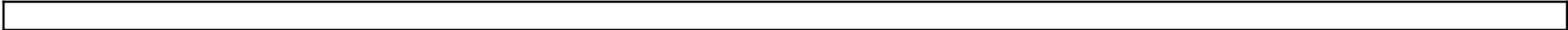




3D Graphics Hardware

15-463 Graphics II
Spring 1999



Topics

- n Graphics Architecture
- n Uniprocessor Acceleration
- n Front-End Multiprocessing
 - Pipelined
 - Parallel
- n Back-End Multiprocessing
 - Pipelined
 - Parallel



Graphics Architecture



Programming Interface

- n *Database* contains primitives from model
- n Immediate mode
 - Program calls library with individual primitives
 - Need to enumerate entire database for each frame
 - No temporal coherence between frames
- n Retained mode
 - Primitives are remembered between frames
 - Program calls library to make incremental changes
 - Can potentially exploit temporal coherence

Frame Buffer Organization

- n Dedicated bank of RAM for pixel values
 - Components are rgb, alpha, z, etc.
- n CPU or display processor writes pixels
- n Video controller reads pixels
 - Generates video signal for monitor
- n Contention between CPU/DP and video controller

DRAM Organization

- n Read or write just a few bits at a time
- n Multiple chips in parallel for more bits
- n Bits organized internally in a 2D grid
- n *Row select* and *column select* identify particular bits
 - Row select has extra set-up latency
 - No penalty for column select
 - n *Page mode* chips exploit this

Video RAM (VRAM) (1983)

- n Extra bit port for video signal generation
 - Avoids contention with video controller
- n Buffer loaded during “stolen” memory cycle
 - Whole row can be copied in a single cycle
- n Multiple banks needed for big frame buffers
 - Single chip can’t keep up with video signal



Uniprocessor Acceleration

Peripheral Display Processor

- n Reads/writes frame buffer in place of CPU
- n Acceleration for 2D operations
- n Read/write many pixels in parallel
- n Built-in support for lines, circles, bitblt, etc.
- n No transformations, floating-point, etc.
- n Now limited only by speed of memory

CPU Extensions

- n Additional instructions in CPU for 3D graphics
- n Pioneered by i860 (1989)
 - 64-bit pixel operations: linear interpolation, z-buffer compare, conditional update
- n MMX (1995?)
 - Reinterprets x86 floating-point “registers”
 - Up to 8 simultaneous integer operations
 - n 8x8bit, 4x16bit, 2x32bit, 1x64bit
 - n (Clamped) arithmetic, logical, pack/unpack

3DNow (1998)

- n Reinterprets MMX registers
- n Two simultaneous floating-point operations
 - Add, subtract, multiply, min, max, etc.
 - Reciprocal and reciprocal square root approximation
 - n Three instructions implement Newton's method
 - n On-chip table lookup for initial approximation
 - n Two successive iterations to refine approximation
- n Superscalar execution permits 4 operations/cycle

Performance Barriers

- n Floating-point geometry processing
 - Transformation, clipping, lighting
- n Integer pixel processing
 - Scan conversion
- n Frame-buffer memory bandwidth
 - Z-buffer comparison, shading, alpha blending
- n What can we do about these?



Front-End Multiprocessing

Pipelining

- n Extra processors for distinct tasks
- n Primitives are processed in discrete stages
 - Need result of stage i to compute stage $i+1$
- n Process stage i of primitive j concurrently with stage $i+1$ of primitive $j+1$
- n Increase in throughput, (small) increase in latency
 - Individual latency usually isn't important

Parallelism

n Extra processors for uniform tasks

n **SIMD**

- One instruction affects multiple operands
- “Disable bit” for conditional operations
- Advantage: no extra control logic
- Disadvantage: inflexible for non-uniform tasks
- Examples: MMX, 3DNow

n **MIMD**

- Each processor has its own instruction stream
- Synchronization/interconnect can be difficult

Pipeline Front End

- n Many distinct stages naturally lead to pipelining
- n CPU usually handles display traversal
- n Geometry subsystem is floating-point intensive
 - Transformations can be done in parallel
 - n By component
 - n By vector (vertex or normal)
 - Transformation bundled with trivial accept/reject
 - One processor per clip plane
 - Division by w is expensive
 - n Compute $1/w$ and multiply x , y , and z by that

Geometry Engine (1982)

- n Used in early SGI machines
- n ~One chip per stage of front-end pipeline
 - Configuration word determines which stage
- n Each chip reads and writes command stream
 - Vertices inserted and deleted during clipping
- n Replaced by commodity chips in later SGIs
 - Weitek 3332 and Intel i860

Parallel Front End

- n Significantly harder than pipelining front end
 - Need to recombine streams at back end for ordered rendering algorithms
 - Processor contention over shared database
- n RealityEngine processes geometry in parallel
 - Command processor sends primitives to geometry engines in round robin order
 - Geometry engine is not pipelined (single i860)
 - Triangle bus broadcasts transformed triangles to back-end processors



Back-End Multiprocessing



Taxonomy

- n Object order: outer loop over objects
 - Z-buffer, depth-sort, and BSP-tree algorithms
- n Image order: outer loop over pixels
- n Image parallel = parallel object order
 - Parallel inner loop over image pixels
 - Partitioned image, logic-enhanced memory
- n Object parallel = parallel image order
 - Parallel inner loop over objects
 - Processor-per-primitive, tree-structured

Pipelined Object Order

- n Polygon-edge-span processor pipeline
 - Polygon unit finds x , z , and rgb deltas for each edge
 - Edge unit computes left/right x , z , and rgb bounds/deltas for each span
 - Span unit interpolates z and rgb for each pixel
 - n Implements z buffer compare
- n Poor load balance: span processor is bottleneck
 - Polygons*edges*spans*pixels
 - Pixel cache to increase memory bandwidth
 - n Excellent locality

Pipelined Image Order

n Scan-line pipeline

- *Y sorter* finds first scan line of each polygon
- *Active segment sorter* sorts segments on scan line by x
 - n A *segment* is a span of a single polygon
- *Visible span generator* compares z values of segments
- *Shader* computes rgb values of visible spans

n No frame buffer if hardware is fast enough (!)

n Again, poor load balance: shader is bottleneck

- Need more than 1 processor/stage: parallel processing

Parallel Object Order/Image Parallel

- n Multiple processors render pixels in parallel
 - Predominant back-end architecture
- n Contiguous partition
 - Each processor handles a specific block of pixels
 - Don't look at primitives outside my region
- n Interleaved partition
 - Each processor handles every n th pixel
 - Must look at every primitive
 - Worst case isn't as bad, best case isn't as good
 - Predominant architecture

Parallel Object Order/Image Parallel

n SIMD

- All processors work on same $n \times n$ pixel block
- Single control logic for all processors
- Many conditional branches (complex algorithms) result in poor utilization

n MIMD

- Processors can work on pixels in different blocks
- Requires control logic for each processor
- Better utilization, but needs fast interconnect

RealityEngine (1993)

- n Third-generation SGI graphics architecture
- n Processing units
 - *Command processor* breaks commands into primitives
 - *Geometry engines* (i860s) transform, light, and clip
 - *Triangle bus* broadcasts transformed triangles
 - *Fragment generators* scan convert triangles into subsample masks and single rgb and z
 - n Traverses vertically: every 5th column
 - *Image engines* combine subsamples in local memory
 - *Display generator* assembles colors from subsamples

Logic Enhanced Memory

- n Massively parallel approach
- n Processing elements built into memory chips
 - Overcomes memory bandwidth limitations
- n Interconnect between chips can be difficult
- n Custom logic increases cost of chips
 - Ordinary DRAM has vast economy of scale

Pixel Planes (1981)

- n 1-bit processor associated with each pixel
 - Enable bit for pixels inside polygon
- n *Linear expression tree* evaluates linear equations in parallel
 - $F(x, y) = Ax + By + C$
 - Edges of convex polygon
 - Depth value of points in a triangle
 - Color components for Gouraud shading
- n Constant rendering time for any size polygon!

Parallel Polygon Rasterization

- n Used by RealityEngine fragment generators
- n *Edge functions* similar to Pixel Planes linear expressions
- n $n \times n$ block of pixel logic traverses frame buffer
- n Traversal algorithm affects performance
 - Vertex bounding box (simplest)
 - Edge-to-edge: search for left edge at each scan line
 - Center line: search left and right from interior
- n Can clip just by adding edges

Parallel Image Order/Object Parallel

- n Specific primitives assigned to each processor
 - Usually 1 primitive per processor
- n Fragments combined from each processor output
- n Processor-per-primitive pipeline
 - My $z >$ previous z : pass my rgb and z to next
 - My $z <$ previous z : pass rgb and z unchanged
- n Processor-per-primitive tree
 - Merge rgb and z streams using binary tree
- n Performance drops if $\#primitives > \#processors$

Talisman (1996)

- n Microsoft design for PC 3D hardware
- n Retained mode API
- n Distinct *image layers* for independent objects
- n Image layers are composited during video signal generation
- n Layer transformations for simple effects
- n Layers are compressed by hardware

References

- n Foley and van Dam, Chapter 18
- n Kurt Akeley, “RealityEngine Graphics”, *SIGGRAPH '93*
- n Kurt Akeley, “High-Performance Polygon Rendering”, *SIGGRAPH '88*
- n James H. Clark, “The Geometry Engine: A VLSI Geometry System for Graphics”, *SIGGRAPH '82*
- n Juan Pineda, “A Parallel Algorithm for Polygon Rasterization”, *SIGGRAPH '88*
- n Jay Torborg and James T. Kajiya, “Talisman: Commodity Realtime 3D Graphics for the PC”, *SIGGRAPH '96*