The Pixel-Planes Family of Graphics Architectures

Pixel-Planes Technology

Processor Enhanced Memory

- **processor and associated memory on same chip**

SIMD operation

- **Each pixel has an associated processor**
- **Perform rasterization in parallel for each primitive**

Expression evaluation hardware

- **Allows efficient evaluation of quadratic or linear expression for all processors**

Rendering a Triangle with Pixel-Planes

Disable pixel processors outside the triangle

Disable pixels with Z closer than triangle

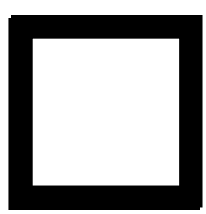
Compute interpolated R,G,B for each pixel

Compute interpolated Nx, Ny, Nz

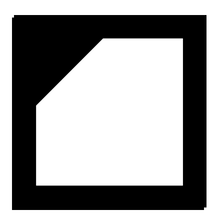
Compute interpolated U,V for each pixel

Later: perform shading calculations for all pixels

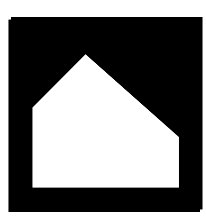
Disabling pixels outside triangle



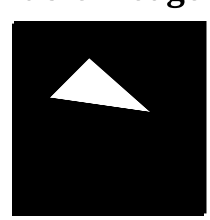
set enable



enable = edge1



enable = edge2



enable = edge2

Linear Expressions

Each edge expressed as linear expression

- Ax + By + C

Enable bit gets true or false based on sign of result at each pixel

Depth test computes depth value at each pixel using LE and compares to current depth value

Each color, normal, texture coordinate component also evaluated as LE

Pixel-Planes 5: A Sort Middle Graphics Architecture

Pixel-Planes 5 Design Criteria

Performance

- Render over 1 M Phong-shaded triangles per second (eventually achieved over 2 M)
  - Demonstrated in 1991

Generality

- No specialized hardware for triangles only
- Allow non-triangle-based applications
  - Curved surfaces, volume rendering, constructive solid geometry, etc.
- Enable research in new algorithms for rendering, shading, etc.

Pixel-Planes 5 Renderer

128x128 SIMD array per renderer board

- 16,384 processors

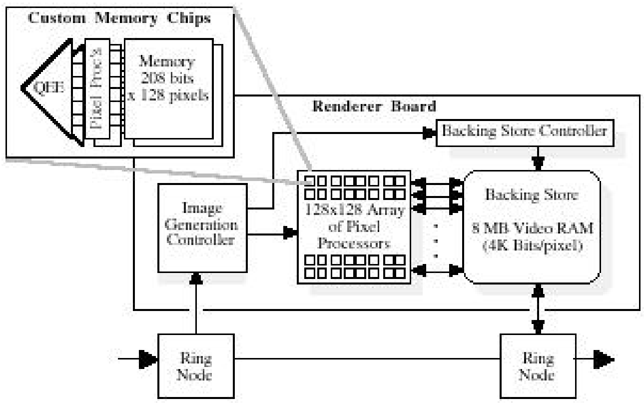
20 renderer boards (approx.) in full system

- Over 300,000 processors!

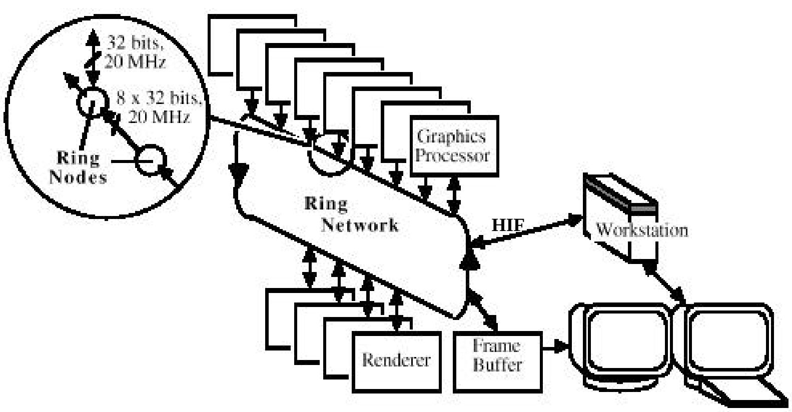
Memory

- 208 bits of local memory per processor
- 4,096 bits of off-chip backing store per processor

Pxpl5 Renderer Board

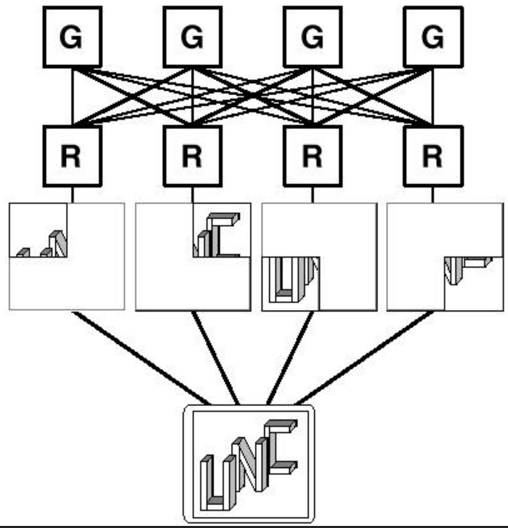


Overall System



Johns Hopkins Department of Computer Science
Course 600.460: Virtual Worlds, Spring 2000, Professor: Jonathan Cohen

Screen Subdivision (Pxpl5)



Graphics Processors

General purpose processors (Intel i860)

Primitives assigned “randomly”

For each primitive

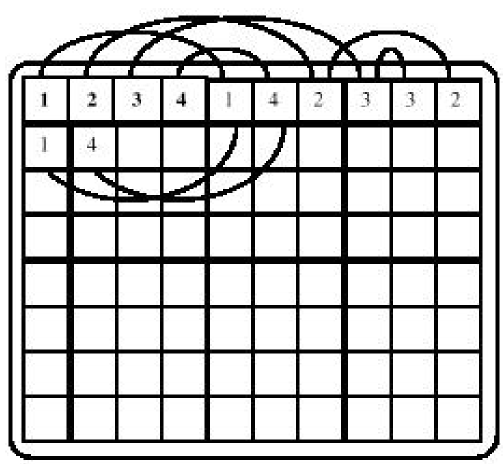
- Transform
- Generate renderer commands
- “Binitize”

Send bins to appropriate renderers after complete database traversal

Renderers

- Assigned to one screen region at a time
- Perform commands from each GP for that region
- Copy resulting colors to backing store
  - eventually copied from Renderer to Frame Buffer
- Begin working on next assigned region

Dynamic Renderer Allocation



Johns Hopkins Department of Computer Science
Course 600.460: Virtual Worlds, Spring 2000, Professor: Jonathan Cohen

Deferred Shading

Store parameter values for each pixel while a region is being rasterized

After all primitives rasterized, perform lighting/shading

Shading performed once for entire region

- independent of number of primitives in region

Programming Levels

PPHIGS

- Dominant graphics API at the time (before OpenGL)

Rendering Control

- Knows about synchronizing GPs

ROS (ring operating system)

- Allows basic communications between GPs

Types of Applications

Standard triangle/sphere graphics using PPHIGS API

- some procedural shading

Volume Rendering

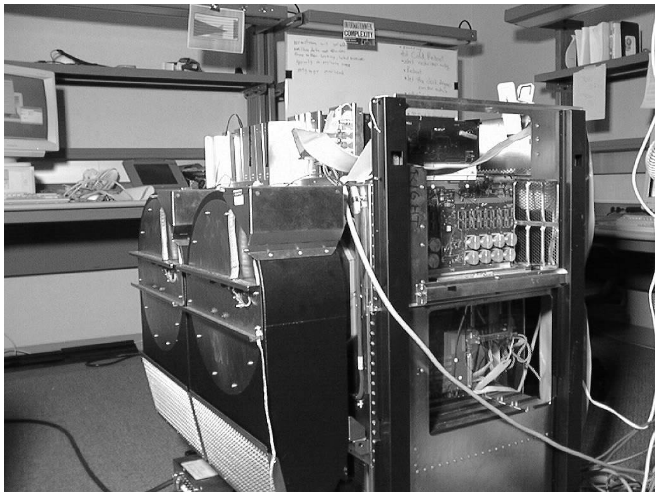
CSG

Julia set

PixelFlow: A Sort-Last Parallel Graphics Machine

Johns Hopkins Department of Computer Science
Course 600.460: Virtual Worlds, Spring 2000, Professor: Jonathan Cohen

Two-Rack PixelFlow Machine



Design Criteria

Tens of millions of triangles/sec throughput

Linearly scalable performance

Programmable shading

Johns Hopkins Department of Computer Science
Course 600.460: Virtual Worlds, Spring 2000, Professor: Jonathan Cohen

PixelFlow Architecture

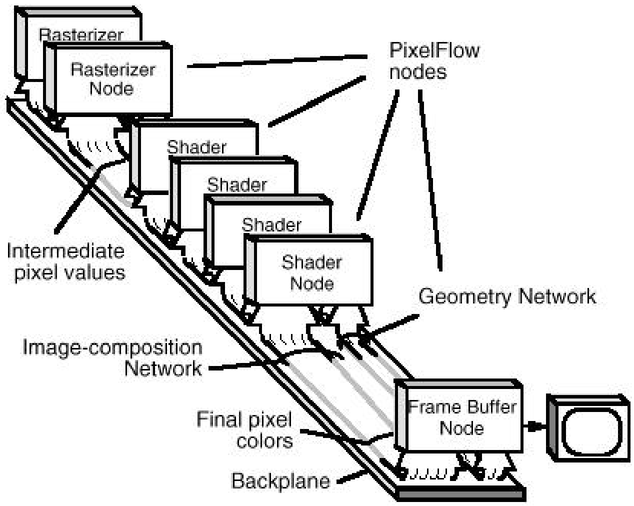
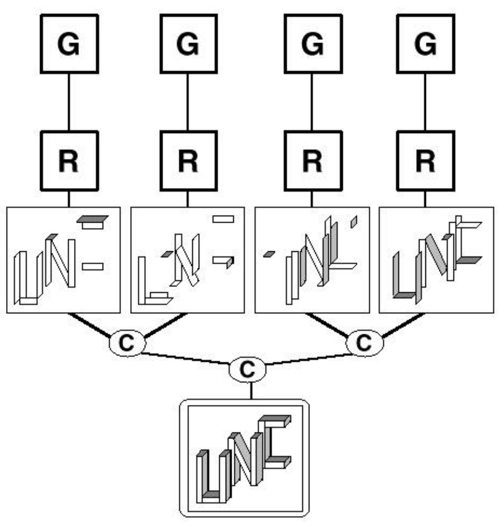


Image Composition



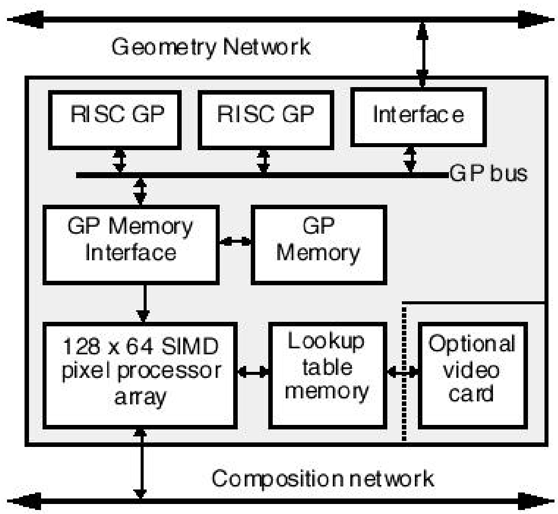
Johns Hopkins Department of Computer Science
Course 600.460: Virtual Worlds, Spring 2000, Professor: Jonathan Cohen

PixelFlow Board

Boards have same hardware components
- 2 PA-RISC processors
  - —transform, generate SIMD commands
- Shared processor memory
- 128x64 SIMD Array (8-bit ALU)
  - —perform pixel operations
- Texture Memory (64 MB per board)
  - —not cost effective?

Board function chosen by application
- Renderer
- Shader
- Frame buffer (requires daughter card)

Board Diagram



Johns Hopkins Department of Computer Science
Course 600.460: Virtual Worlds, Spring 2000, Professor: Jonathan Cohen

Actual PixelFlow Board

Renderer Board

Operates on subset of geometry

PA-RISC

- **Stores display lists of static geometry**
- **Transforms geometry**
- **Generates/binitizes SIMD commands**

EMC (Enhanced Memory Chip)

- **Enable primitives pixels**
  - **—including setting Z**
- **Set shader id**
- **Load/interpolate parameters**
  - **—colors, normals, texcoords**
  - **—other arbitrary shader parameters**

Shader Board

Operates on one particular screen region

PA-RISC

- **Generate/cache shading commands for EMCs**
- **Loop through shader functions**
  - **—Pre-light, light, post-light**

EMCs

- **Perform shading computation**
  - **—Image texture lookup**
  - **—Lighting**
  - **—Programmable shading operations**

Image Composition

One region at a time on renderer boards

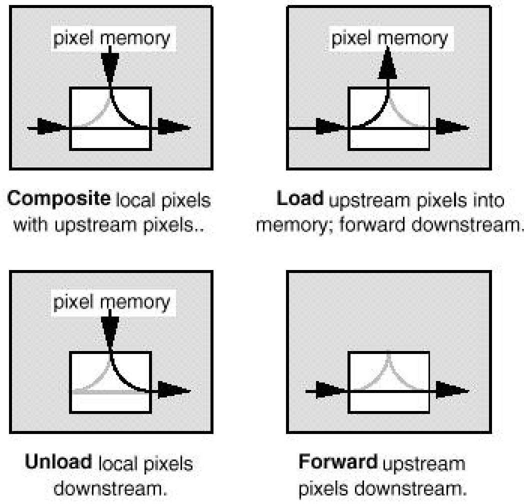
Composite each region, sending to one shader board

Shading boards send results to frame buffer

Composition network

- **100 Gbit/sec bandwidth**
- **Bidirectional signaling hardware**

Compositor Operating Modes



Programmable Shading

Procedural shading

High-level language for programming

- Modified RenderMan language

Shading compiler generates C-code for storing EMC commands on PA-RISC

256 bytes of local memory per pixel

(show Olano SIGGRAPH 98 video)

API

Modified OpenGL

Added support for programmable shading

Added frame synchronization commands

Restrictions apply

OpenGL on PixelFlow

Application runs on host machine

Global state changes broadcast to all boards

- lights, matrices, etc.

Primitives (glBegin/glEnd blocks) distributed round-robin among renderer boards

Textures loaded/replicated across all shader boards

OpenGL Extensions

- Load/instance shader function
- Set current shader
- glMaterial extended to arbitrary shader parameters
  - global attribute state stores arbitrary parameters as well as built-ins (color, coord, etc.)
  - named shader parameters may be shared among different shader functions
- Frame synchronization commands

OpenGL Restrictions

- No global state changes within glBegin/glEnd
  - changes within glBegin/glEnd sent to a single Renderer, not broadcast
- Cannot read back frame buffer during frame
  - Frame buffer not complete until composited and shaded at end of frame
- Primitive ordering not currently guaranteed
  - bad for geometry-based decals (e.g. runway stripes)

Commercialization

Pixel-Flow originally developed in collaboration with Division and later Hewlett Packard

- Visualize PxFl product dropped by HP just before production

PC card product developed by PixelFusion

- Products due to ship 2nd quarter 2000