

IMAGE-COMPOSITION ARCHITECTURES FOR REAL-TIME IMAGE GENERATION

by

Steven Edward Molnar

A Dissertation submitted to the faculty of The University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill

1991

Approved by:

_____ Advisor
Henry Fuchs

_____ Reader
John W. Poulton

_____ Reader
T. Gary Bishop

John G. Eyles

© 1991
Steven E. Molnar
ALL RIGHTS RESERVED

STEVEN EDWARD MOLNAR. Image-Composition Architectures for Real-Time Image Generation (under the direction of Henry Fuchs).

ABSTRACT

This dissertation describes a new approach for high-speed image-generation based on image compositing. Application software distributes the primitives of a graphics database over a homogeneous array of processors called renderers. Each renderer transforms and rasterizes its primitives to form a full-sized image of its portion of the database. A hardware datapath, called an image-composition network, composites the partial images into a single image of the entire scene.

Image-composition architectures are linearly scalable to arbitrary performance. This property arises because: 1) renderers compute their subimages independently, and 2) an image-composition network can accommodate an arbitrary number of renderers, with constant bandwidth in each link of the network. Because they are scalable, image-composition architectures promise to achieve much higher performance than existing commercial or experimental systems. They are flexible as well, supporting a variety of primitive types and rendering algorithms. Also, they are efficient, having approximately the same performance/price ratio as the underlying renderers.

Antialiasing is a special challenge for image-composition architectures. The compositing method must retain primitive geometry within each pixel to avoid aliasing. Two alternatives are explored in this dissertation: simple z-buffer compositing combined with supersampling, and A-buffer compositing.

This dissertation describes the properties of image-composition architectures and presents the design of a prototype z-buffer-based system called PixelFlow. The PixelFlow design, using only proven technology, is expected to render 2.5 million triangles per second and 870 thousand antialiased triangles per second in a two-card-cage system. Additional card cages can be added to achieve nearly linear increases in performance.

ACKNOWLEDGEMENTS

I am thankful to Henry Fuchs, John Poulton, and the other members of my committee, for their support and encouragement during the entire dissertation process. John Poulton and John Eyles, in particular, contributed to many of the ideas described here, especially the PixelFlow architecture, described in Part II. John Poulton, Gary Bishop, Fred Brooks, and Turner Whitted, read drafts of the dissertation and suggested many needed improvements. Marc Levoy, a member of my committee before leaving for Stanford, helped set the initial direction for this research.

My father, Charles Molnar, read and critiqued an early draft of this dissertation, providing a valuable outside perspective.

The Pixel-Planes group, led by Henry Fuchs and John Poulton, provided a stimulating and fruitful environment for this research. I am particularly indebted to Trey Greer, David Ellsworth, Brice Tebbs, and Greg Turk for many creative discussions

I am grateful to IBM, DARPA, and NSF for financial support.

Finally, I am grateful to my wife, Wanda, and daughter, Jennifer, for their love and patience, which made this work possible.

TABLE OF CONTENTS

	Page
LIST OF FIGURES	ix
LIST OF ABBREVIATIONS.....	xiii
Chapter	
1 INTRODUCTION.....	1
1.1 Realistic Images in Real Time.....	1
1.2 Multiprocessor Graphics	1
1.3 Image Composition	2
1.4 Thesis Summary and Contribution of this Work.....	2
1.5 Organization of this Dissertation.....	3
PART I: IMAGE-COMPOSITION ARCHITECTURES	5
2 REAL-TIME IMAGE GENERATION	7
2.1 Objectives for Real-Time Systems.....	7
2.1.1 Screen Resolution	7
2.1.2 Color Fidelity	8
2.1.3 Lighting and Shading	8
2.1.4 Antialiasing	8
2.1.5 Frame Rate	9
2.1.6 Latency	9
2.1.7 Scene Complexity.....	9
2.1.8 Sophisticated Effects	9
2.2 The Rendering Task	9
2.2.1 Standard Rendering Pipeline	10
2.2.2 Computational Requirements	10
2.3 Multiprocessor Architectures	10
2.3.1 Classes of Multiprocessor Architectures	11
2.3.2 Sort Middle (Screen-Space Subdivision)	12
2.3.3 Sort First (Dynamic Database Redistribution)	13
2.3.4 Sort Last (Image Composition)	14
2.3.5 Comparison of Approaches	15
3 IMAGE COMPOSITION	17
3.1 Previous Work	17
3.1.1 Processor-per-Primitive Systems	17
3.1.2 Video-Overlay Systems.....	18
3.1.3 Image-Composition Algorithms.....	18
3.1.4 Large-Grain Image-Composition Systems.....	19
3.1.5 Contribution of this Work	19
3.2 Fixed-Priority Image Composition.....	19
3.2.1 Chroma Key	19

3.2.2	RGB α Overlay	20
3.3	Pixel-Priority Image Composition	23
3.3.1	The Aliasing Problem	23
3.3.2	A Simple Pixel-Priority System	25
4	SOLVING THE ALIASING PROBLEM	29
4.1	Antialiasing Approaches	29
4.1.1	Supersampling	29
4.1.2	A-Buffer	30
4.2	Supersampling Architectures	32
4.2.1	The Factor of k	32
4.2.2	Computing Additional Samples	33
4.2.3	Increasing the Bandwidth	33
4.3	A-Buffer Architectures	34
4.3.1	Applying the A-Buffer to Image Composition	34
4.3.2	Minimizing Artifacts	35
4.3.3	A-Buffer Renderers	40
4.3.4	A-Buffer Compositors	41
4.3.5	Pixel Assembler	42
5	ANALYSIS OF IMAGE-COMPOSITION APPROACHES	43
5.1	Tree vs. Pipeline	43
5.1.1	Balanced Tree	44
5.1.2	Pipeline	44
5.2	Frame Rate and Latency	45
5.2.1	Latency in a Simple Z-Buffer System	45
5.2.2	Latency in a Supersampling System	46
5.2.3	Latency in an A-buffer System	47
5.3	Static Load Balancing	48
5.3.1	Database Distribution	49
5.3.2	Retained-Mode Traversal	49
5.3.3	Hierarchical Display Structures	50
5.3.4	Immediate-Mode Traversal	52
5.4	Dynamic Load Balancing	52
5.4.1	Z-Buffer Systems	52
5.4.2	A-Buffer Systems	52
5.4.3	Dynamic Load-Balance Ratio	53
5.5	Advanced Rendering Algorithms	54
5.5.1	Deferred Shading	55
5.5.2	Transparency	55
5.5.3	Textures	56
5.5.4	Volume Rendering	57
5.6	Taxonomy of Image-Composition Architectures	58
5.7	Scalability and Economics	59
	PART II: A PROTOTYPE SYSTEM DESIGN	63

6	PIXELFLOW: A PROTOTYPE IMAGE-COMPOSITION SYSTEM	65
6.1	Design Objectives	66
6.2	Architectural Decisions	66
6.2.1	Z-buffer Rendering with Supersampling Antialiasing	66
6.2.2	Pixel-Planes 5-Style Renderers	67
6.2.3	Wide, Slow Image Composition Network	67
6.2.4	Support for Deferred Shading	68
6.2.5	Buffering for Multiple Regions	68
6.3	PixelFlow System Overview	69
6.3.1	Host and Host Interface	70
6.3.2	Message-Passing Network	71
6.3.3	Renderer/Shaders	72
6.3.4	Image-Composition Network	74
6.3.5	Frame Buffer	75
6.3.6	Algorithms and Performance	77
7	IMAGE-COMPOSITION NETWORK	79
7.1	Data Path	79
7.1.1	Compositor PLD	79
7.1.2	Wiring Delays and Clock Distribution	80
7.1.3	Region-by-Region Transfers	81
7.2	Control Path	81
7.2.1	Ready and Go Chains	81
7.2.2	Controlling the Compositors	83
7.3	Performance Model	83
7.3.1	Simple Rendering Algorithm	83
7.3.2	Shaders and Supersampling	84
8	RENDERER/SHADER BOARD	87
8.1	Graphics Processor	88
8.1.1	Intel i860 Microprocessor	88
8.1.2	VRAM Memory System	89
8.1.3	Message-Passing Network Interface	90
8.1.4	Status and Command Registers	91
8.1.5	IGC Input FIFOs	91
8.1.6	Graphics-Processor Algorithm	92
8.2	Rasterizer	93
8.2.1	Input FIFOs and Stream Parser	94
8.2.2	Image Generation Controller	95
8.2.3	EMC Array	96
8.2.4	Rasterizer Algorithm	97
8.3	Performance Model	99
8.3.1	Bin Classification	99
8.3.2	Renderer Performance	101
8.3.3	Shader Performance	102

9	CONTROL ALGORITHMS AND PERFORMANCE.....	103
9.1	Synchronization and Control.....	103
9.1.1	Rasterizer/Compositor Synchronization.....	103
9.1.2	Rendering Recipes	106
9.2	Dynamic Load Balance	108
9.3	Estimated Performance.....	109
9.3.1	Simulation Approach.....	109
9.3.2	Scalability	110
9.3.3	Results for Sample Datasets	111
10	CONCLUSION	115
10.1	Future Research.....	115
10.2	Toward Implementation	115
Appendix		
A	STATISTICS FOR SAMPLE DATASETS.....	117
A.1	Space Station and Shuttle	119
A.2	Poliovirus.....	121
A.3	Radiosity Lobby	123
A.4	House Interior	125
A.5	Earth	127
A.6	Pipes	129
B	COST ESTIMATES FOR THE PROTOTYPE SYSTEM	131
B.1	Card-Cage Components	131
B.2	Renderer/Shader Components	132
B.3	Host Interface Components	132
B.4	Frame Buffer Components	133
B.5	Total System Cost.....	133
B.5.1	One Card-Cage Configuration.....	133
B.5.2	Two Card-Cage Configuration	133
	REFERENCES	135

LIST OF FIGURES

Figure 2.1:	Standard rendering pipeline for z -buffer rendering of Gouraud or Phong-shaded polygons (adapted from [MOLN90]).	10
Figure 2.2:	Canonical multiprocessor graphics system. G units are geometry processors. R units are rasterizers.	11
Figure 2.3:	Three classes of multiprocessor graphics architectures: (a) Sort Middle, (b) Sort First, and (c) Sort Last.	12
Figure 2.4:	Screen-partitioning methods: (a) each processor is assigned a contiguous region of pixels.	12
Figure 3.1:	Block diagram of chroma-key compositor.	20
Figure 3.2:	Chroma-key overlay of three 160x128-pixel images (one for each polygon) over a blue background.	20
Figure 3.3:	Block diagram of one channel of an $RGB\alpha$ compositor.	21
Figure 3.4:	$RGB\alpha$ overlay of images above.	22
Figure 3.5:	Front polygon A and middle polygon B have the same coverage in cases (a), (b), and (c), but expose differing amounts of the background color.	22
Figure 3.6:	Non-associativity of alpha-blend compositing. A , B , and C are polygon fragments that partially cover the pixel (pixel is viewed from the side	23
Figure 3.7:	16 possible coverage cases for Duff's algorithm. Pixel corners are labelled A or B depending on whether z_A or z_B is in front. Labels a and b indicate regions where A or B are assumed visible (figure adapted from [DUFF85]).	24
Figure 3.8:	Coverage ambiguity in Duff's algorithm. Cases (a) and (b) result in the same composited pixel value.	25
Figure 3.9:	Block diagram of a simple pixel-priority compositor.	26
Figure 3.10:	Pixel-priority system for displaying 3.2 million triangles per second.	26
Figure 3.11:	Conventional high-resolution video system	27
Figure 3.12:	Block diagram of a high-speed ECL compositor.	27
Figure 4.1:	Conceptual pipeline of operations performed during supersampling.	29
Figure 4.2:	Space station scene with 1 sample per pixel (upper left), 5 samples per pixel (upper right), 16 samples per pixel (lower left), and 32 samples per pixel (lower right).	30
Figure 4.3:	Sources of high-frequency components in a displayed image: (a) Explicit primitive edge, (b) implicit edge where primitives intersect, and (c) specular highlight.	31
Figure 4.4:	Magnified view of pixel and linked list of fragments for the three surfaces A , B , and C visible at the pixel.	32
Figure 4.5:	Visibility errors caused by sampling z outside of polygon boundaries (scene is viewed from above).	35
Figure 4.6:	A-buffer-rendered image of the two-polygon scene in Figure 4.5. Several pixels along the intersection between polygons are colored pink, whereas only green should be visible.	36
Figure 4.7:	(a) Compact representation of dz/dx and dz/dy . (b) Data format for 4-word fragment.	36

Figure 4.8:	Two-polygon scene rendered using z slope information.	37
Figure 4.9:	Gouraud-shaded image of an X-29 aircraft with color-wraparound artifacts	38
Figure 4.10:	Scene in Figure 4.9 with color clamping.	38
Figure 4.11:	A-buffer rendering of highly specular cylinders (640x512-pixel resolution with 32 samples in the fragment bitmask).....	39
Figure 4.12:	Supersampled rendering of same scene (also 640x512-pixel resolution with 32 samples per pixel in same locations as A-buffer).....	40
Figure 5.1:	Classes of binary trees: (a) balanced tree, (b) mixed tree, (c) left-heavy tree (pipeline).	43
Figure 5.2:	Properties of different composition-network topologies.	44
Figure 5.3:	Rendering time line for a z -buffer image-composition system.	45
Figure 5.4:	Rendering time line for pipelined z -buffer system.	46
Figure 5.5:	Time line for naive supersampling image-composition system.	46
Figure 5.6:	Time line for low-latency supersampling system. ($R_{i,j}$ means render region i , sample j . $C_{i,j}$ means composite region i , sample j).....	47
Figure 5.7:	Time line for simple A-buffer image-composition system.	47
Figure 5.8:	Time line for pipelined A-buffer image-composition system.	47
Figure 5.9:	Time line for A-buffer system with early bin-sorting pass.	48
Figure 5.10:	Hierarchical display structure.	50
Figure 5.11:	Distributing a display structure by scattering.	51
Figure 5.12:	Distributing a display structure by clustering.....	51
Figure 5.13:	Dynamic load imbalance caused by uneven distribution of primitives across the screen.	53
Figure 5.14:	Dynamic load imbalance caused by uneven distribution of primitives across the screen at each renderer.	53
Figure 5.15:	Texturing implemented as a deferred-shading process.	57
Figure 5.16:	Voxel splatted onto image buffer.	58
Figure 5.17:	Taxonomy of image-composition architectures.	59
Figure 5.18:	Performance and price for various systems available in 1991 (including estimates for the prototype system).	60
Figure 5.19:	Performance vs. price for PixelFlow prototype system and representative systems in 1991.	61
Figure 6.1:	Block diagram of the prototype PixelFlow system.	65
Figure 6.2:	Two-card–cage PixelFlow system.....	70
Figure 6.3:	Block diagram of host interface.	71
Figure 6.4:	Pseudocode for host algorithm.	71
Figure 6.5:	Ring node transmit and receive interface.	72
Figure 6.6:	Ring flattened on backplane to reduce wire lengths.	72
Figure 6.7:	Block diagram of renderer/shader board.	73
Figure 6.8:	Connections to a single compositor chip.	74
Figure 6.9:	Bit-serial composition of two pixel streams.....	75
Figure 6.10:	Block diagram of a frame-buffer board.....	76
Figure 6.11:	Demultiplexer PLDs and corner turners.....	76
Figure 6.12:	Assignment of rendering steps to processing units.	77

Figure 7.1:	Timing specifications for the Gazelle 22VP10-7 PLD (from [GAZE88]).	80
Figure 7.2:	Compositor operating modes.	80
Figure 7.3:	Ready/go controller.	82
Figure 7.4:	Ready and go chains at various stages during a transfer.	82
Figure 7.5:	Transfers for system with 4 renderers and 3 shaders, computing 2 samples per pixel.	84
Figure 7.6:	Composition-network performance under varying conditions.	85
Figure 8.1:	Block diagram of a renderer/shader board.	87
Figure 8.2:	Block diagram of the graphics processor.	88
Figure 8.3:	Block diagram of the i860 microprocessor.	89
Figure 8.4:	Four-bank, two-way interleaved memory organization.	90
Figure 8.5:	Interleaved region processing order.	92
Figure 8.6:	Outer-level GP algorithm.	92
Figure 8.7:	GP region-processing algorithm.	93
Figure 8.8:	Block diagram of the rasterizer.	94
Figure 8.9:	Logical diagram of a PixelFlow EMC.	96
Figure 8.10:	Snapshot of the EMC transfer port in operation.	97
Figure 8.11:	Rasterizer algorithm.	98
Figure 8.12:	Geometric construction for evaluating $r(x,y)$.	100
Figure 8.13:	Bin replication factor as function of region size.	100
Figure 8.14:	Predicted and actual bin-replication factors	101
Figure 8.15:	Graph of results in Figure 8.14.	101
Figure 8.16:	Renderer performance model.	102
Figure 9.1:	Synchronization components on a renderer board.	103
Figure 9.2:	Snapshot of renderer in various stages of operation.	105
Figure 9.3:	Rendering recipes to compute a 4-region image without shaders or antialiasing.	106
Figure 9.4:	Rendering recipes for system with 2 shaders and 2-sample-per-pixel antialiasing.	107
Figure 9.5:	Rendering recipes for system with 4 shaders and 3-sample-per-pixel antialiasing.	107
Figure 9.6:	Simulated performance in visible triangles per second for system with 36 renderers and various amounts of buffering.	108
Figure 9.7:	Plot of simulated performance (in visible triangles per second) vs. number of regions of buffering for a 36-renderer system.	108
Figure 9.8:	Simulation results (in visible triangles per second) for systems of various sizes rendering Gouraud-shaded, 1280x1024 images.	111
Figure 9.9:	Rendering performance (visible triangles per second) vs. number of renderers (log/log scale).	111
Figure 9.10:	Simulated rendering performance for 640x512-pixel, Gouraud-shaded images on a 36-renderer system with 4 shaders.	112
Figure 9.11:	Simulated rendering performance for 640x512-pixel, Phong-shaded images on a 36-renderer system with 4 shaders.	112
Figure 9.12:	Simulated rendering performance for 1280x1024-pixel, Gouraud-shaded images on a 36-renderer system with 4 shaders.	113

Figure 9.13:	Simulated rendering performance for 1280x1024-pixel, Phong-shaded images on a 36-renderer system with 4 shaders.	113
Figure A.1:	Image of Space Station and Shuttle database (1280x1024 resolution).	119
Figure A.2:	Image of Poliovirus database (1280x1024 resolution).	121
Figure A.3:	Image of Radiosity Lobby database (1280x1024 resolution).	123
Figure A.4:	Image of House Interior database (1280x1024 resolution).	125
Figure A.5:	Image of Earth database (1280x1024 resolution).	127
Figure A.6:	Image of Pipes database (1280x1024 resolution).	129

LIST OF ABBREVIATIONS

ALU	Arithmetic Logic Unit
BSP	Binary Space Partitioning
CRT	Cathode Ray Tube
DMA	Direct Memory Access
DRAM	Dynamic Random-Access Memory
EMC	Enhanced Memory Chip
FIFO	First-In-First-Out (queue)
GHz	gigahertz (10^9 Hz)
GP	Graphics Processor
Gbit	gigabit (10^9 or 2^{30} bits)
Gbyte	gigabyte (10^9 or 2^{30} bytes)
Gword	gigaword (10^9 words)
Hz	hertz (cycles per second)
I/O	input/output
IGC	Image Generation Controller
Kbit	kilobit (10^3 or 2^{10} bits)
Kbyte	kilobyte (10^3 or 2^{10} bytes)
LSB	least-significant bit (or byte)
MFLOPS	million floating-point operations per second
MHz	megahertz (10^6 Hz)
MIP	<i>multum in parvo</i> (multi-resolution table for filtering textures)
MIPS	million instructions per second
MSB	most-significant bit (or byte)
Mbit	megabit (10^6 or 2^{20} bits)
Mbyte	megabyte (10^6 or 2^{20} bytes)
Mword	megaword (10^6 words)
NURB	Non-Uniform Rational B-Spline
PC	printed-circuit (board)
PLD	Programmable Logic Device
RGB	Red/Green/Blue
RGBα	Red/Green/Blue/Alpha
RISC	Reduced Instruction Set Computer
SIMD	Single-Instruction Multiple-Data
TLB	Translation Lookaside Buffer
U	Height-unit (1.75 inches—for PC boards)
VRAM	Video Random-Access Memory
mil	10^{-3} inches
msec	millisecond (10^{-3} seconds)
<i>n</i>-T	<i>n</i> -transistor (in a memory cell)
nsec	nanosecond (10^{-9} seconds)
tri	triangle
μsec	microsecond (10^{-6} seconds)

