David Zhao Akeley – Résumé

Majors	CS, Mathematics	**Select Engineering Courses**	**Select Math Courses**
GPA	3.762	Parallel & Distributed Computing	Linear Algebra
Graduated	August 2020	Advanced Computer Architecture	Complex Analysis Honors
Primary Email	dza724[at]gmail.com	Machine Learning	Algebra Honors
SMS	1-408-763-5241	Algorithms & Complexity	Galois Theory
Academic Email	akeley98[at]ucla.edu	Formal Langauges & Automata	Mathematical Modeling

○ Work Experience

× Nvidia – October 2020 - Present – Developer Technology Engineer

1. TODO: actually do this job.

× Sholari LLC – July - September 2019 – Contractor

1. Worked on a tumor growth and treatment simulator written with the Unity game engine.
2. Implemented tools for visualizing tumor responses to treatments: line graphs, waterfall (bar) plots, and the user interface for the timeline (graph x-axis).
3. Wrote a multithreaded `C++11` plugin for visualizing tumors & immune system responses as particle clouds, and integrated it with the single-threaded `C#` Unity Engine.

× Stanford University – June - September 2018 – Research Assistant

1. Undergraduate co-author of Aetherling with David Durst (lead author), Dr. Kayvon Fatahalian, and Dr. Pat Hanrahan.
2. Aetherling supports automatic parallelization and static scheduling[1] of hardware image processing pipelines designed using a Haskell-embedded domain-specific language.
3. Contributed to the functional simulator of an early prototype of Aetherling and revised the Aetherling type system to remove impediments to parallelizing the prototype's line buffers.[2]

`https://aetherling.org` ("Type-Directed Scheduling of Streaming Accelerators" - PLDI 2020)

× MediocrePy – March - June 2017 – Independent Project

1. Created an optimized library for reducing stacks of telescope images to a single image using pixel means or medians and optional outlier rejection (sigma clipping) for noise reduction.
2. Multithreaded C core with AVX vectorization; C and Python (numpy) interface. Decreased runtime (compared to the Python implementation replaced) from hours to milliseconds.
3. Collaborated with Dr. Zheng Cai, UC Santa Cruz Astrophysics.

`https://github.com/akeley98/MediocrePy`

× Tsinghua University – July - August 2016 – Summer Intern

1. Designed a small library for fitting and plotting standard microlensing event light curves given discrete measurements of a star's apparent magnitude (brightness) through time.
2. Used Python, `C++`, SciPy, Matplotlib.
3. Collaborated with Dr. Shude Mao, Tsinghua Department of Astronomy.

× Jide Technology Co. – June - July 2015 – Summer Intern

1. Performed product testing for RemixOS, an Android derivative with a desktop-like interface.
2. Edited international marketing material and wrote user documentation in English.
3. Collaborated with Jason Zheng and Jeff Zhao (International Marketing Manager).

[1] i.e. not using ready-valid hardware interfaces.

[2] A line buffer device reads in an image as a stream of pixel values and outputs rectangular portions ("windows") of the image.

○ Other Projects

× WebGL Jelly Cube Project

Simple mass-spring system simulation written with Javascript, WebAssembly, and WebGL 2.0 (for refractive and reflective effects). Earned third place in the UCLA computer graphics class contest, Fall 2017.[3]

https://github.com/akeley98/JellyMcJelloFace https://youtu.be/YwvMSeB6NzU

× Myricube – Vulkan Voxel Renderer

This proof-of-concept hybrid raycast/mesh voxel renderering was inspired by my discontent with Minecraft's low render distance and slow chunk updates, even on high settings.

Conventional voxel renderers such as Minecraft's work by converting voxel models into a mesh of triangles to draw, possibly with some optimizations like hidden face removal or merging coplanar triangles. This uses the GPU for its designed purpose, but as render distance increases, the GPU is bottlenecked by the large number of small triangles generated. Myricube uses conventional mesh rendering[4] for nearby voxels but draws more distant voxels with an alternative raycasting renderer. This renderer subdivides the voxel model into cubic chunks, computes a minimal AABB[5] for each chunk's visible voxels, and draws the AABBs themselves with a raycasting fragment shader that only checks for intersections within the drawn AABB. Dividing the model into AABBs reduces my renderer's memory bandwidth requirements and time wasted raycasting through empty space, compared to a conventional raycasting renderer.

Additionally, the project leverages memory-mapped files and Vulkan asynchronous transfers to allow for low-latency, high-throughput model upload and real-time animation.

https://github.com/akeley98/myricube

× Proposed gem5 Hardware Simulator Partial Bypassing Patch

This patch is refactored `C++11` code that I wrote for my Advanced Computer Architecture course project.

In an out-of-order CPU, hardware bypassing allows instructions stalled waiting for a register to start execution as soon as the functional unit (ALU, memory port, etc.) writing to said register completes, without waiting for the actual commitment to register file. This is crucial for performance, so mainline `gem5` simulates complete bypassing between all functional units in the simulated CPU. However, this is not practical to realize in modern superscalar hardware, as bypassing costs grow quadratically with the number of functional units. My patch splits the CPU's functional units into separate pools, with bypassing simulated only between pools. The commit-to-register-file delay is imposed for values communicated between pools.

The patch received positive code reviews; however, it doesn't look like there's enough interest in the feature to actually merge it into mainline `gem5`.

https://gem5-review.googlesource.com/c/public/gem5/+/27767

https://github.com/akeley98/FU-pools/blob/b4d291429edb6aa4988888656b6867ff99591b90/fu-pools.pdf

[3] https://www.facebook.com/vasilescu.alex/posts/10155206917936588

[4] More precisely, instanced rendering to convert subsets of a list of voxels into a mesh to draw.

[5] Axis aligned bounding box – requires only 12 triangles to draw.

◦ Appendix: UCLA Education – September 2017 - August 2020

First Major Computer Science **Second Major** Mathematics **GPA** 3.762 (August 2020)

		Title	Content Notes
A+	CS 174A	Intro to Computer Graphics	See WebGL Jelly Cube Project
A+	EE M16	Digital Systems	Verilog Lab
A+	Math 115A	Linear Algebra	
A	Math 170A	Probability Theory	
A	CS 35L	Software Construction Lab	POSIX basics (e.g. pthreads, bash)
A	CS M146	Machine Learning	
A	Math 110A	Algebra	Ring Theory
A	Math 131AH	Analysis Honors	Metric Spaces
A+	CS 180	Algorithms & Complexity	
A	Engr 185EW	Art of Engineering Endeavors	Writing Intensive Team Project
A	Math 131BH	Analysis Honors	Derivation, Riemann Integration
B-	CS 131	Programming Languages	
B+	CS M152A	Digital Design Lab	Verilog Team Project
B+	Math 110AH	Algebra Honors	Group Theory
C	Math 120A	Differential Geometry	
A-	CS 111	Operating Systems Principles	Focus on POSIX
A+	CS 181	Formal Languages & Automata	Regex, CFG, Turing Machines, Decidability
A-	Math 132H	Complex Analysis Honors	
A-	CS 161	Fundamentals of Artificial Intelligence	
A+	EE M116C	Computer Systems Architecture	
B+	Math 111	Theory of Numbers	Overview of p-adic Numbers
A-	CS 118	Computer Network Fundamentals	
B	CS 130	Software Engineering	Java Team Project
A	Math 134	Systems of Differential Equations	
A+	CS 133	Parallel & Distributed Computing	OpenMP, OpenCL, MPI, GPGPU, FPGA
A+	CS 251A	Advanced Computer Architecture	`gem5` Hardware Sim Project, Graduate Course
A	Math 110BH	Algebra Honors	Ring Theory, Module Theory
A	Math 110C	Algebra*	Field Theory, Galois Theory
A	Math 115B	Linear Algebra	
A	Math 142	Mathematical Modeling	
A+	Math 167	Game Theory	

** There is no honors equivalent to the Galois Theory Course.*

◦ West Valley College Education – 2015-2017

GPA 4.0

Select Courses

	Title	Content Notes
Math 4B	Differential Equations	
Math 19	Discrete Mathematics	
Psych 2	Experimental Psychophysiology	Experiment Design & Paper
Phys 4D	Modern Physics	Relativity

Typeset in LaTeX

Fonts: Computer Modern, FreeSans (GNU FreeFont), `Ubuntu Mono` (Dalton Maag & Canonical Ltd.)