Taichi: An Introduction

Languages Seminar

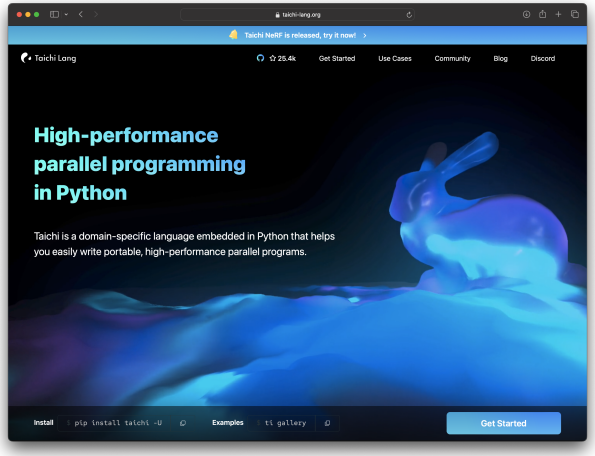
Russell Bentley

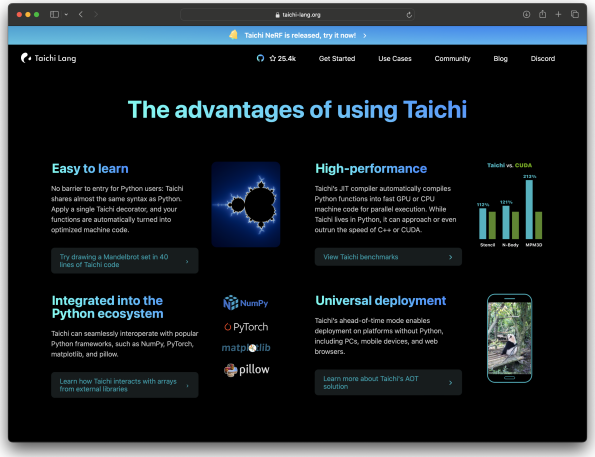
Stony Brook

Thursday 3rd October, 2024

Useful Links

- Taichi homepage:
  https://www.taichi-lang.org
- Official docs / guides:
  https://docs.taichi-lang.org
- API reference:
  https://docs.taichi-lang.org/api/
- Github:
  https://github.com/taichi-dev/taichi

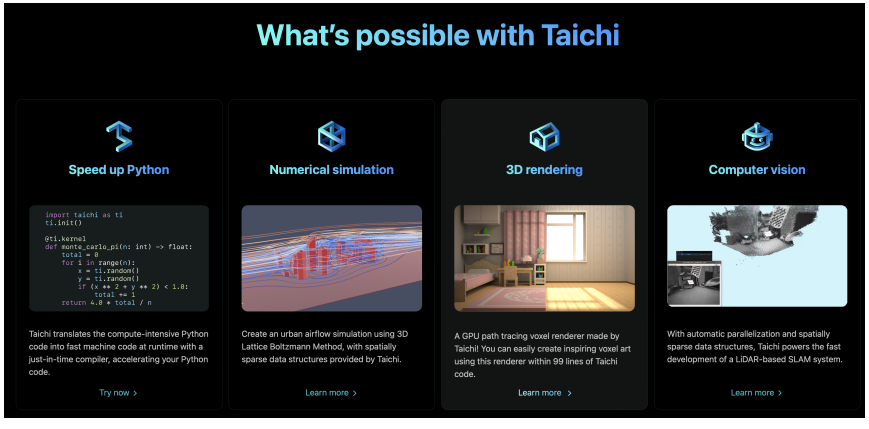


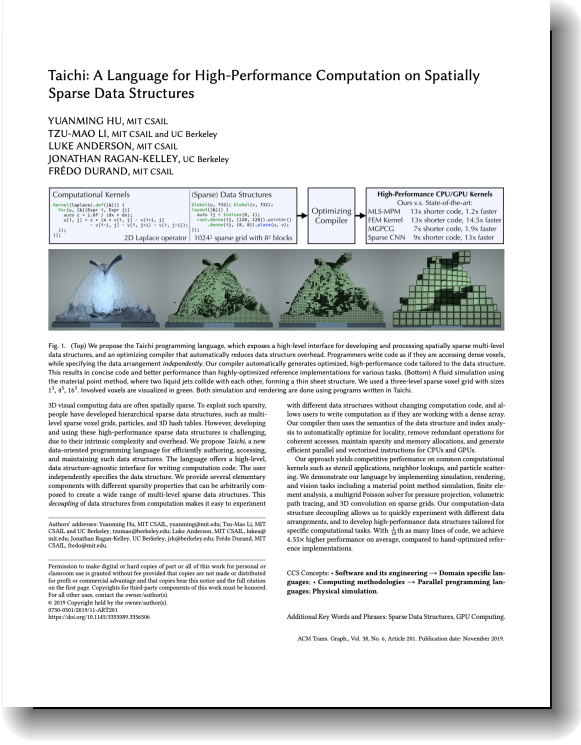


Stony Brook University

Taichi?

- Large and Growing Project
- Language for compute kernels
- Data structure DSL
- Python frontend
- Compiler Infrastructure
- Focus on Graphics



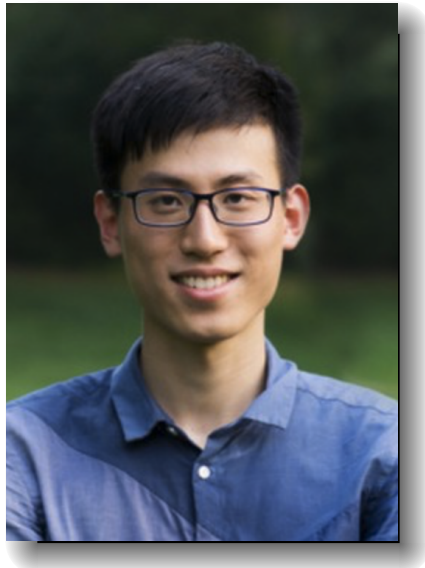

Taichi: A Language for High-Performance Computation on Spatially Sparse Data Structures

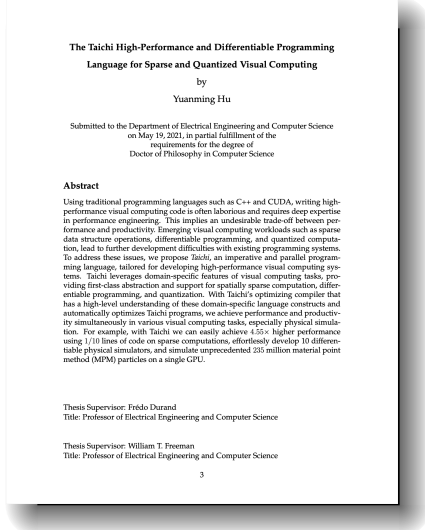
YUANMING HU, MIT CSAIL
TZU-MAO LI, MIT CSAIL and UC Berkeley
LUKE ANDERSON, MIT CSAIL
JONATHAN RAGAN-KELLEY, UC Berkeley
FRÉDO DURAND, MIT CSAIL

2019 [4]

Taichi Background

- Taichi is Yuanming Hu's 2021 Thesis
- Growing collection of related papers





The Taichi High-Performance and Differentiable Programming Language for Sparse and Quantized Visual Computing

by

Yuanming Hu

Submitted to the Department of Electrical Engineering and Computer Science
on May 19, 2021, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science

Abstract

Using traditional programming languages such as C++ and CUDA, writing high-performance visual computing code is often laborious and requires deep expertise in performance engineering. This implies an undesirable trade-off between performance and productivity. Emerging visual computing workloads such as sparse data structure operations, differentiable programming, and quantized computation, lead to further development difficulties with existing programming systems. To address these issues, we propose *Taichi*, an imperative and parallel programming language, tailored for developing high-performance visual computing systems. Taichi leverages domain-specific features of visual computing tasks, providing first-class abstraction and support for spatially sparse computation, differentiable programming, and quantization. With Taichi's optimizing compiler that has a high-level understanding of these domain-specific language constructs and automatically optimizes Taichi programs, we achieve performance and productivity simultaneously in various visual computing tasks, especially physical simulation. For example, with Taichi we can easily achieve 4.55× higher performance using 1/10 lines of code on sparse computations, effortlessly develop 10 differentiable physical simulators, and simulate unprecedented 235 million material point method (MPM) particles on a single GPU.

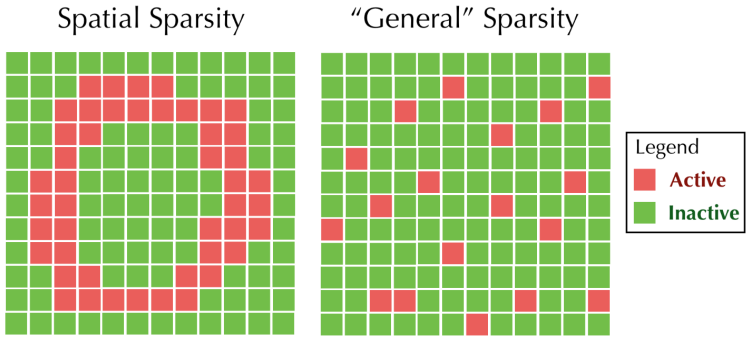
Thesis Supervisor: Frédo Durand
Title: Professor of Electrical Engineering and Computer Science

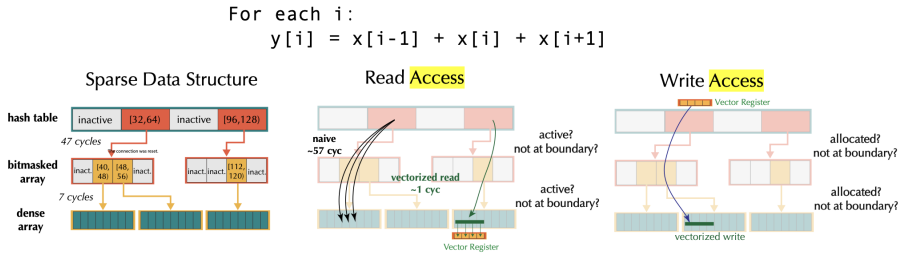
Thesis Supervisor: William T. Freeman
Title: Professor of Electrical Engineering and Computer Science

3

Design Goals

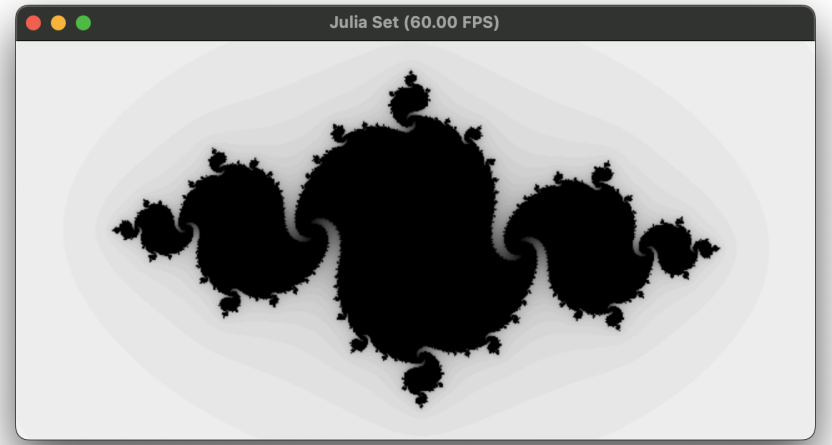
- Explore data representation
  - Sparse data structures
  - Quantized data types
- Decouple data and algorithms
- Differentiable kernels
- Optimize access patterns
- Optimize parallelism



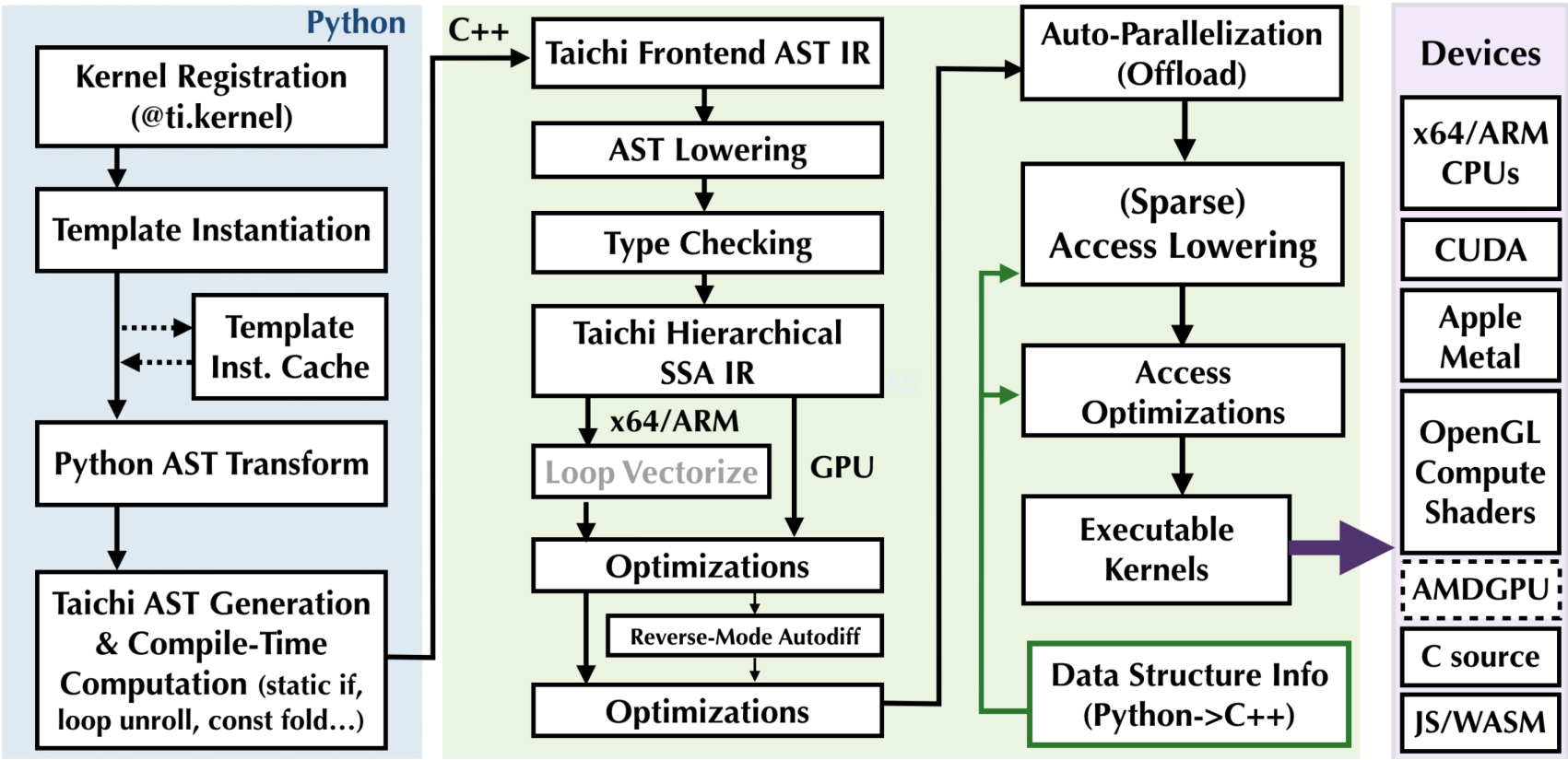



```
i5 = ti.quant.int(bits=5)
u19 = ti.quant.int(bits=19, signed=False)
```

Interactive Example



Life of a Taichi Kernel



From Life of a Taichi Kernel slides

Why use a Python Frontend?

Taichi's old C++ frontend

Taichi used to be embedded in C++14. However, that solution is mostly abandoned, because

1. C++ itself is too complex for most users to learn, not to say a DSL embedded in C++. E.g., ▸`math/svd.h` ☹
2. Getting C++ AST is almost impossible. We had to heavily use templates/macros tricks, which harm readability. ☹

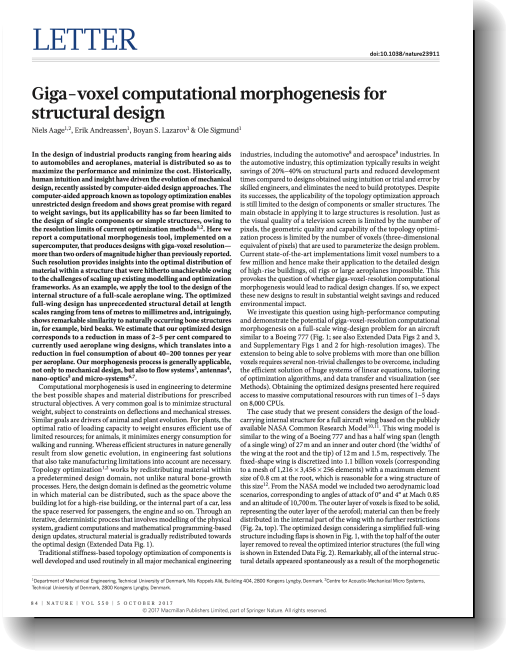
Taichi's new Python frontend

Now the whole Taichi system is deeply embedded in Python.

1. Python is easy to learn and widely adopted. ☺
2. Python allows flexible AST inspection and manipulation. ☺

Both of these allow us to invent a new high-performance language out of Python.

Applications: Topology Optimization



LETTER

doi:10.1038/nature23911

Giga-voxel computational morphogenesis for structural design

Niels Aage[1,2], Erik Andreassen[1], Boyan S. Lazarov[1] & Ole Sigmund[1]

In the design of industrial products ranging from hearing aids to automobiles and aeroplanes, material is distributed so as to maximize the performance and minimize the cost. Historically, human intuition and insight have driven the evolution of mechanical design, recently assisted by computer-aided design approaches. The computer-aided approach known as topology optimization enables unrestricted design freedom and shows great promise with regard to weight savings, but its applicability has so far been limited to the design of single components or simple structures, owing to the resolution limits of current optimization methods[1,2]. Here we report a computational morphogenesis tool, implemented on a supercomputer, that produces designs with giga-voxel resolution—more than two orders of magnitude higher than previously reported. Such resolution provides insights into the optimal distribution of material within a structure that were hitherto unachievable owing to the challenges of scaling up existing modelling and optimization frameworks. As an example, we apply the tool to the design of the internal structure of a full-scale aeroplane wing. The optimized full-wing design has unprecedented structural detail at length scales ranging from tens of metres to millimetres and, intriguingly, shows remarkable similarity to naturally occurring bone structures in, for example, bird beaks. We estimate that our optimized design corresponds to a reduction in mass of 2–5 per cent compared to currently used aeroplane wing designs, which translates into a reduction in fuel consumption of about 40–200 tonnes per year per aeroplane. Our morphogenesis process is generally applicable, not only to mechanical design, but also to flow systems[3], antennas[4], nano-optics[5] and micro-systems[6,7].

Computational morphogenesis is used in engineering to determine the best possible shapes and material distributions for prescribed structural objectives. A very common goal is to minimize structural weight, subject to constraints on deflections and mechanical stresses. Similar goals are drivers of animal and plant evolution. For plants, the optimal ratio of loading capacity to weight ensures efficient use of limited resources; for animals, it minimizes energy consumption for walking and running. Whereas efficient structures in nature generally result from slow genetic evolution, in engineering fast solutions that also take manufacturing limitations into account are necessary. Topology optimization[1,2] works by redistributing material within a predetermined design domain, not unlike natural bone-growth processes. Here, the design domain is defined as the geometric volume in which material can be distributed, such as the space above the building lot for a high-rise building, or the internal part of a car, less the space reserved for passengers, the engine and so on. Through an iterative, deterministic process that involves modelling of the physical system, gradient computations and mathematical programming-based design updates, structural material is gradually redistributed towards the optimal design (Extended Data Fig. 1).

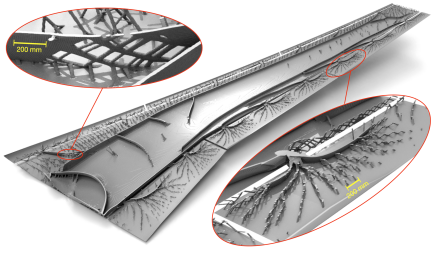
Traditional stiffness-based topology optimization of components is well developed and used routinely in all major mechanical engineering industries, including the automotive[8] and aerospace[9] industries. In the automotive industry, this optimization typically results in weight savings of 20%–40% on structural parts and reduced development times compared to designs obtained using intuition or trial and error by skilled engineers, and eliminates the need to build prototypes. Despite its successes, the applicability of the topology optimization approach is still limited to the design of components or smaller structures. The main obstacle in applying it to large structures is resolution. Just as the visual quality of a television screen is limited by the number of pixels, the geometric quality and capability of the topology optimization process is limited by the number of voxels (three-dimensional equivalent of pixels) that are used to parameterize the design problem. Current state-of-the-art implementations limit voxel numbers to a few million and hence make their application to the detailed design of high-rise buildings, oil rigs or large aeroplanes impossible. This provokes the question of whether giga-voxel-resolution computational morphogenesis would lead to radical design changes. If so, we expect these new designs to result in substantial weight savings and reduced environmental impact.

We investigate this question using high-performance computing and demonstrate the potential of giga-voxel-resolution computational morphogenesis on a full-scale wing-design problem for an aircraft similar to a Boeing 777 (Fig. 1; see also Extended Data Figs 2 and 3, and Supplementary Figs 1 and 2 for high-resolution images). The extension to being able to solve problems with more than one billion voxels requires several non-trivial challenges to be overcome, including the efficient solution of huge systems of linear equations, tailoring of optimization algorithms, and data transfer and visualization (see Methods). Obtaining the optimized designs presented here required access to massive computational resources with run times of 1–5 days on 8,000 CPUs.

The case study that we present considers the design of the load-carrying internal structure for a full aircraft wing based on the publicly available NASA Common Research Model[10,11]. This wing model is similar to the wing of a Boeing 777 and has a half wing span (length of a single wing) of 27 m and an inner and outer chord (the 'widths' of the wing at the root and the tip) of 12 m and 1.5 m, respectively. The fixed-shape wing is discretized into 1.1 billion voxels (corresponding to a mesh of 1,216 × 3,456 × 256 elements) with a maximum element size of 0.8 cm at the root, which is reasonable for a wing structure of this size[12]. From the NASA model we included two aerodynamic load scenarios, corresponding to angles of attack of 0° and 4° at Mach 0.85 and an altitude of 10,700 m. The outer layer of voxels is fixed to be solid, representing the outer layer of the aerofoil; material can then be freely distributed in the internal part of the wing with no further restrictions (Fig. 2a, top). The optimized design considering a simplified full-wing structure including flaps is shown in Fig. 1, with the top half of the outer layer removed to reveal the optimized interior structures (the full wing is shown in Extended Data Fig. 2). Remarkably, all of the internal structural details appeared spontaneously as a result of the morphogenetic

[1]Department of Mechanical Engineering, Technical University of Denmark, Nils Koppels Allé, Building 404, 2800 Kongens Lyngby, Denmark. [2]Centre for Acoustic-Mechanical Micro Systems, Technical University of Denmark, 2800 Kongens Lyngby, Denmark.

84 | NATURE | VOL 550 | 5 OCTOBER 2017



2017 [1]

Giga-voxel Problems

- $\approx$ 1 Billion Active Voxels
- 1-5 day runtime

$\leftarrow$ Prior SotA

- 8000 cores over many nodes

Taichi Based $\rightarrow$

- Single 14 core workstation

Narrow-Band Topology Optimization on a Sparsely Populated Grid

HAIXIANG LIU*, University of Wisconsin-Madison
YUANMING HU*, MIT CSAIL
BO ZHU, MIT CSAIL and Dartmouth College
WOJCIECH MATUSIK, MIT CSAIL
EFTYCHIOS SIFAKIS, University of Wisconsin-Madison

A variety of structures in nature exhibit sparse, thin, and intricate features. It is challenging to investigate these structural characteristics using conventional numerical approaches since such features require highly refined spatial resolution to capture and therefore they incur a prohibitively high computational cost. We present a novel computational framework for high-resolution topology optimization that delivers leaps in simulation capabilities, by two orders of magnitude, from the state-of-the-art approaches. Our technique accommodates computational domains with over one billion grid voxels on a single shared-memory multiprocessor platform, allowing automated emergence of structures with both rich geometric features and exceptional mechanical performance. To achieve this, we track the evolution of thin structures and simulate its elastic deformation in a dynamic narrow-band region around high-density sites to avoid wasted computational effort on large void regions. We have also designed a mixed-precision multigrid preconditioned iterative solver that keeps the memory footprint of the simulation to a compact size while maintaining double-precision accuracy. We have demonstrated the efficacy of the algorithm through optimizing a variety of complex structures from both natural and engineering systems.

CCS Concepts • **Computing methodologies** → **Massively parallel and high-performance simulations**;

Additional Key Words and Phrases: topology optimization, sparsely populated grid, multigrid solver

ACM Reference Format:
Haixiang Liu, Yuanming Hu, Bo Zhu, Wojciech Matusik, and Eftychios Sifakis. 2018. Narrow-Band Topology Optimization on a Sparsely Populated Grid. *ACM Trans. Graph.* 37, 6, Article 251 (November 2018), 14 pages. https://doi.org/10.1145/3272127.3275012

1 INTRODUCTION

Mechanical structures in nature exhibit a broad spectrum of thin features. Such examples include interior structures supporting insect wings, animal bones, bird beaks, honeycombs, *etc.* These codimensional structures have negligible volume compared to the size of geometries they support. Under the influence of natural selection,

*Indicates equal contribution.

Authors' addresses: Haixiang Liu, University of Wisconsin-Madison, hliu253@wisc.edu; Yuanming Hu, MIT CSAIL, yuanming@mit.edu; Bo Zhu, MIT CSAIL and Dartmouth College, bo.zhu@dartmouth.edu; Wojciech Matusik, MIT CSAIL, wojciech@csail.mit.edu; Eftychios Sifakis, University of Wisconsin-Madison, sifakis@cs.wisc.edu.

0730-0301/2018/11-ART251 $15.00
https://doi.org/10.1145/3272127.3275012

ACM Trans. Graph., Vol. 37, No. 6, Article 251. Publication date: November 2018.

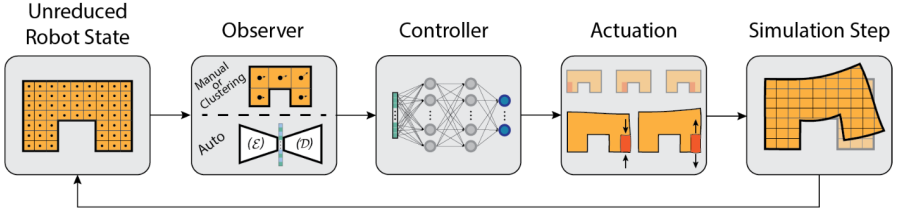
Side View
Top View
Back View

Fig. 1. *The interior structures supporting the shell of a bird beak generated using our narrow-band topology optimization algorithm with* 1, 940, 875, 347 *(1.04 billion) active FEM voxels. The resolution of the background grid is* 3000 × 2000 × 1600*. The three figures show the volumetric rendering of the same structure from different views.*

2018 [5]

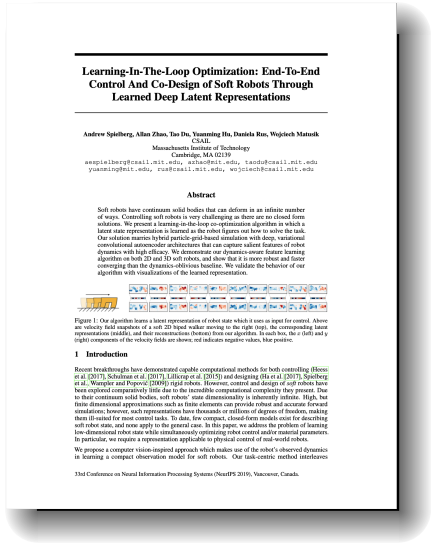
Applications: Differentiable Physics

- Soft Robotics
  - Difficult to design
  - Difficult to control
- Simulation
  - Sparse representation
  - Differentiable for learning / optimizing



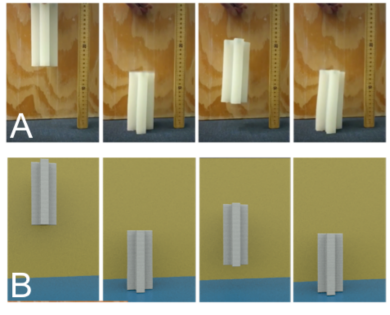
ChainQueen: A Real-Time Differentiable Physical Simulator for Soft Robotics

Yuanming Hu, Jiancheng Liu*, Andrew Spielberg*, Joshua B. Tenenbaum, William T. Freeman, Jiajun Wu, Daniela Rus, Wojciech Matusik[1,2]



Learning-In-The-Loop Optimization: End-To-End Control And Co-Design of Soft Robots Through Learned Deep Latent Representations

Andrew Spielberg, Allan Zhao, Tao Du, Yuanming Hu, Daniela Rus, Wojciech Matusik

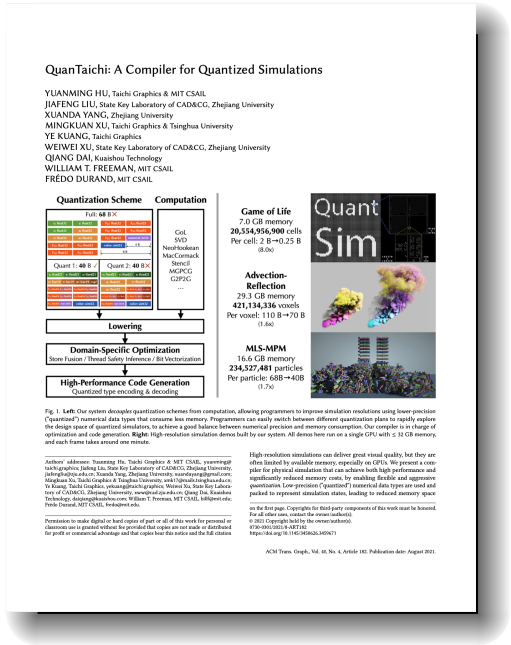


2019 [2]

2019 [8]

Applications: Quantized Graphics

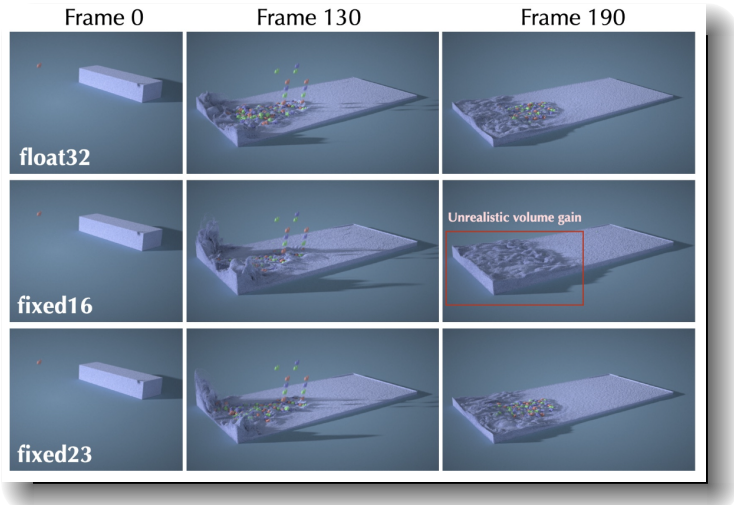
- Decouple algorithms from data representation
- Lower memory requirements
  - 8x Game of Life
  - 1.5x Eulerian Fluid
  - 1.7x Material point method
- Leads to faster runtimes



QuanTaichi: A Compiler for Quantized Simulations

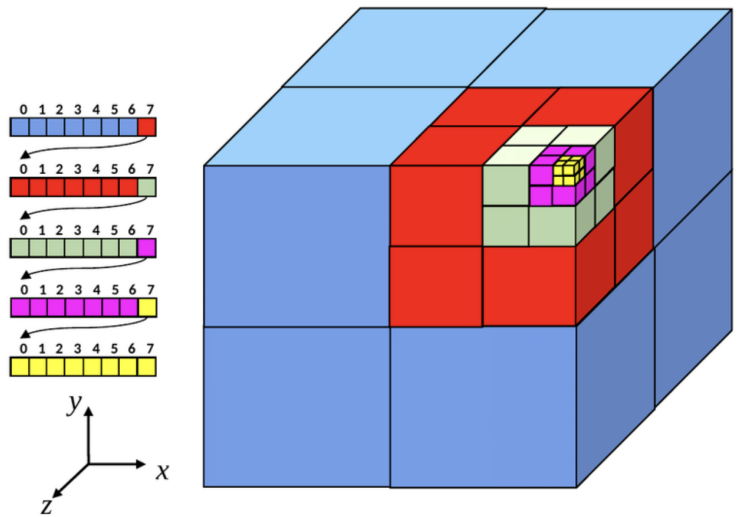
YUANMING HU, Taichi Graphics & MIT CSAIL
JIAFENG LIU, State Key Laboratory of CAD&CG, Zhejiang University
XUANDA YANG, Zhejiang University
MINGKUAN XU, Taichi Graphics & Tsinghua University
YE KUANG, Taichi Graphics
WEIWEI XU, State Key Laboratory of CAD&CG, Zhejiang University
QIANG DAI, Kuaishou Technology
WILLIAM T. FREEMAN, MIT CSAIL
FRÉDO DURAND, MIT CSAIL

2021 [3]



Sparse Data Structures: Octree

- Easy to implement
- Hard to write access patterns
- Not cache friendly
- Can only represent data in a bounded space



From ResearchGate

OpenVDB

- Originally from Dreamworks animation
- Published in ACM Trans. Graphics
  - Describes VDB Datastructure
  - Introduced OpenVDB implementation
- Widely used for special effects and animation
  - Supported by industry standard software like Houdini and Blender
- Often used with implicit geometry

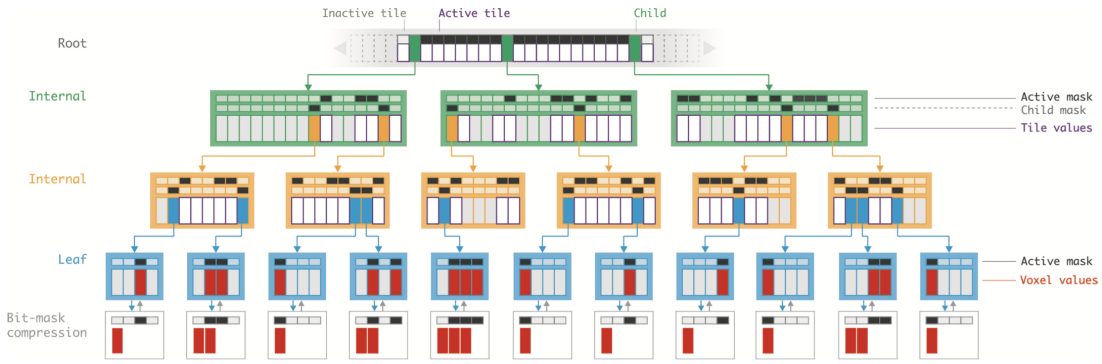
VDB: High-Resolution Sparse Volumes with Dynamic Topology

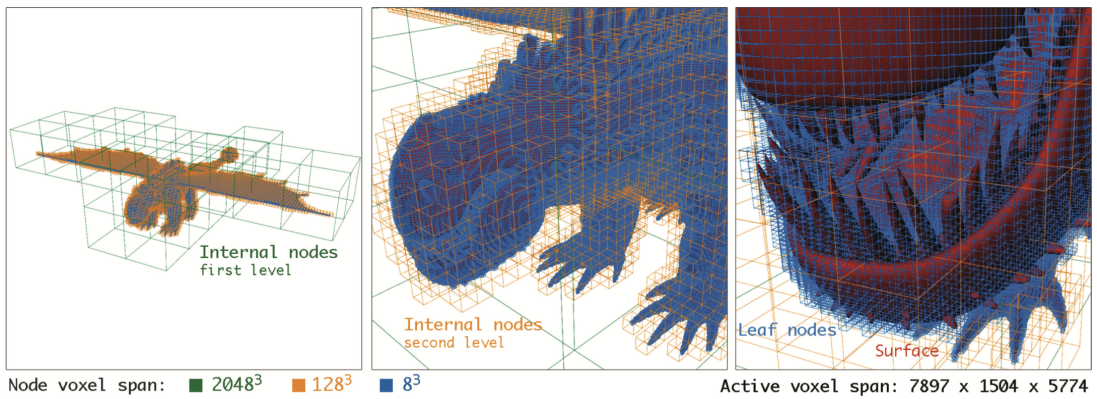
KEN MUSETH
DreamWorks Animation

2013 [6]

VDB Datastructure

- Similar to B+Trees
- Map based root
  - Dynamic branching factor
  - Virtually infinite spatial bounds
- Fixed height
- Large (fixed) branching factors for children





Images from [6]

SPGrid

- ACM Trans. Graphics 2014 [7]
- Key ideas
  - Large finite grid replaces infinite map
  - Morton key as virtual memory address
  - Translation Lookahead Buffer (TLB) handles address mapping

SPGrid: A Sparse Paged Grid structure applied to adaptive smoke simulation

Rajsekhar Setaluri Mridul Aanjaneya Sean Bauer Eftychios Sifakis

University of Wisconsin - Madison

Figure 1: *Smoke flow past sphere with 135M active voxels, 1K×1K×2K maximum resolution. Adaptive grid shown on the right.*

Abstract

We introduce a new method for fluid simulation on high-resolution adaptive grids which rivals the throughput and parallelism potential of methods based on uniform grids. Our enabling contribution is *SPGrid*, a new data structure for compact storage and efficient stream processing of sparsely populated uniform Cartesian grids. SPGrid leverages the extensive hardware acceleration mechanisms inherent in the x86 Virtual Memory Management system to deliver sequential and stencil access bandwidth comparable to dense uniform grids. Second, we eschew tree-based adaptive data structures in favor of storing simulation variables in a pyramid of sparsely populated uniform grids, thus avoiding the cost of indirect memory access associated with pointer-based representations. We show how the costliest algorithmic kernels of fluid simulation can be implemented as a composition of two kernel types: (a) stencil operations on a single sparse uniform grid, and (b) structured data transfers between *adjacent* levels of resolution, even when modeling non-graded octrees. Finally, we demonstrate an adaptive multigrid-preconditioned Conjugate Gradient solver that achieves resolution-independent convergence rates while admitting a lightweight implementation with a modest memory footprint. Our method is complemented by a new interpolation scheme that reduces dissipative effects and simplifies dynamic grid adaptation. We demonstrate the efficacy of our method in end-to-end simulations of smoke flow.

CR Categories: I.3.7 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling;

Keywords: fluid simulation, sparse grids, adaptive discretizations

Links: DL PDF WEB

1 Introduction

Computer graphics research has explored opportunities for spatial adaptation in fluid simulation using a broad spectrum of techniques, including adaptive Cartesian grids [Losasso et al. 2004; Zhu et al. 2013; Fersil et al. 2014], adaptive tetrahedral meshes [Klingner et al. 2006; Ando et al. 2013] and particle based approaches [Adams et al. 2007; Solenthaler and Gross 2011]. For the same number of degrees of freedom, the visual richness of adaptive techniques is clearly superior to uniform grids. However, the computational cost is markedly higher. Common operations such as stencil computations and sparse linear algebra are significantly more efficient on uniform grids, as they can leverage cache-optimized memory access patterns, maximize prefetching efficiency and simplify balanced domain partitioning for parallelism.

This has prompted many authors to pursue balanced compromises between highly adaptive representations and purely uniform grids, to achieve the best of both worlds in terms of both detail and performance. RLE-based techniques [Houston et al. 2006; Irving et al. 2006; Chentanez and Müller 2011] propose a hybrid between a 2D uniform grid and a 1D run-length encoding scheme. Adaptive mesh refinement (AMR) and chimera grid techniques [Sussman et al. 1999; Patel et al. 2005; Dobashi et al. 2008; Tan et al. 2008; Cohen et al. 2010; Golas et al. 2012; English et al. 2013] achieve spatial adaptation by patching together overlapping uniform grids of different resolutions. It has also been proposed that dimension-by-dimension adaptation can focus resolution on areas of interest, while maintaining a topologically uniform underlying grid [Zhu et al. 2013]. Finally, one of the most popular and performant recent

ACM Transactions on Graphics, Vol. 33, No. 6, Article 205, Publication Date: November 2014

Stony Brook University

Composable Data Structures

```
root.hash(ijk, 32).dense(ijk, 16).pointer()
    .dense(ijk, 8).place(u, v, w);
```

(a) 3D VDB-style [Museth 2013] structure with configuration [5, 4, 3]. The root-level hash table allows negative coordinates to be accessed, providing the user with an unbounded domain.

```
root.dense(ijk, 512).morton().bitmasked()
    .dense(ijk, {8, 4, 4}).place(flags, u, v, w);
```

(b) 3D SPGrids [Setaluri et al. 2014] occupying voxels in the bounding box $[0, 4096) \times [0, 2048) \times [0, 2048)$. The data structure is relatively shallow (only two levels), so root-to-leaf accesses have relatively low cost.

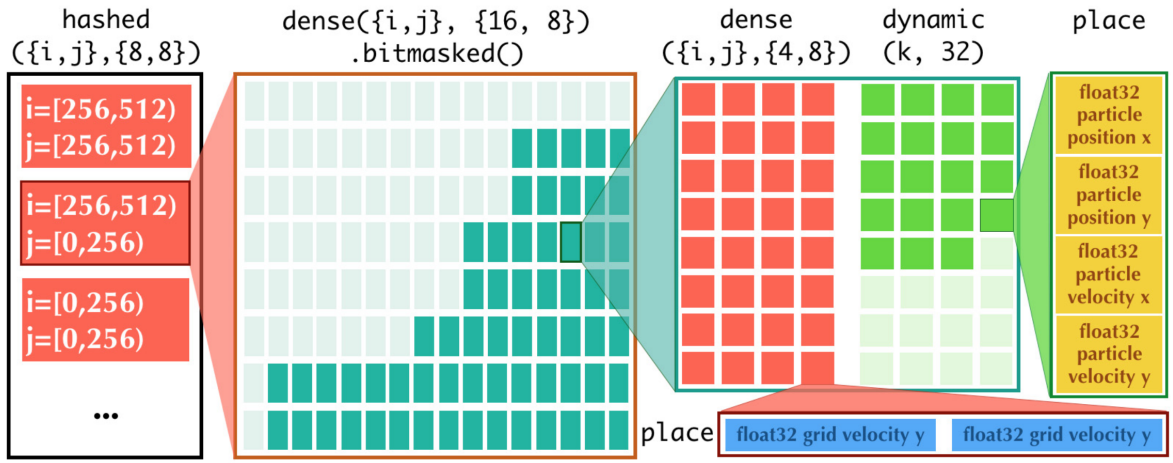
```
// "Hierarchical Particle Buckets": each leaf block contains all indices of particles within its range
root.dynamic(l, 2048).place(particle_x, particle_y, particle_z, particle_mass);
root.hash(ijk, 512).dense(ijk, 32).pointer().dense(ijk, 8).pointer().dynamic(l, 2048).place(particle_index);

// "SPVDB": Unbounded shallow data structures with bitmasks and Morton coding. (VDB and SPGrid combined.)
root.hash(ijk, 512).dense(ijk, 512).morton().bitmasked().dense(ijk, {8, 4, 4}).place(flags, u, v, w);
```

```
// "Hybrid Eulerian-
     Lagrangian Grid"
auto &block = root
 .hash(ij, 8)
 .dense(ij, {16, 8})
 .bitmasked();

// Child 1: grid nodes
block.dense(ij, {4, 8})
     .place(grid_vx)
     .place(grid_vy);

// Child 2: particles
block.dynamic(i, 32)
     .place(part_x)
     .place(part_y)
     .place(part_vx)
     .place(part_vy);
```

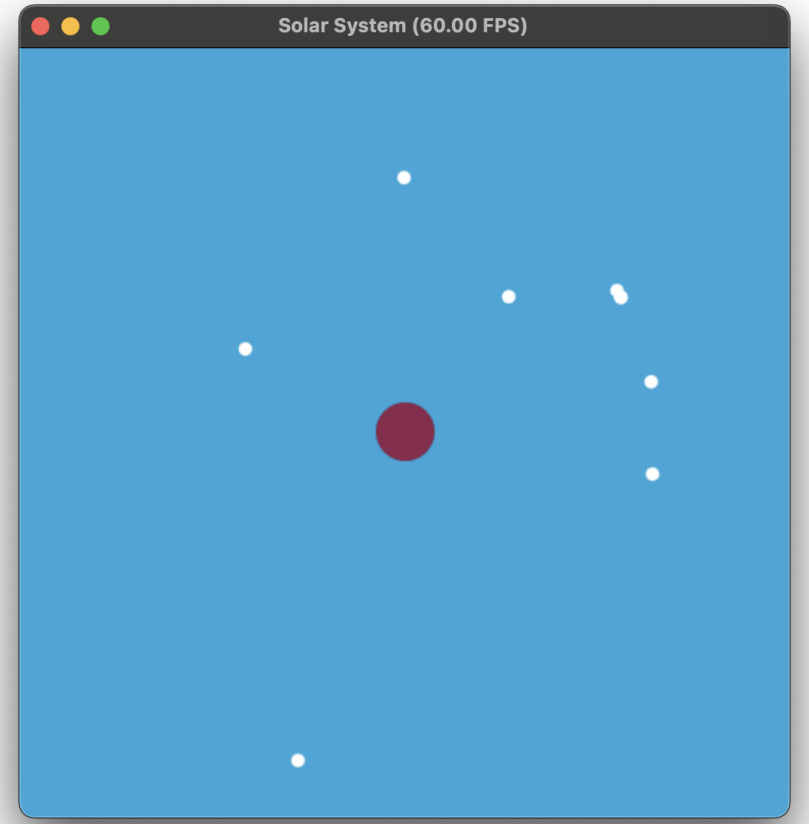


(c) **"HPB", "SPVDB", "HLEG":** We can easily design new data structures with customized features.

Fig. 6. The layout language allows users to define data structures using our building blocks. We can reproduce two popular multi-level sparse grid used in simulation (a) (b). Furthermore, we can use our language to design new data structures (c) by chaining and forking elementary components. Hybrid Eulerian-Lagrangian simulations (e.g. FLIP [Zhu and Bridson 2005] and MPM [Stomakhin et al. 2013]) often need to maintain both particles and grids, and the required data structures are usually complicated. Using these building blocks, we easily found a data structure with a hierarchical pointer list of particles, which we call *Hierarchical Particle Buckets*, that is especially useful for the material point method simulation (Sec.6.1).

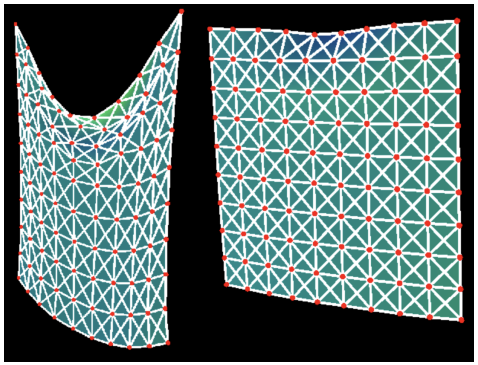
From [4]

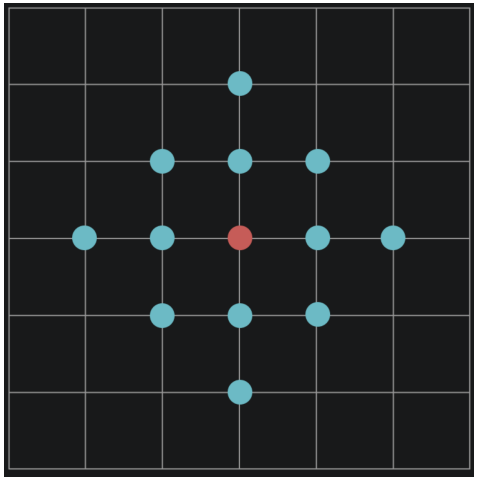
Objective Data-oriented Programming Example

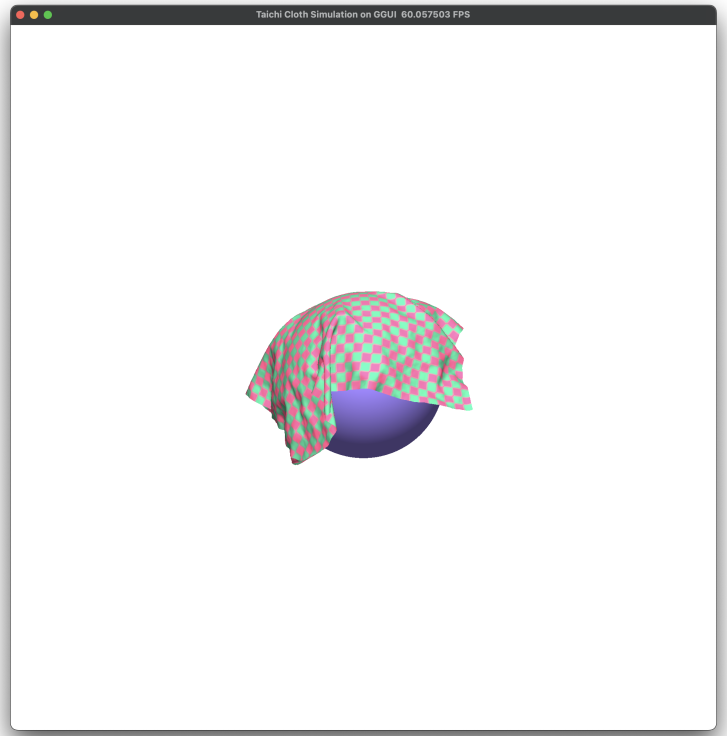


Adapted from Taichi SIGGRAPH Workshop slides

Physical Simulation Example







Stony Brook University

Adapted from Conduct Physical Simulation guide

Conclusion

- Today we've introduced Taichi
  - Python based implementation
  - Design Goals
  - Applications and Research Projects
  - Sparse Data Structures
- Questions?

References I

[1] Niels Aage et al. "Giga-voxel computational morphogenesis for structural design". In: *Nature* 550.7674 (2017), pp. 84–86.

[2] Yuanming Hu et al. "ChainQueen: A Real-Time Differentiable Physical Simulator for Soft Robotics". In: *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)* (2019).

[3] Yuanming Hu et al. "QuanTaichi: A Compiler for Quantized Simulations". In: *ACM Transactions on Graphics (TOG)* 40.4 (2021).

[4] Yuanming Hu et al. "Taichi: a language for high-performance computation on spatially sparse data structures". In: *ACM Transactions on Graphics (TOG)* 38.6 (2019), p. 201.

[5] Haixiang Liu et al. "Narrow-band topology optimization on a sparsely populated grid". In: *ACM Transactions on Graphics (TOG)* 37.6 (2018), pp. 1–14.

References II

[6] Ken Museth. "VDB: High-resolution sparse volumes with dynamic topology". In: *ACM transactions on graphics (TOG)* 32.3 (2013), pp. 1–22.

[7] Rajsekhar Setaluri et al. "SPGrid: A sparse paged grid structure applied to adaptive smoke simulation". In: *ACM Transactions on Graphics (TOG)* 33.6 (2014), pp. 1–12.

[8] Andrew Spielberg et al. "Learning-in-the-loop optimization: End-to-end control and co-design of soft robots through learned deep latent representations". In: *Advances in Neural Information Processing Systems* 32 (2019), pp. 8284–8294.

Source Files

The source files live on github:
https://github.com/SallySoul/stencil_latex/tree/main/slides/taichi
$\approx$ 570 lines of latex
35 images