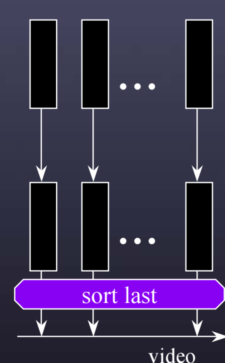


- Each node renders subset of primitives.
- Depth determines which sample proceeds.

- 256 wires @ 200 MHz, bi-directional (> 100 Gb/s) board to board

Sort Last (Image Composition)

- Scalable in # of primitives
- Requires high-bandwidth network (next slide)
- Anti-aliasing expensive
- Transparency difficult.
- Not scalable in display size.



Composition Bandwidth

- Need enough bandwidth for:

  display size × frame rate × subsamples
- For 1280 × 1024 at 72 Hz, with 5 sample antialising,
  - need 10 Gigabits/sec of bandwidth

Lastra, Oct. 2001 Slide 31

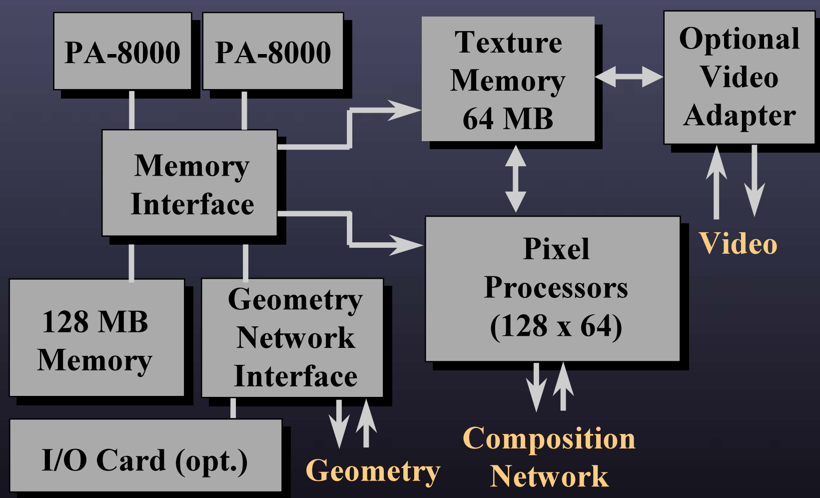
PixelFlow Goals

- Investigate image composition
- Add high-level shading language
  - Turned into programmability everywhere in pipeline
- Immediate mode from a parallel machine

Video

Lastra, Oct. 2001 Slide 32

PixelFlow Node



Shading Functions

- Pins
  - Crown, label, scuffs, dirt, Phong
- Alley
  - Wood, reflection map
- Ball
  - Phong
- Light
  - Shadow map

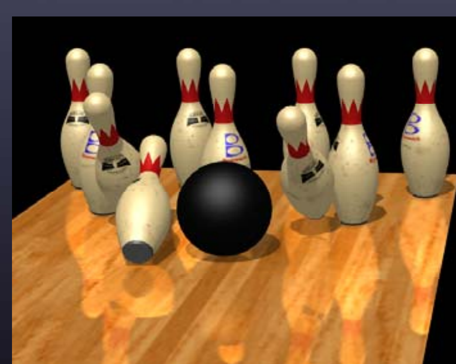
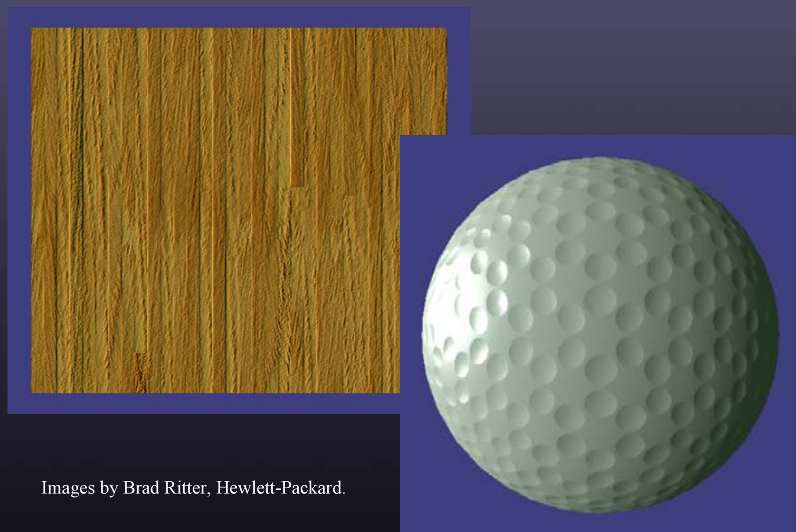


Image by Yulan Wang , UNC.

Bump Maps



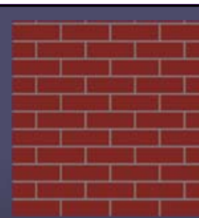
Images by Brad Ritter, Hewlett-Packard.

Enabling Programmable Shading

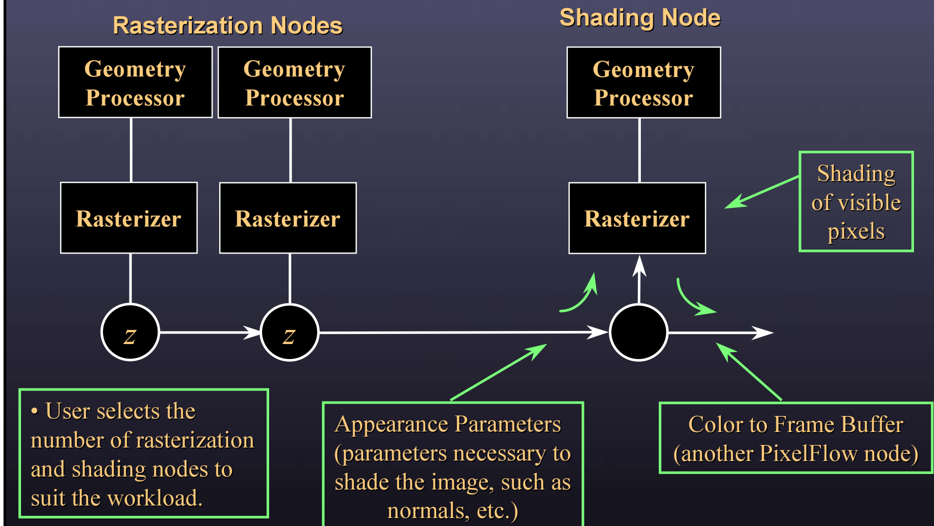
- Shading Language
  - –pfman
  - –Similar to RenderMan with extensions mainly for speed on PixelFlow
- Extensions to OpenGL to allow access from geometry code.

Videos

```
#define BRICK_WIDTH 0.25
#define BRICK_HEIGHT 0.08
#define MORTAR 0.01

surface brick(
    output unsigned varying fixed<8,8> gl_rc_co[3],
    unsigned texture varying fixed<16,16>
        gl_material_texcoord[2]) {

        float ss, tt;
        fixed<8,0> row;

        tt = gl_material_texcoord[1] % (BRICK_HEIGHT + MORTAR);
        row = gl_material_texcoord[1] / (BRICK_HEIGHT + MORTAR);

        gl_rc_co[0] = 0.5;              // both brick & mortar same red
        if(tt > brick_height) {         // Is it a row of brick or mortar?
                gl_rc_co[1] = 0.5;      // within mortar row
                gl_rc_co[2] = 0.5;
        }
        else {                          // within brick row
              ...
```



Deferred Shading



Advantages and Disadvantages

- You only shade pixels that are visible
- Increased coherence

  but...

- Must save and transmit many parameters → high bandwidth demand

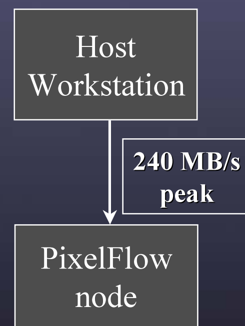
Video

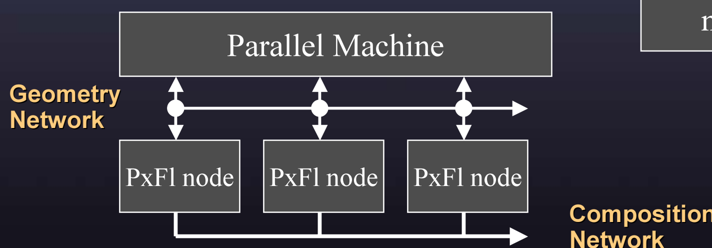
Transparency

- A problem with image composition
- We implemented
  - Screen door when antialising
  - Mammen's algorithm with transparent polygons on shading nodes
- Still, sort-last is not good when many polygons are transparent

Immediate vs. Retained Mode
(direct vs. indirect rendering)

- Immediate-mode limited by host to graphics bandwidth.
- Display lists stored on nodes
- Our intent was to attach PixelFlow to a parallel machine

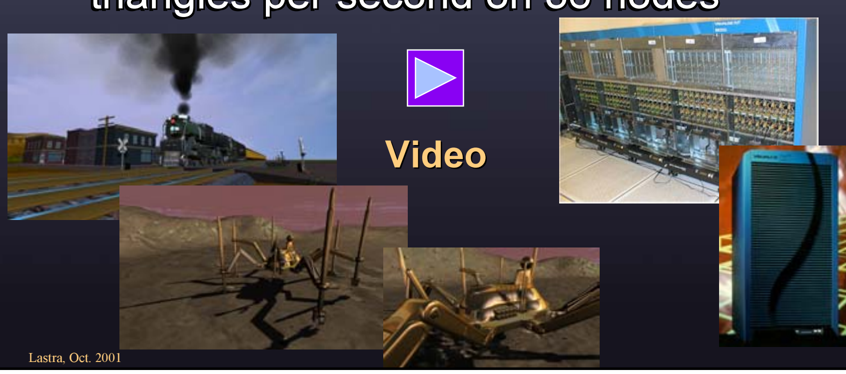




Lastra, Oct. 2001 Slide 41

Performance

- PixelFlow was built with industrial partners, first Division, then HP
- Demonstrated running at about 43 million triangles per second on 36 nodes



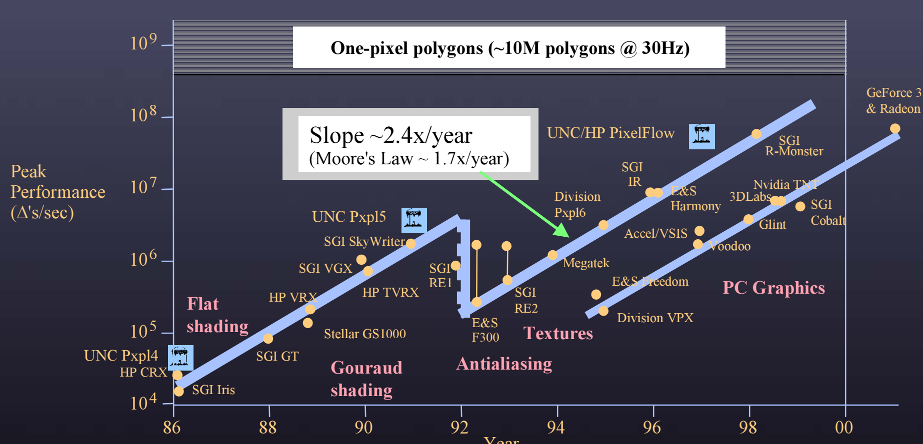
Video

Lastra, Oct. 2001 Slide 42

References

- Steve Molnar, John Eyles, and John Poulton, *"PixelFlow: High-Speed Rendering Using Image Composition,"* SIGGRAPH '92.
- John Eyles, Steven Molnar, John Poulton, Trey Greer, Anselmo Lastra, Nick England and Lee Westover, *"PixelFlow: The Realization",* Siggraph/Eurographics Workshop on Graphics Hardware, Los Angeles, CA, August 3-4, 1997, 57-68.
- Marc Olano and Anselmo Lastra, *"A Shading Language on Graphics Hardware: The PixelFlow Shading System",* Proceedings of SIGGRAPH 98, pp. 159-168, Orlando, Florida, July 19-24, 1998.

Graphics Performance



Graph courtesy of Professor John Poulton

Why Use Polygons At All?

- Pros
  - –Convenient for modeling (by hand)
  - –Good representation when large on screen
  - –Useful for man made objects
- Cons
  - –Huge number to model natural scenes
  - –Fairly complex to render

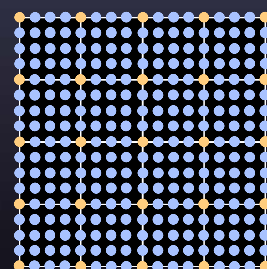
WarpEngine

- Hardware architecture for rendering from depth images
- Voicu Popescu
  - –Also John Eyles,
  - –Josh Steinhurst

Kamov Video

Rendering Algorithm

- WarpEngine algorithm
  - –Interpolate between *reference* image samples
  - –Warp (transform) them forward to image space
  - –Z-composite into sub-pixel (2x2) *warp buffer*
- No interpolation across “skins”

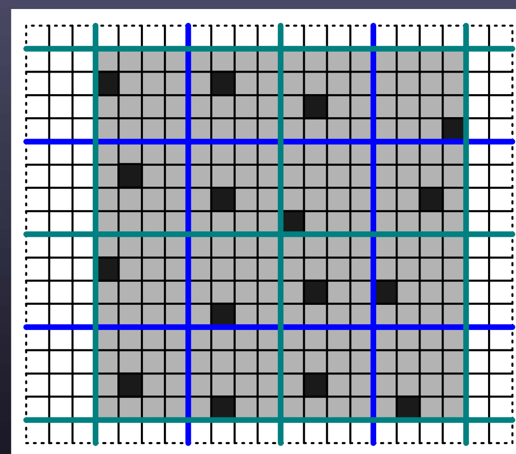


Forward vs. Backward Map

- Conventional scan conversion
  - –For each pixel, compute color
  - –Basically backward map
- WarpEngine
  - –Warp sample forward

Offsets Make it Work

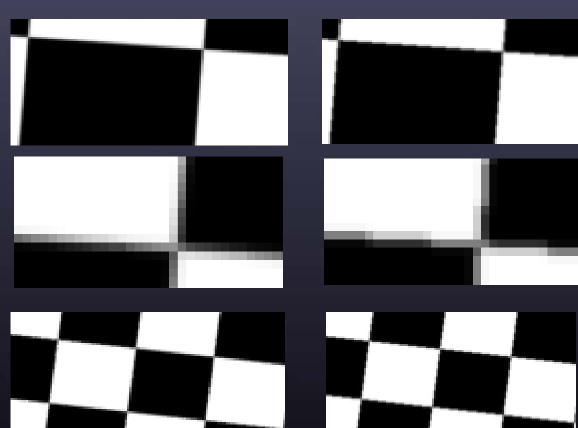
- 2-bit offset
- More precise sample location
- 2-pixel wide filter kernel
- Similar to sparse buffer



Blue – pixel, Green – warp buffer, Black - offset

Inexpensive Antialiasing

2 x 2 Offset | No Offset

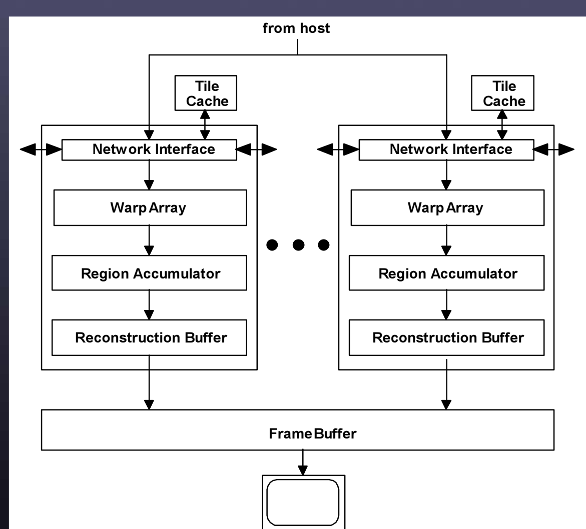


Zoomed

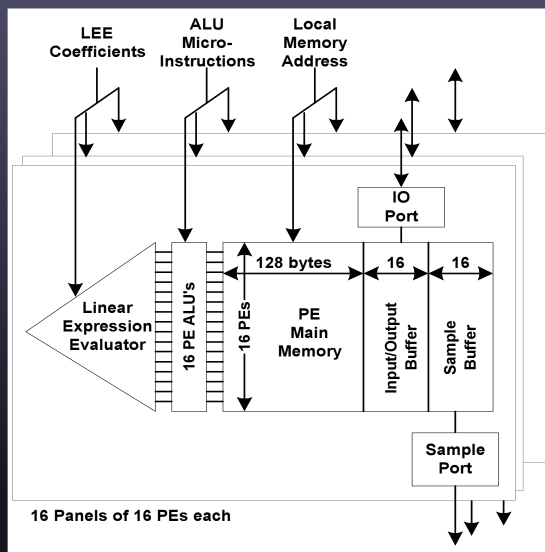
Why Forward Map?

- Low setup cost!
  - –No edge-expression computation
- Exploits coherence
  - –IBR tile (16x16 image) tends to need same interpolation factor
  - –Can use efficient SIMD warper

Architecture

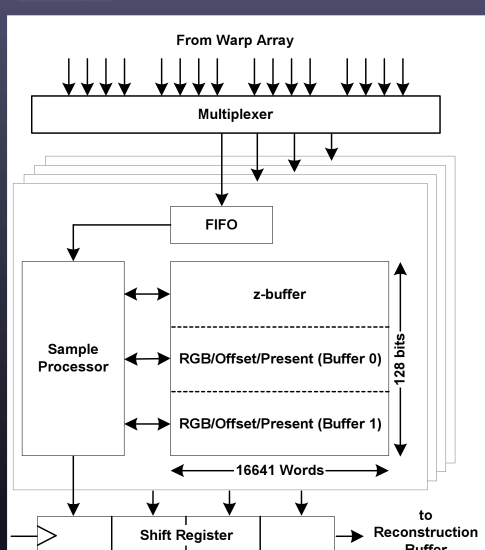


WarpArray



- Nearest neighbor connectivity
- In/Out/Warp pipelined
- Similar to PixelFlow design

Region Accumulator



- Pixel interleaved
- 128 x 128
- Soft z?
- Reconstruction pipelined with next region rendering

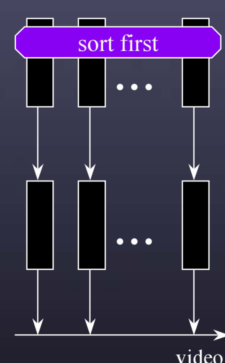
Sort First for Parallelism

- How to distribute work across chips?
- Sort by screen space regions
  - 128x128 pixel region
- Sort First [Mueller] refers to sorting primitives as soon as possible
- Tile coherence lowers overlap factor

Lastra, Oct. 2001 Slide 55

Sort First

- At first glance, like sort middle.
- Advantage: only primitives that move to other screen regions need to be transferred
- <u>May scale better in display size</u>
- Difficulties in memory access and editing



Lastra, Oct. 2001 Slide 56

Expected Chip Specs

- ASIC 12x16 mm
  0.18 micron
- $\geq$ 300 MHz
- 4-node VGA
- 32-node HDTV
- Each chip
  - 100M Samples/sec
  - 4.8G Bytes/sec bandwidth

Simulation on video

Lastra, Oct. 2001 Slide 57

References

- Voicu Popescu, John Eyles, Anselmo Lastra, Josh Steinhurst, Nick England and Lars Nyland, *WarpEngine: An Architecture for the Post-Polygonal Age*, Proceedings of SIGGRAPH 2000, New Orleans, July 2000, 433-442
- Voicu Popescu, Anselmo Lastra, John Eyles, Sort-First Parallelism for Image-Based Rendering, *Proceedings of Eurographics Workshop on Parallel Graphics and Visualization,* Girona, Spain, September 2000.

Lastra, Oct. 2001 Slide 58

Future

- Programmable shading (finally happening!)
  - More memory
  - Floating point
  - Regular architecture?
- Take advantage of small primitives
- Support for image-based primitives?
- Big displays with lots of pixels!

Credits

Pixel-Planes 4: Greg Abram, John Austin, Fred Brooks, Vern Chi, John Eyles, Henry Fuchs, Eric Grant, Trey Greer, Jack Goldfeather, Scott Hennes, Jeff Hultquist, Mark Monger, John Poulton, Susan Spach, John Thomas

Pixel-Planes 5: Mike Bajura, Andrew Bell, Brad Bennett, Jeff Butterworth, Vern Chi, Jon Cohen, Grant Cooper, Claire Durand, David Ellsworth, Niall Emmart, John Eyles, Henry Fuchs, Howard Good, Edward "Chip" Hill, Sonya Holder, Linda Houseman, Heather Kerns, Victoria Interrante, Roman Kuchkuda, Subodh Kumar, Anselmo Lastra, Jonathan Leech, Jeff Mauldin, Steven Molnar, Carl Mueller, Ulrich Neumann, Marc Olano, Sherry Palmer, Mark Parris, John Poulton, John Rhoades, Eddie Saxe, Andrei State, Brice Tebbs, Steve Tell, John Thomas, Russ Tuck, Greg Turk, Sharon Walters, Laura Weaver

PixelFlow (UNC): Daniel Aliaga, Vern Chi, Jon Cohen, Grant Cooper, Lynne Duncan, Nick England, John Eyles, Henry Fuchs, Arthur Gregory, Paul Hilts, Vincent Illiano, Kurtis Keller, Paul Keller, Lawrence Kesteloot, Anselmo Lastra, Jon Leech, David McAllister, Jon McAllister, Peter McMurry, Steven Molnar, Bryon Nordquist, Michael North, Marc Olano, John Poulton, Greg Pruett, Voicu Popescu, Jiang Qian, Jason Smith, Steve Tell, John Thomas, Greg Welch, Yulan Wang, Rob Wheeler

PixelFlow (Division and Hewlett-Packard): Byron Alcorn, Greg Allen, Peter Arnold, Phil Atkin, John Austin, Randy Bailey, Tony Barbour, Tony Bennett, Dave Bradley, John Cook, John Dykstal, Trey Greer, Charles Grimsdale, Bill Hale, Larry Harrold, Fred Heaton, Justin Heinecke, Kenny Hoff, Maureen Hoffert, Rich Holloway, Mary Jo Janowicz , Paul Keller, Kevin Koehler, Roman Kuchkuda, Paul Layne, Ray McConnell, Mike Meyers, Chris Pratico, Brad Ritter, John Rhoades, Durward Rogers, Gary Schaffer, Doug Schiff, Rex Seader, Robert Seward, Mel Snyder, Herman Towles, Jim Weaver, Lee Westover, Robert Yockey

WarpEngine: Gary Bishop, Nick England, John Eyles, Anselmo Lastra, Lars Nyland, Voicu Popescu, Josh Steinhurst

Funding: DARPA and NSF. Additional support from Division, Intel, Hewlett Packard.