CS**5643**

01 Introduction

Physically Based Animation for Computer Graphics

Steve Marschner
Cornell University
Spring 2025

Physics Based Animation: History

Early work established a set of problems

Particle Systems: sparks, snow, fireworks; also fake fire, smoke, dust, …

Deformable bodies: rubber, soft tissue, cloth, string, …

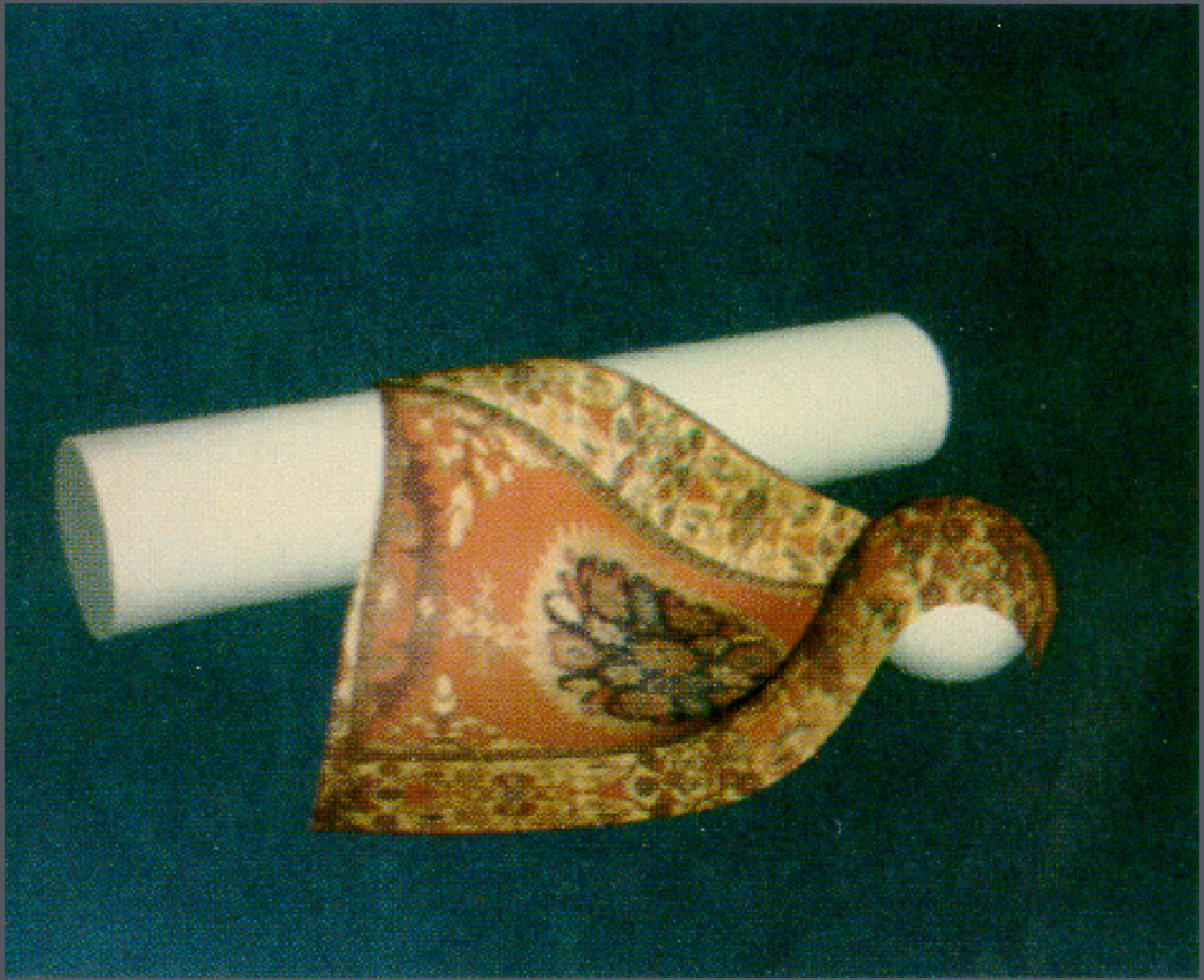
Rigid bodies: falling objects, fracture, …

Character motion: walking, running, jumping, …

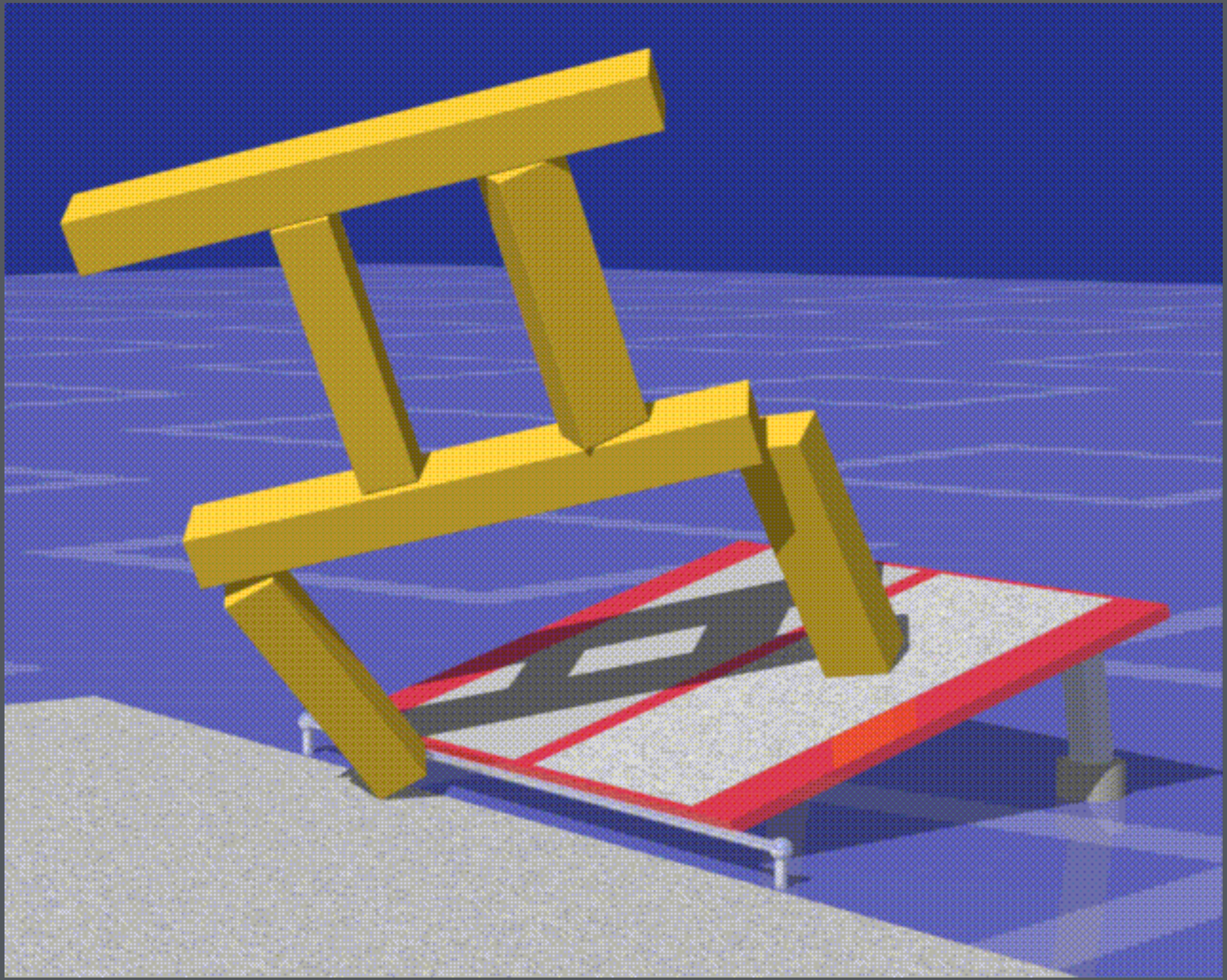
- hierarchies of rigid bodies

Fluids: water, smoke, …

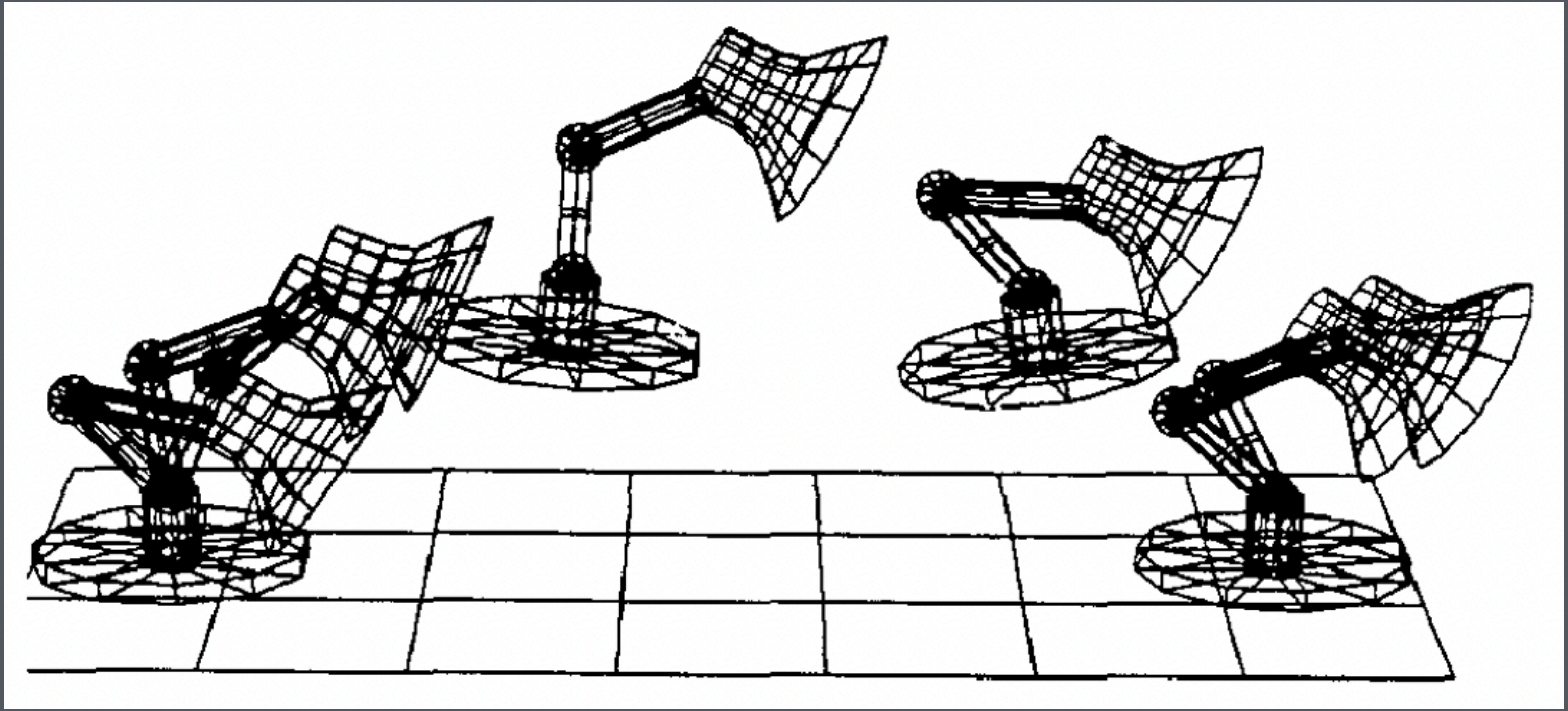
Particle Dreams [Karl Sims, 1988]



Terzopoulos, Platt, Barr, & Fleischer. “Elastically Deformable Models,” 1987



David Baraff, 1991



Witkin & Kass. “Spacetime Constraints,” 1988

Foster & Metaxas. “Modeling the Motion of a Hot, Turbulent Gas,” 1997

Physics Based Animation: Progress!

Physics of all these things mainly understood

Simulation for graphics has particular goals:

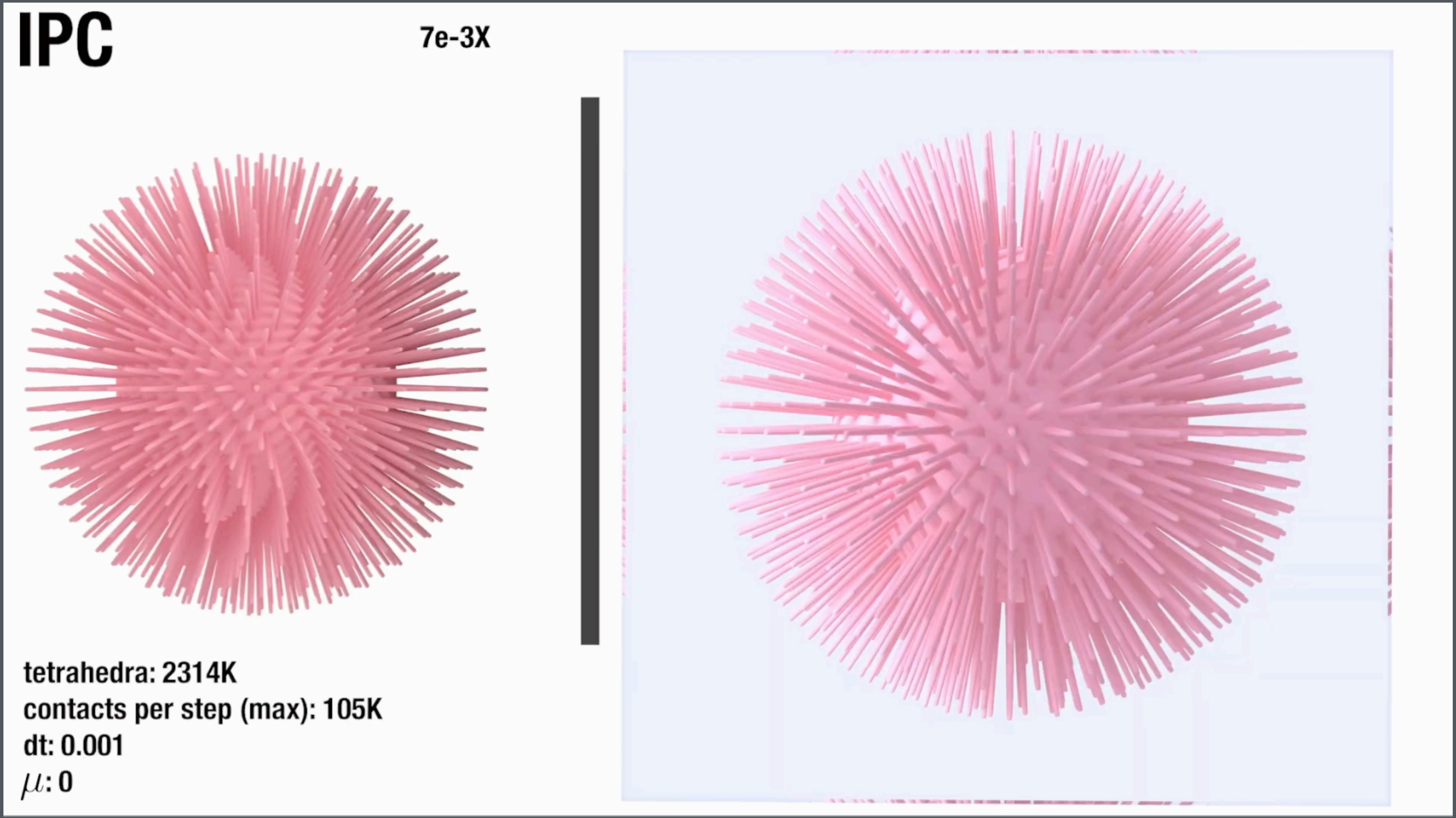
- scalability and efficiency
- generality
- stability and robustness
- usability and controllability
- visual fidelity to reality

These goals drive a particular style of simulation

- engineering applications need accuracy or there is no point
- animation applications need generality and robustness or there is not point

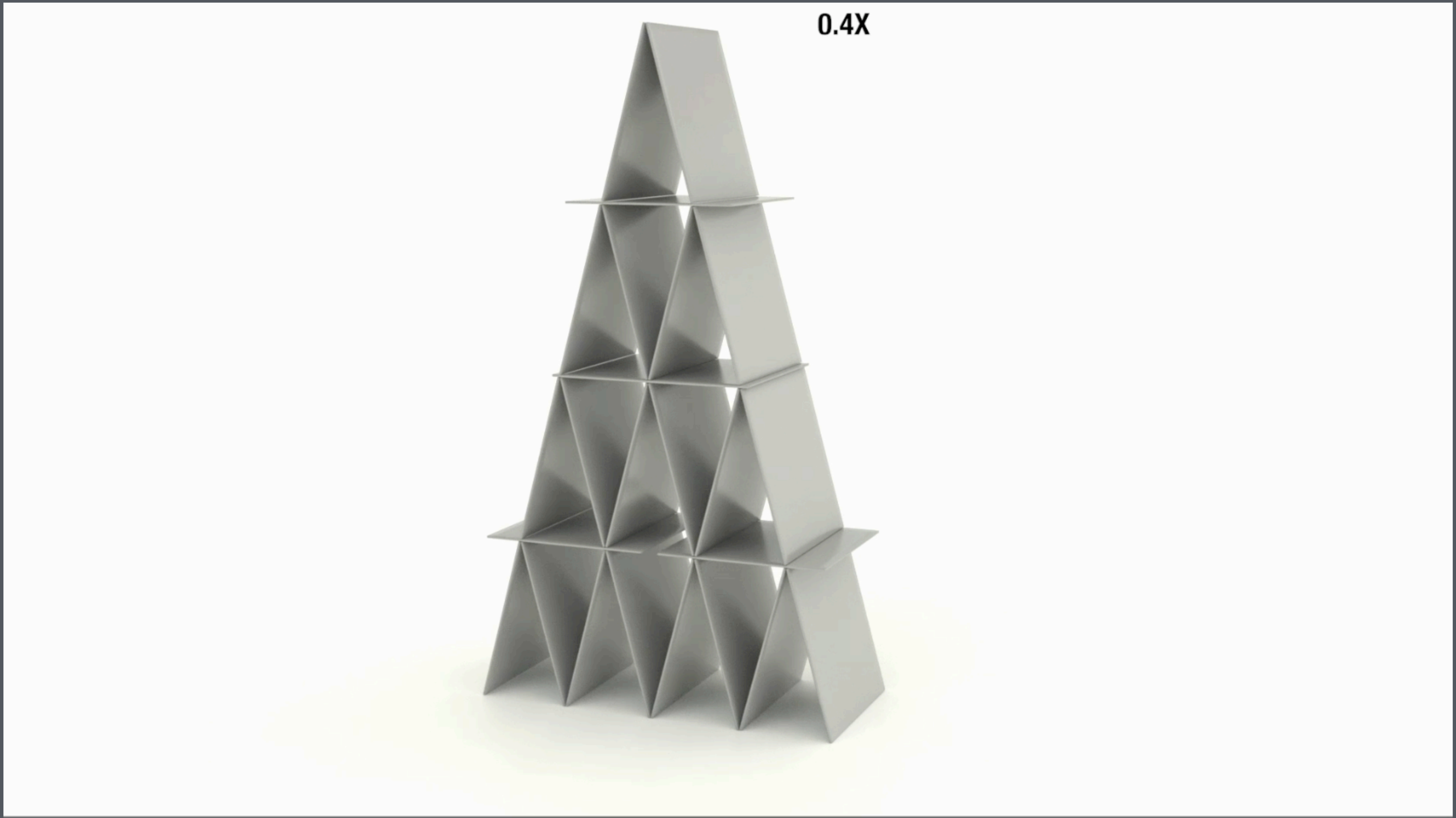
Efficient yarn-based cloth [Kaldor et. al, SIGGRAPH 2010]

https://www.cs.cornell.edu/projects/YarnCloth/



Incremental Potential Contact [Minchen Li et. al, SIGGRAPH 2020]

https://ipc-sim.github.io/



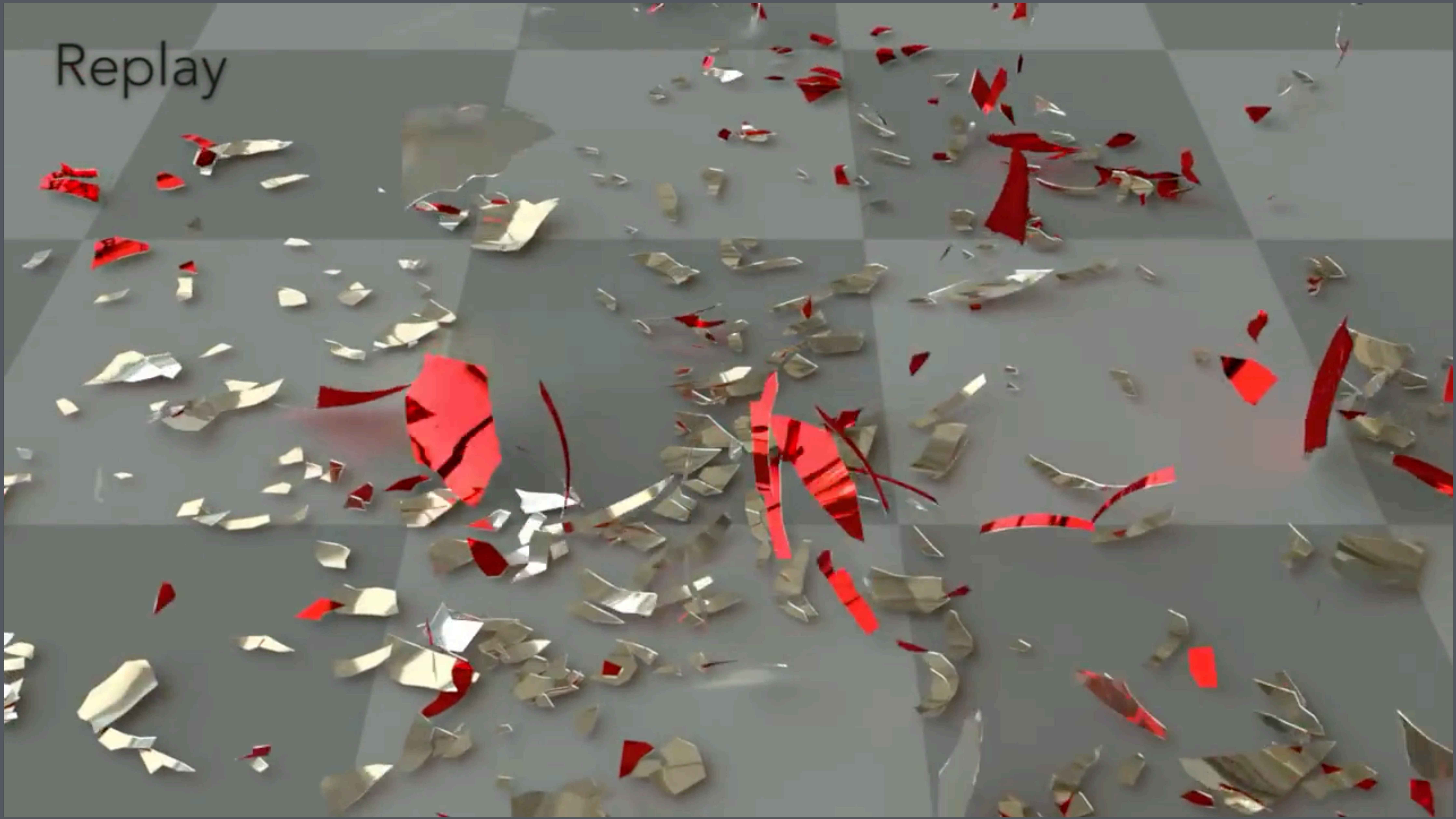
Incremental Potential Contact [Minchen Li et. al, SIGGRAPH 2020]

C-IPC: Inelastic Thickness with Constraint Offset

88K nodes
Contacts/step (max):
2.2M
h: 0.04s
2x playback speed

Codimensinonal Incremental Potential Contact [Minchen Li et. al, SIGGRAPH 2021]

https://ipc-sim.github.io/C-IPC/

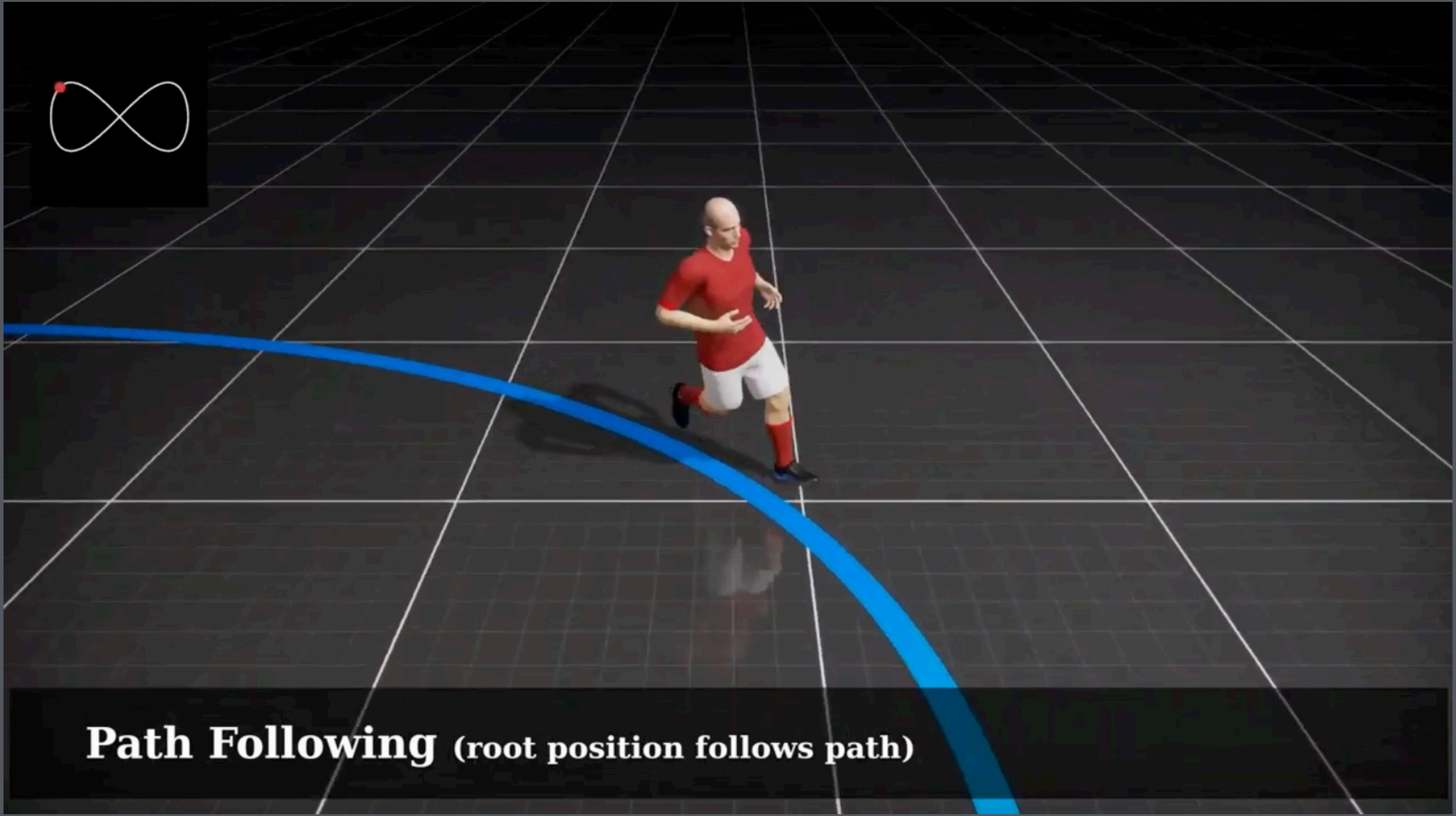


Adaptive Tearing and Cracking of Thin Sheets [Pfaff et al. SIGGRAPH 2014]

https://www.youtube.com/watch?v=ATU6IGCMpUA

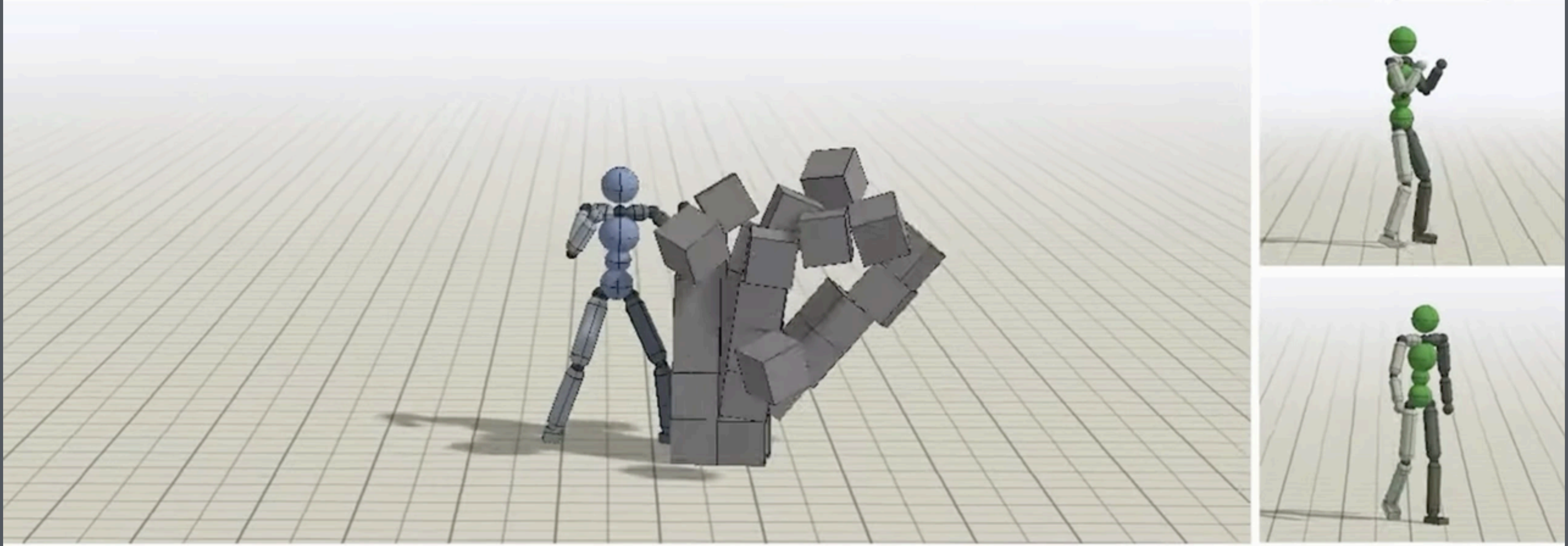
Real-Time Dynamic Fracture (NVIDIA demo 2013)

https://www.youtube.com/watch?v=ATU6IGCMpUA



Character Controllers Using Motion VAEs [Hung Yu Ling et al. SIGGRAPH 2020]

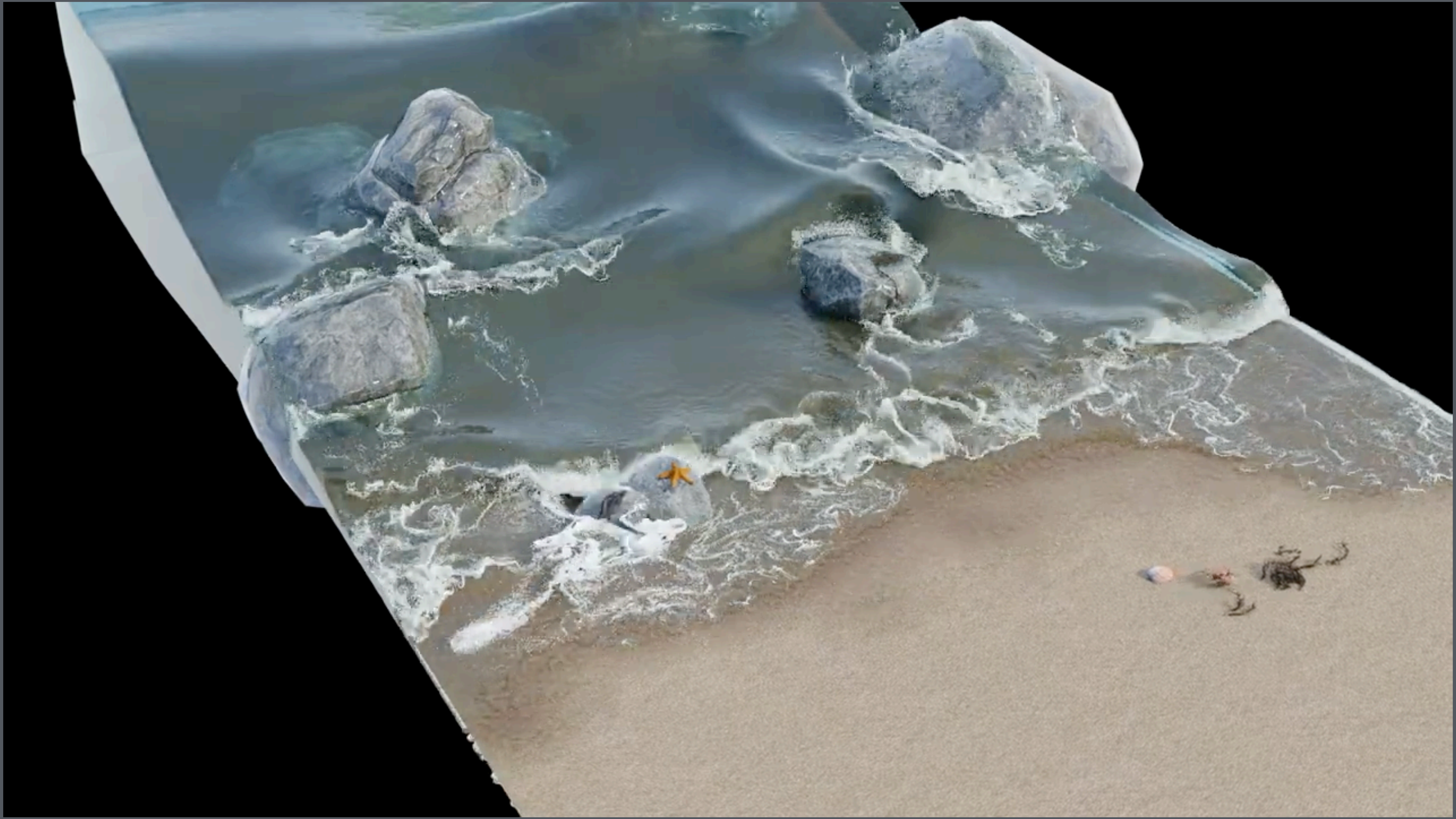
Humanoid: Strike (Walk + Punch)



When the target is far away, the character imitates various walking motions in order to move towards the target.

Adversarial Motion Priors for Stylized Physics-Based Character Control [Xue Bin Peng et al. SIGGRAPH 2021]

Schrödinger's Smoke [Chern et al. SIGGRAPH 2016]



Beach waves in Blender (Max Nadolny using Blender's FLIP simulator)

https://www.youtube.com/watch?v=J62bviAnL9I

Unified Particle Physics for Real-Time Applications [Macklin et. al, SIGGRAPH 2014]

Course overview

Organized around assignments

- mass-spring systems and deformables
- collisions and rigid body motion
- fluids

Final project

- take something we did in 2D, make it 3D
- take something we did in one assignment, make it work with another
- do something we didn't make it to in the assignments (particle systems? character motion?)

Course website

3-week per-assignment structure

Written problem set (work together freely, write up solo)

- primarily paper & pencil
- sometimes include small numerical experiments in NumPy
- goal: understand the basic math & physics behind the implementation

Implementation project (solo or in pairs)

- implement standalone demos in Python + Taichi
- some 2D, some 3D simulations
- goal: get familiar with the details and get experience with this style of code

Analysis assignment (solo or in pairs)

- do experiments on your implementation
- written or NumPy problems about the performance of the methods
- goal: understand the limitations and when these methods work and don't

Prerequisites

Things you would learn in a graphics course (e.g. 4620)

- transformations and hierarchies
- meshes and triangles
- spatial data structures

Things you would learn in math courses (e.g. Math 1920/2940)

- calculus and vector calculus (Taylor series, div, grad, curl, …)
- linear algebra (linear transformations, rank, null space, …)

Things you would learn in physics courses (e.g. AP Physics, Physics 1112)

- Newtonian mechanics (force, torque, momentum, angular velocity, …)

I will assume you have heard of this stuff but might be rusty :)

Introductions

Steve Marschner (prof.)

- research area = realism, modeling materials
- yarn-based cloth simulation
- wave-based material appearance simulation

Caroline Sun (PhD TA)

- research areas = yarn-based simulation;
  imaging & photography

Yi (Bill) Xu (PhD TA)

- research area = physics based simulation

Zane Neelin (MEng TA)

cloth mechanics
yarn-based cloth modeling

Jonathan Kaldor, Doug James, and Steve Marschner. "Simulating Knitted Cloth at the Yarn Level." SIGGRAPH 2008

Jonathan Kaldor, Doug James, and Steve Marschner. "Efficient Yarn-based Cloth with Adaptive Contact Linearization." SIGGRAPH 2010

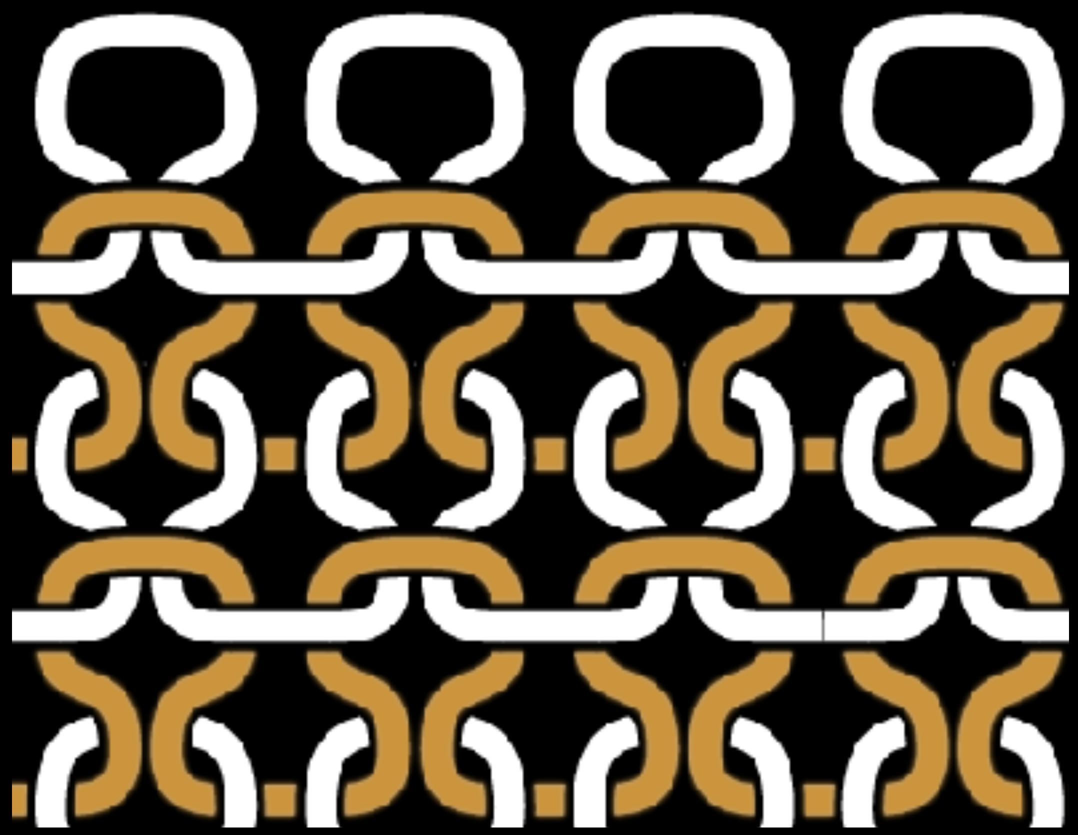
Cem Yuksel, Jonathan Kaldor, Doug James, and Steve Marschner. "Stitch Meshes for Modeling Knitted Clothing with Yarn-level Detail." SIGGRAPH 2012

Why Yarns Are Important

- Cloth is not a continuum
  - Discrete yarn behavior drives overall cloth behavior
- Particularly evident in knit fabrics

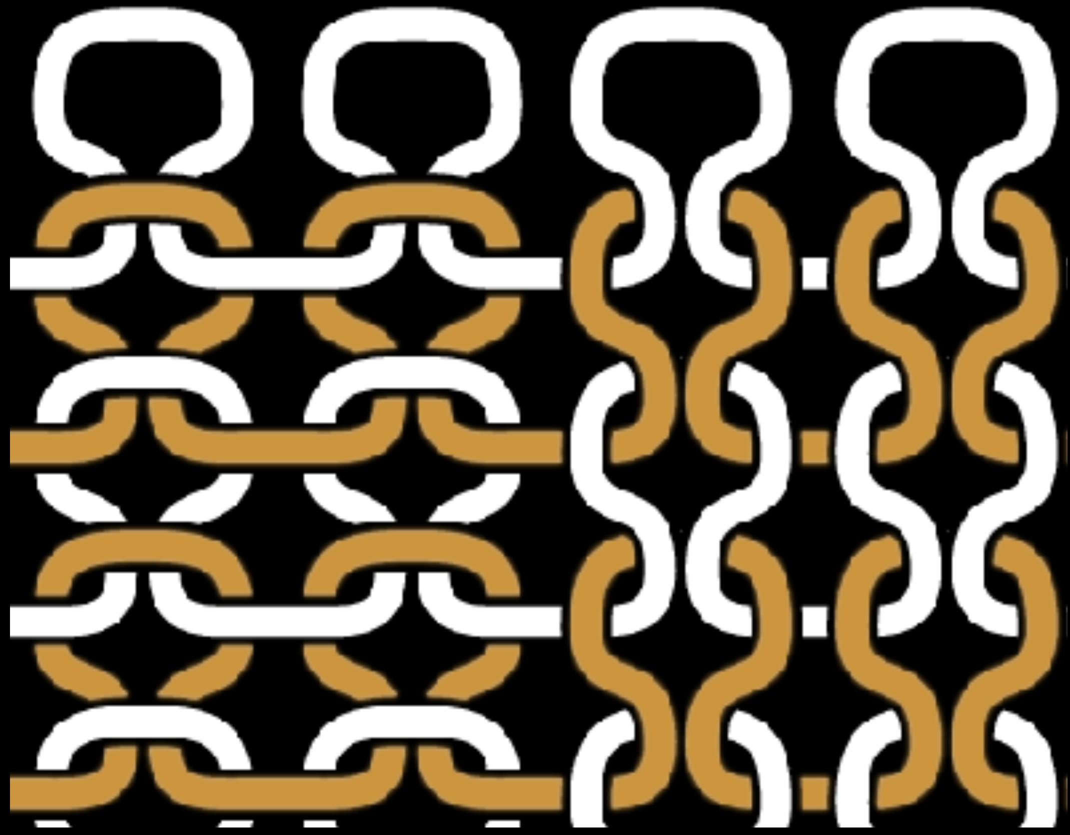
http://toveb.typepad.com/

Structure-Dependent Behavior



Garter

Stockinette



Rib

Yarn Properties

- Thin, flexible rods, with many degrees of freedom
- Strongly resist stretching
- Weakly resist bending
- Can compress laterally
- Friction between yarns

- Constrained Lagrangian dynamics

$$\mathbf{M}\ddot{\mathbf{q}} = \mathbf{f} - \nabla E - \nabla D$$

$$\mathbf{C}(\mathbf{q}) = 0$$

- Inextensibility constraints
- Bending, twisting energies
- Collision energy
- Velocity filter for damping

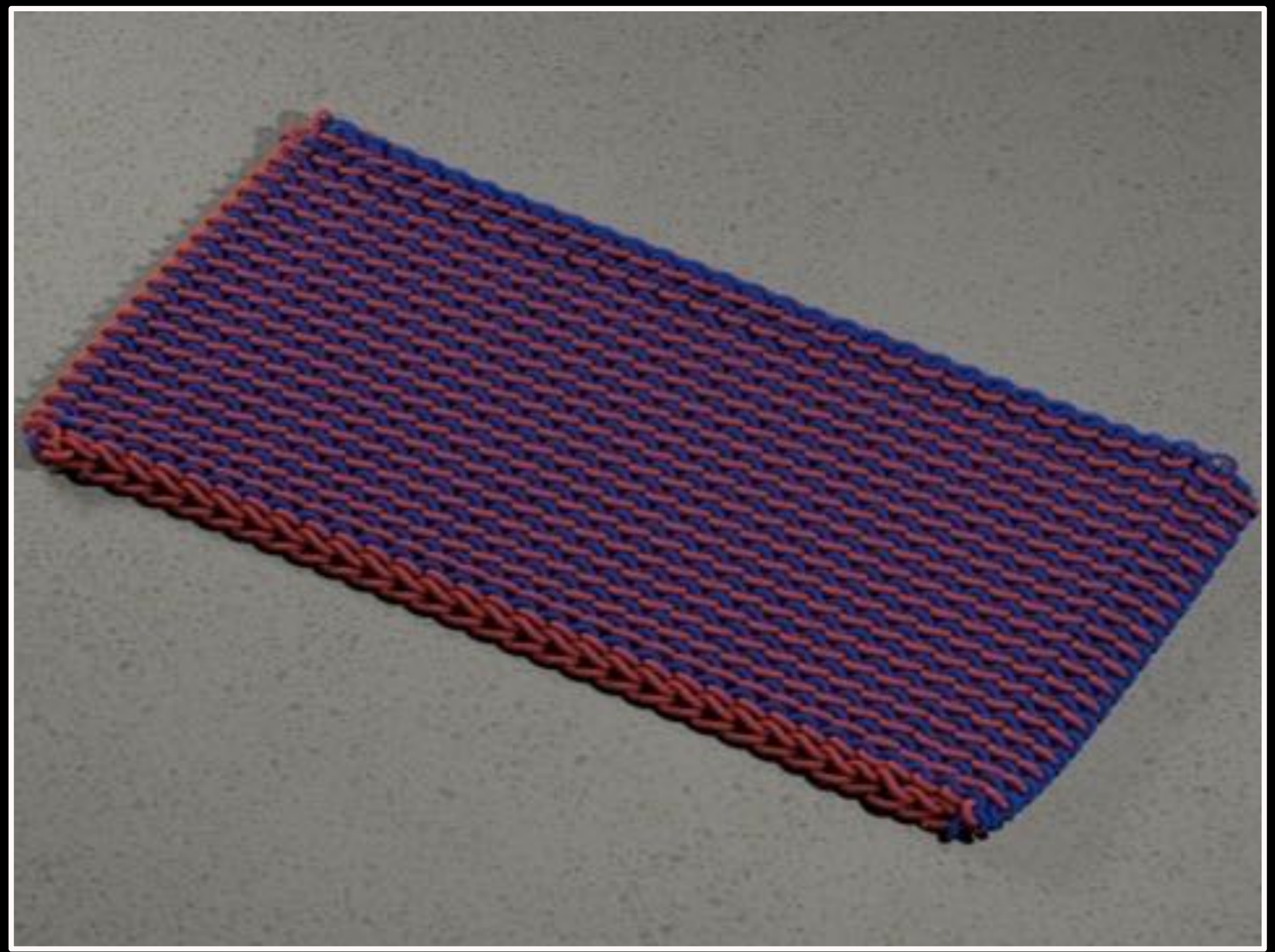
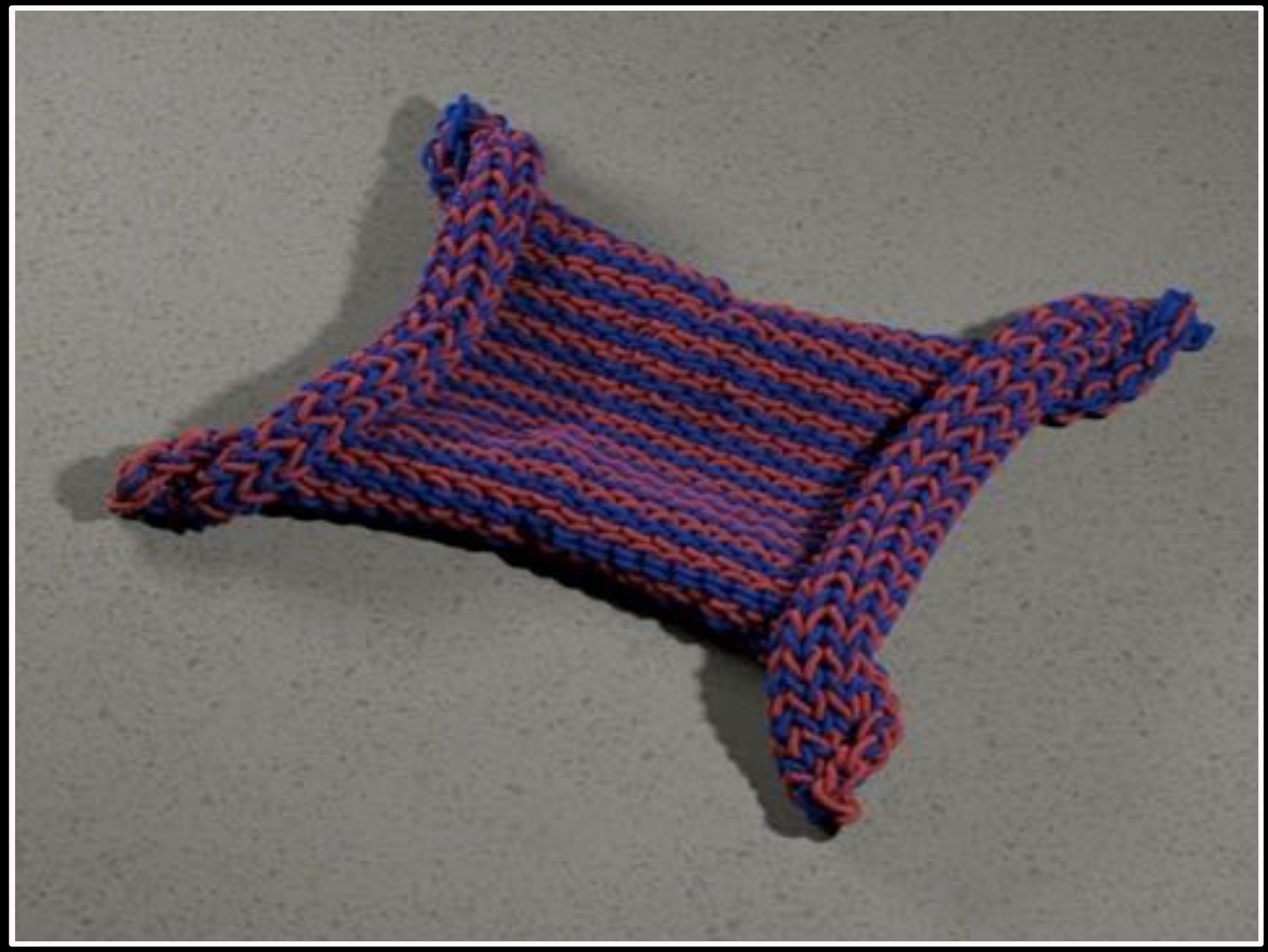
Modeling Dissipation

- Damp yarn-yarn contacts
- Damp non-rigid motion
  - [Müller et al. 2006]
    [Rivers and James 2007]
- Small regions: stabilizing collisions
- Large regions: damp cloth-level motion

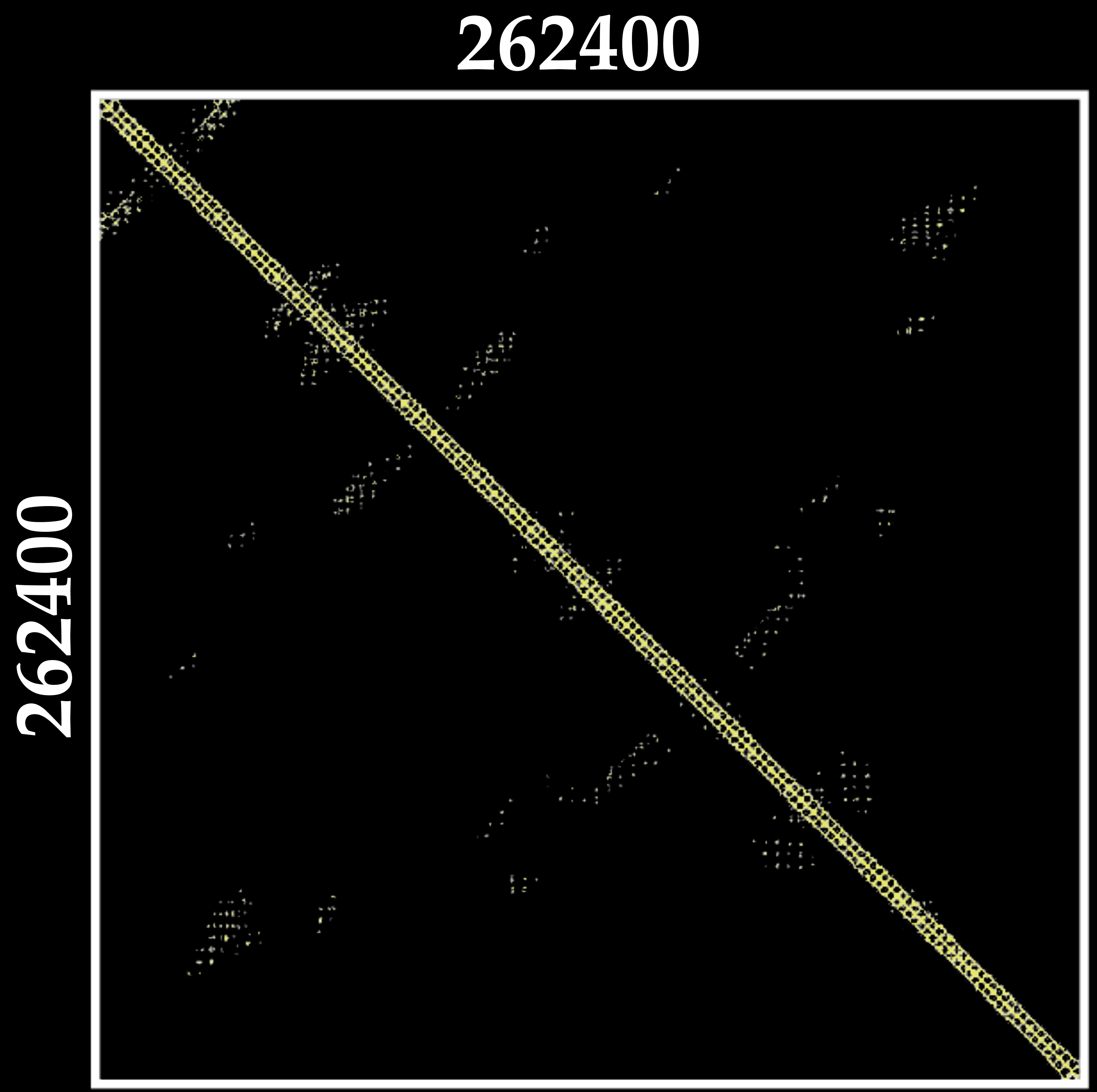
Relaxed Models

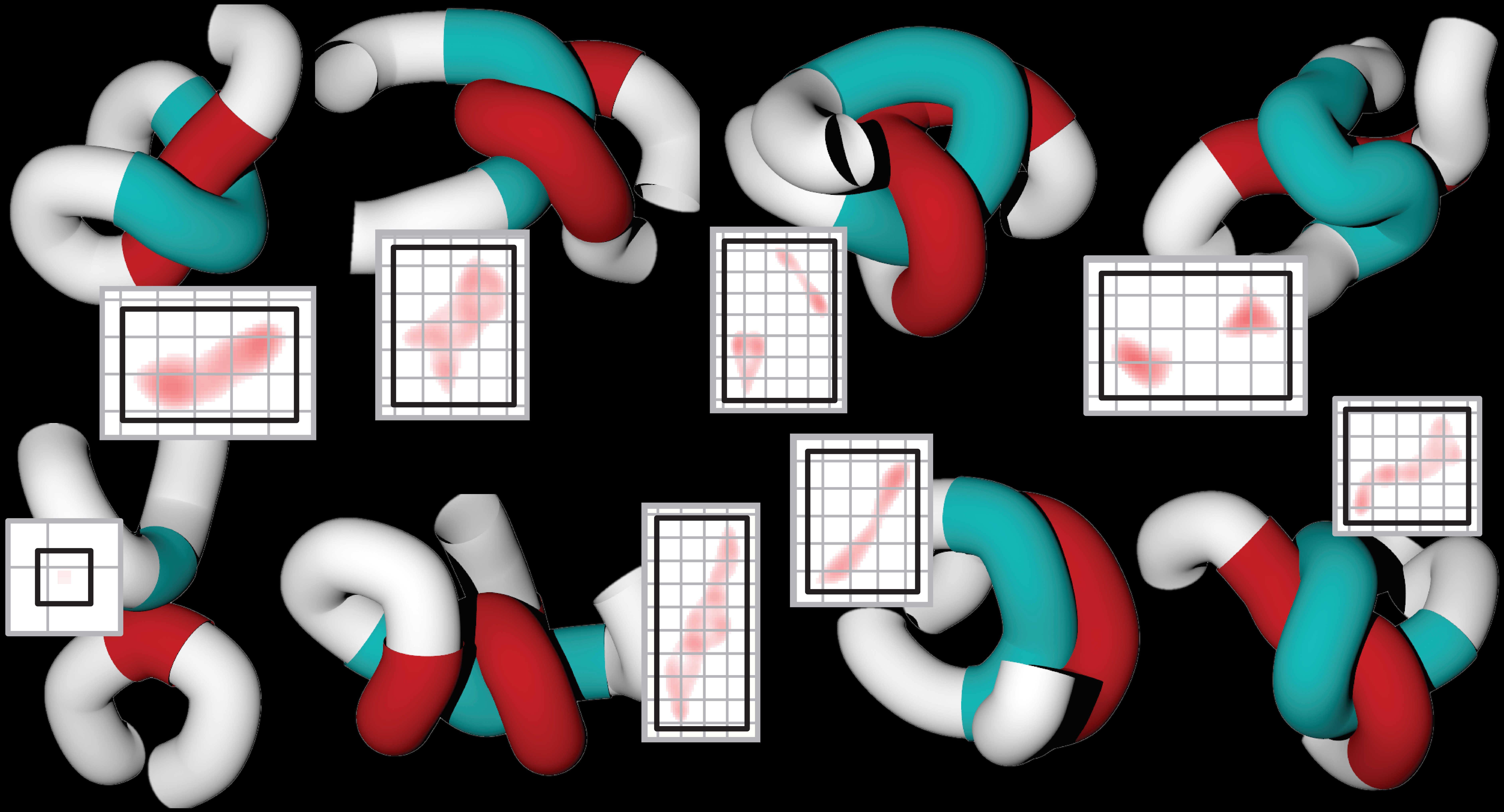


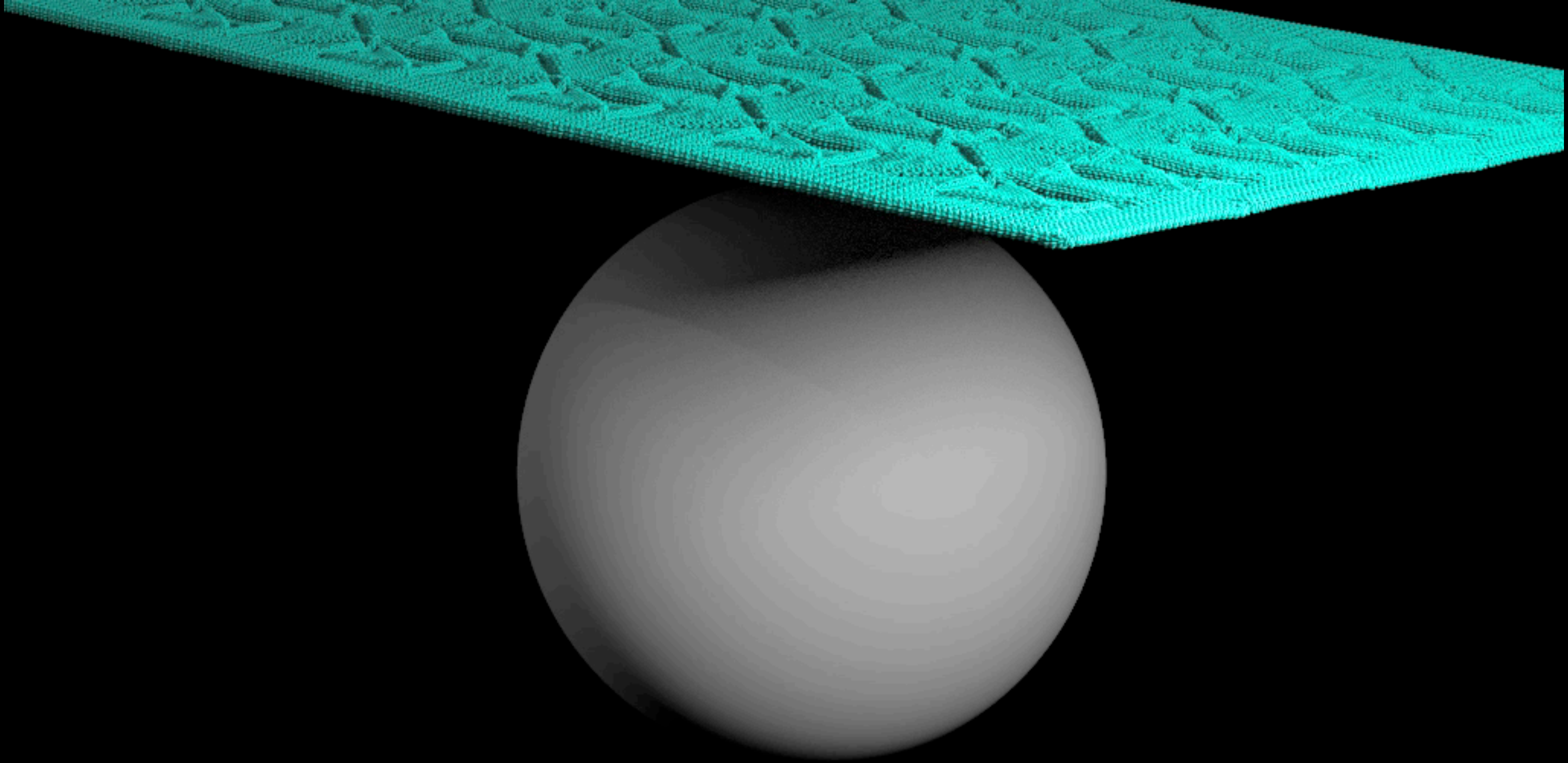
Garter Stockinette Rib


Contact Matrix







54,340 knit loops, ~365K contact sets
6.7X contact force speedup, 4.2X overall
10.5m per 1/30s frame

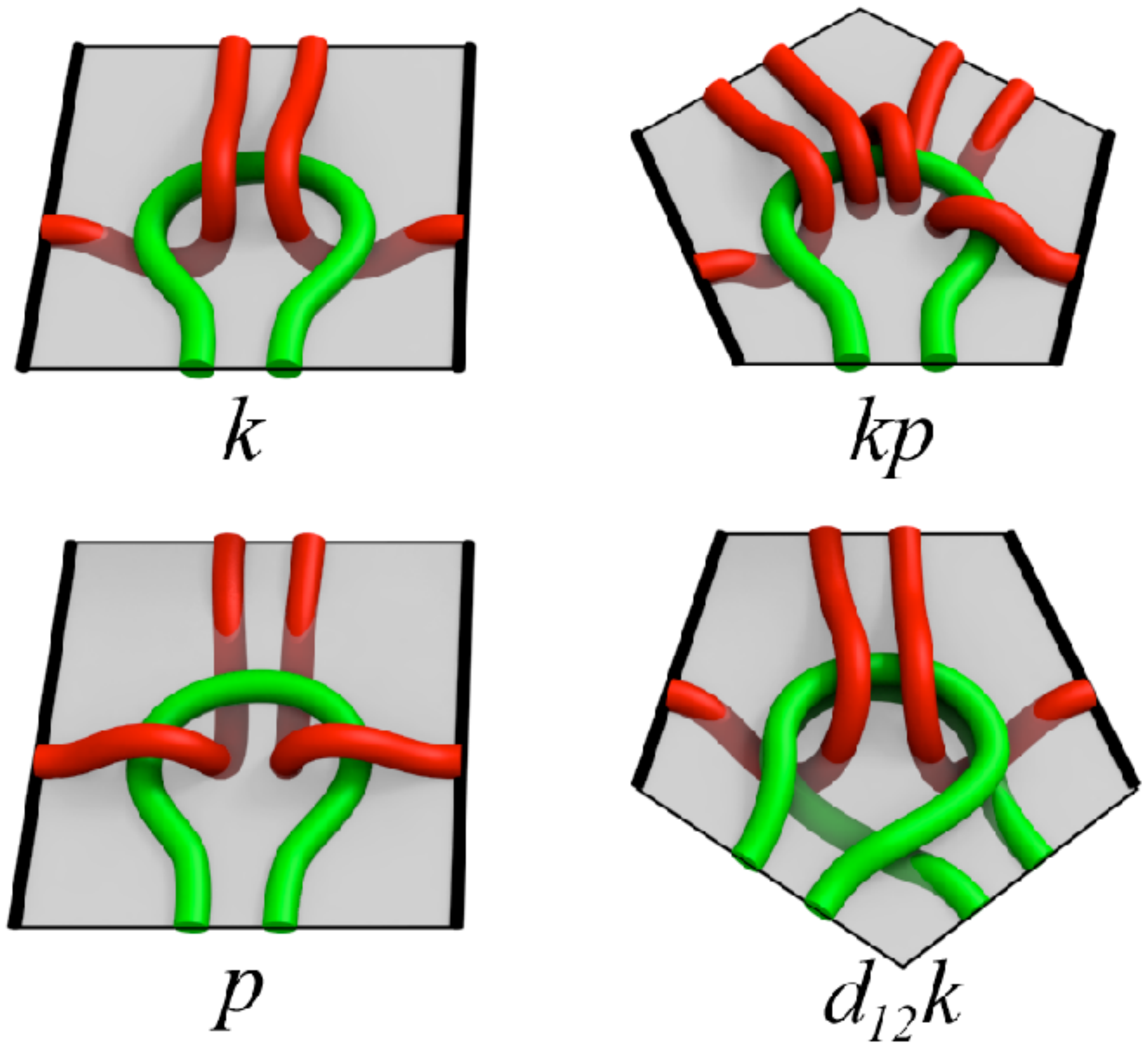
⅓ speed

45,960 knit loops, ~295K contact sets
9.1X contact force speedup, 5.0X overall
8m per 1/30s frame

½ speed

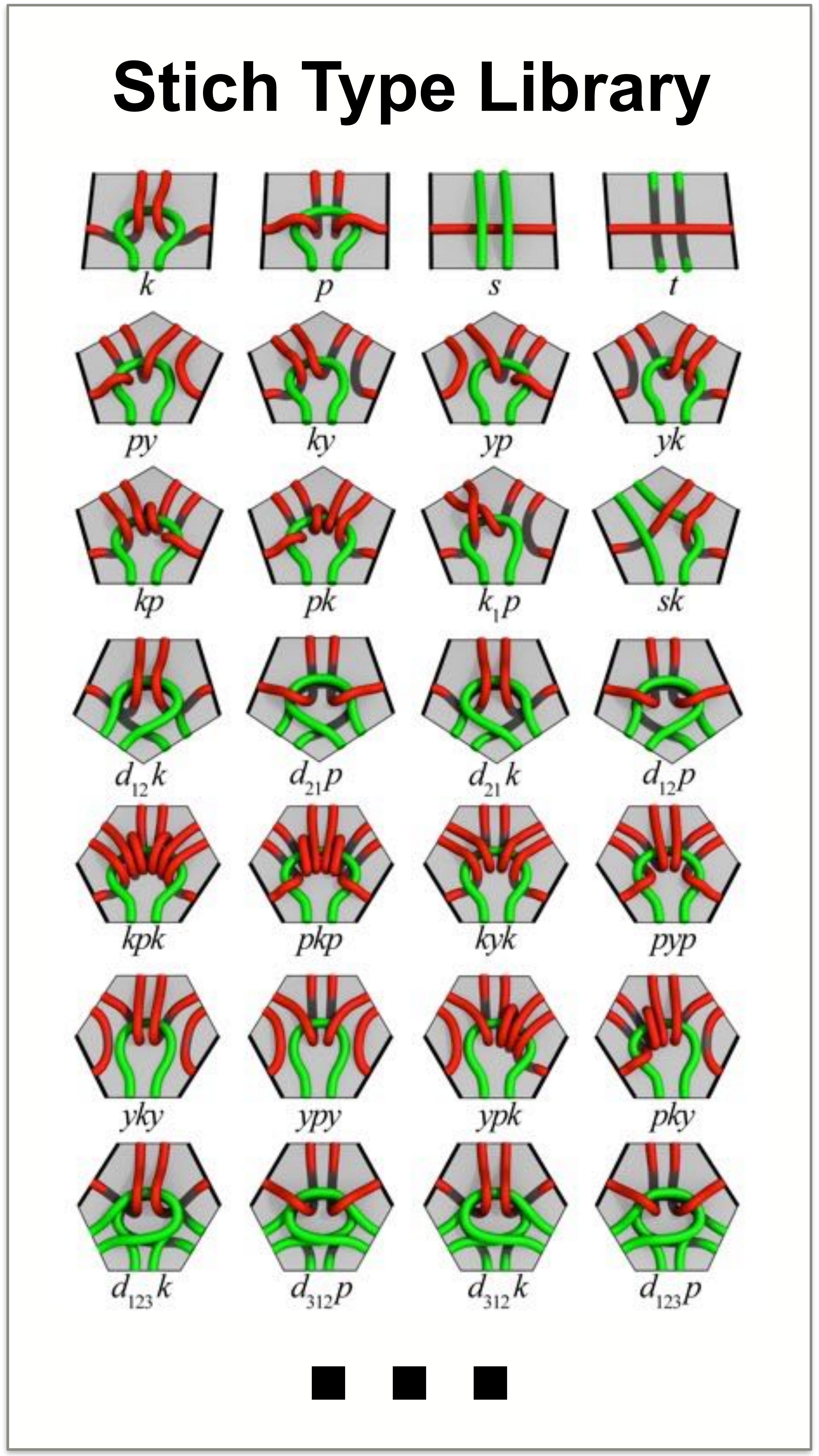
Stitch Meshes

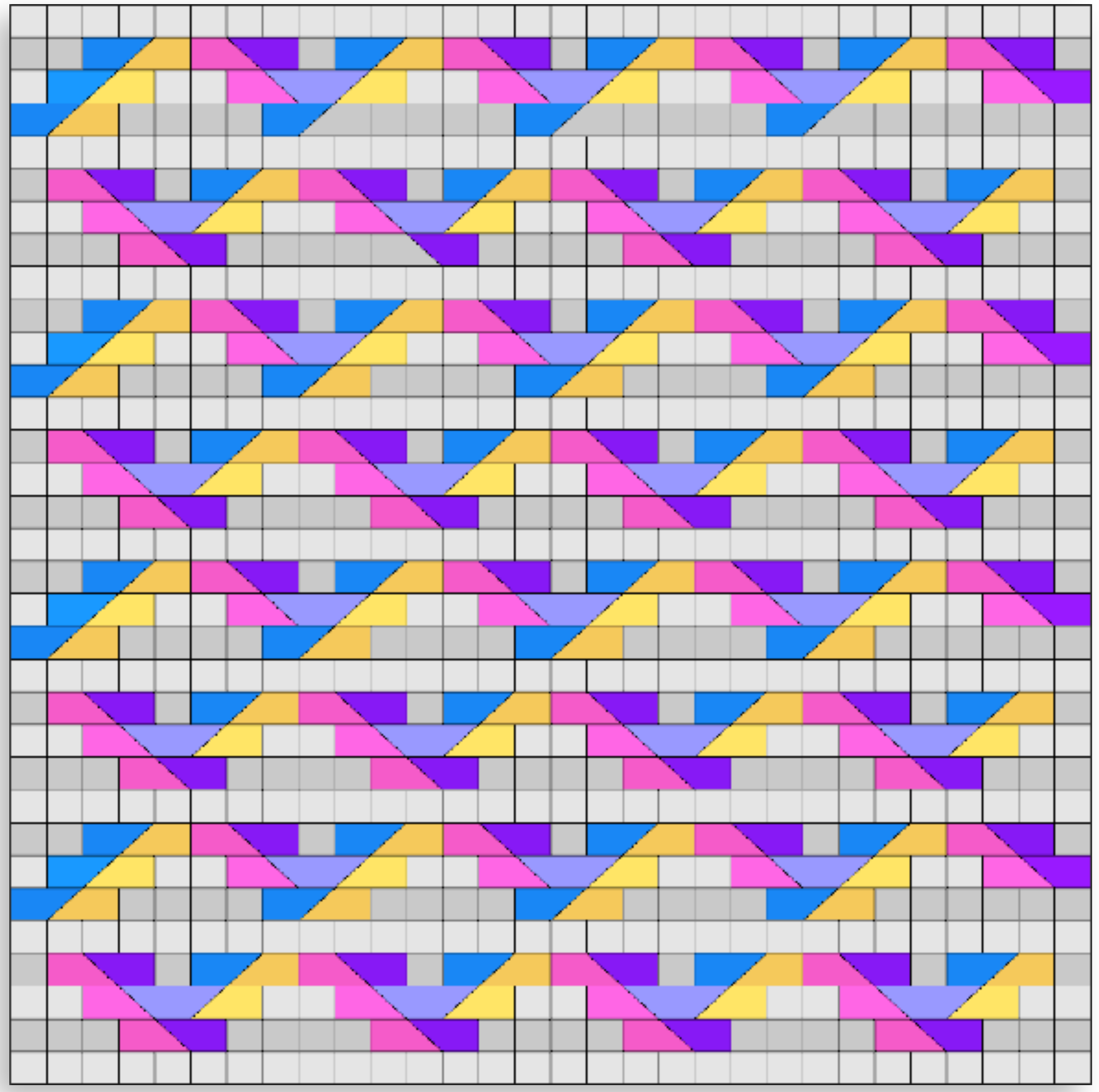
Stitch Mesh



Stitch Mesh Faces

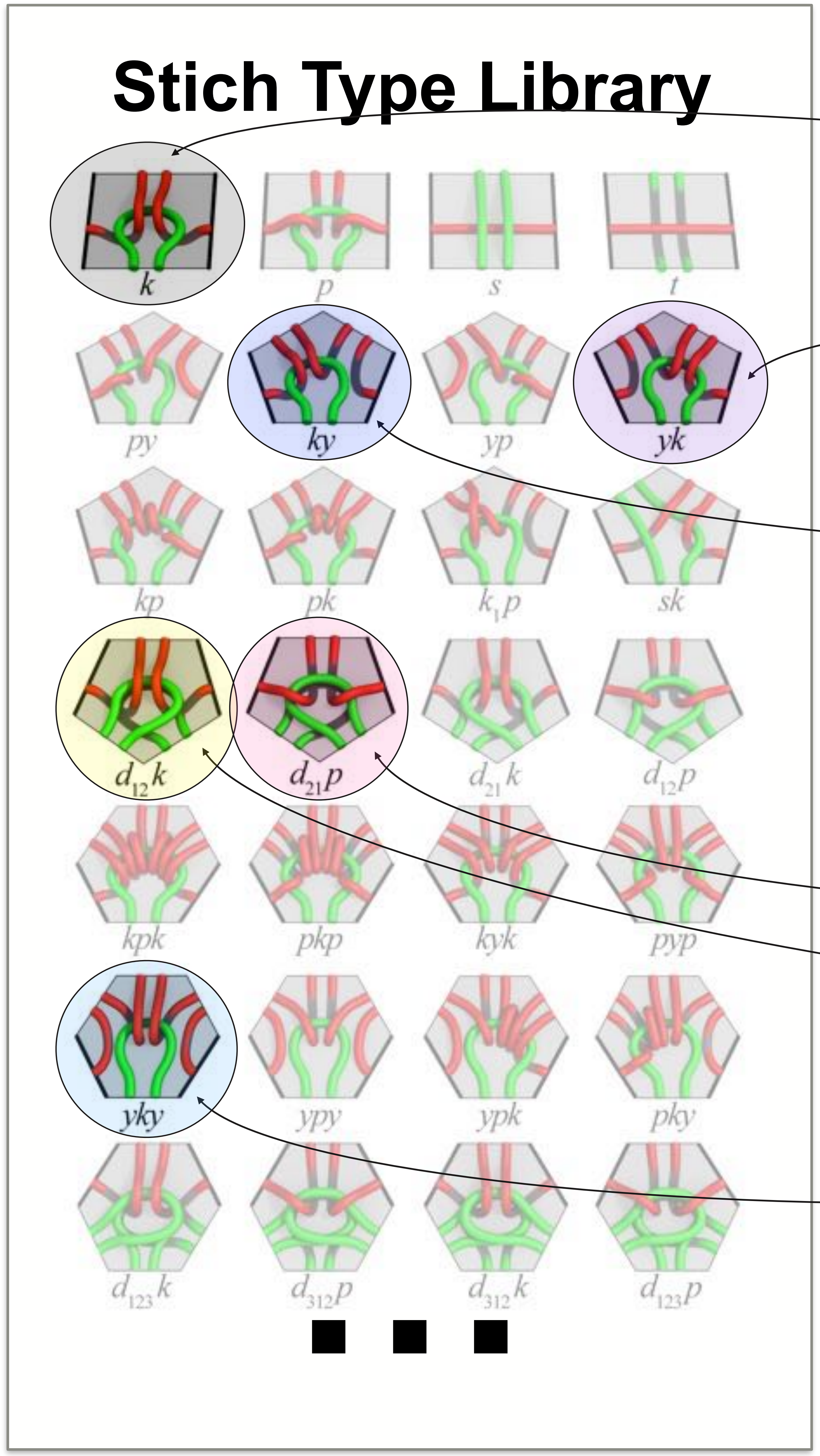
Stitch Type Library

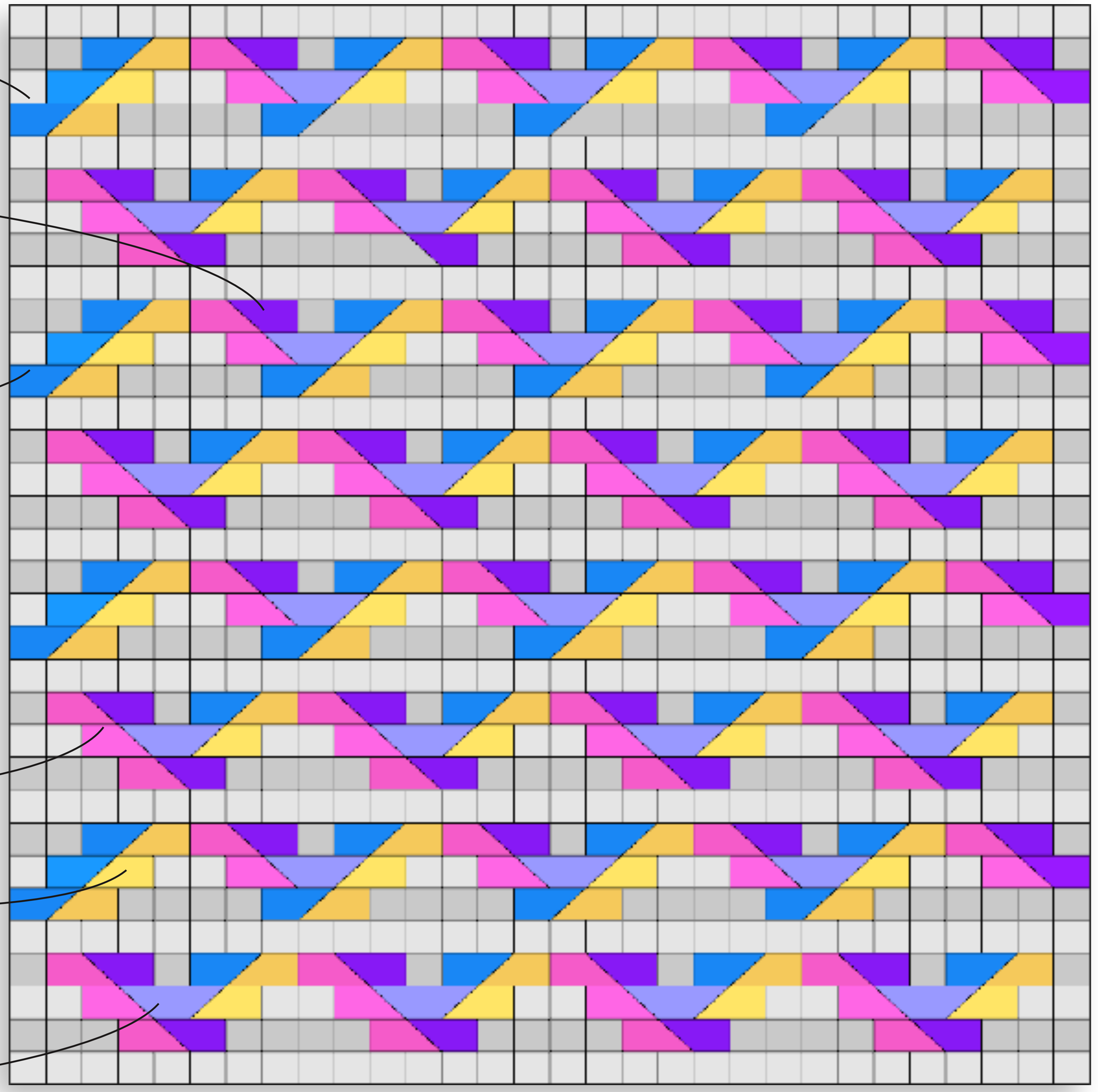




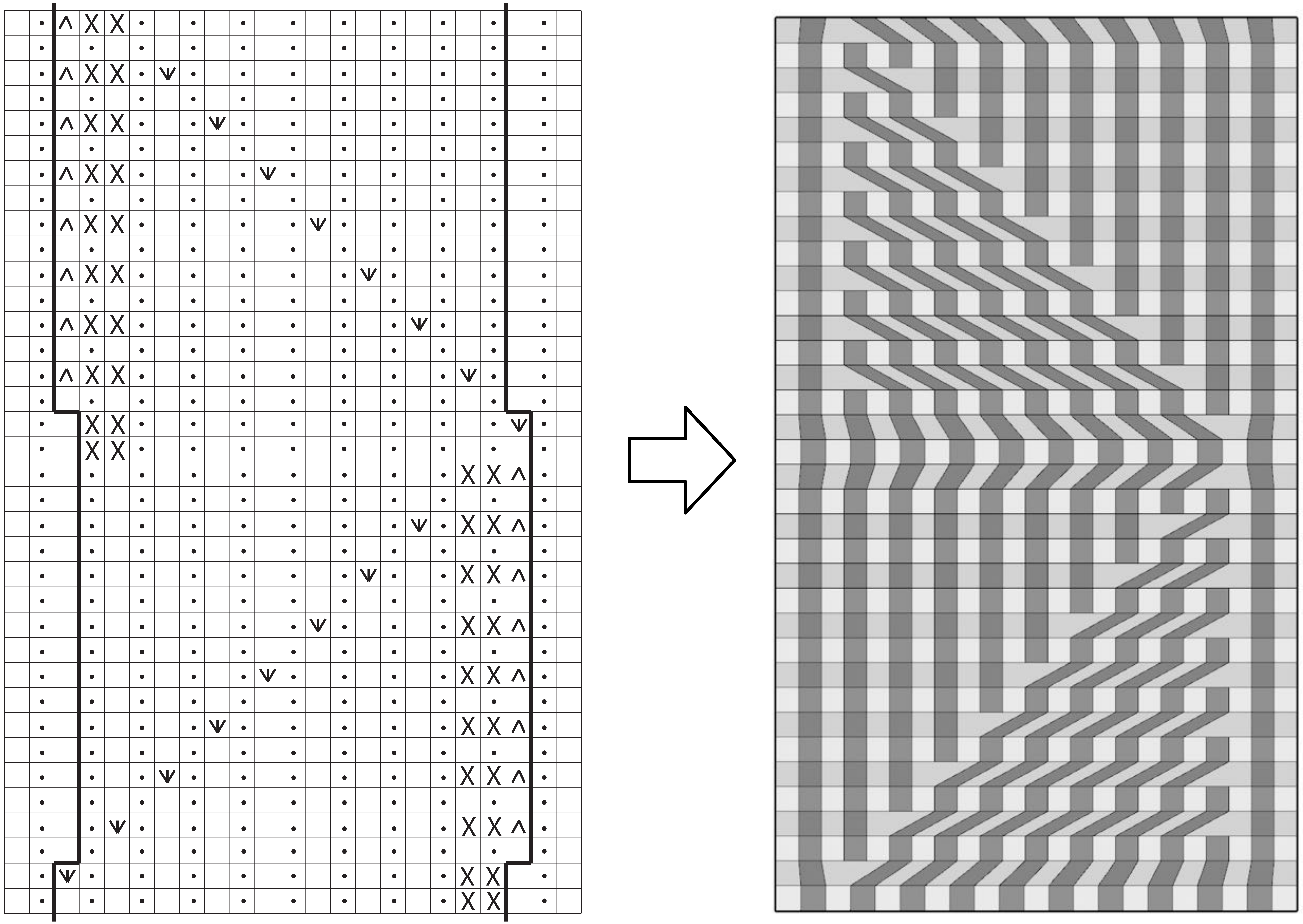
Stitch Mesh

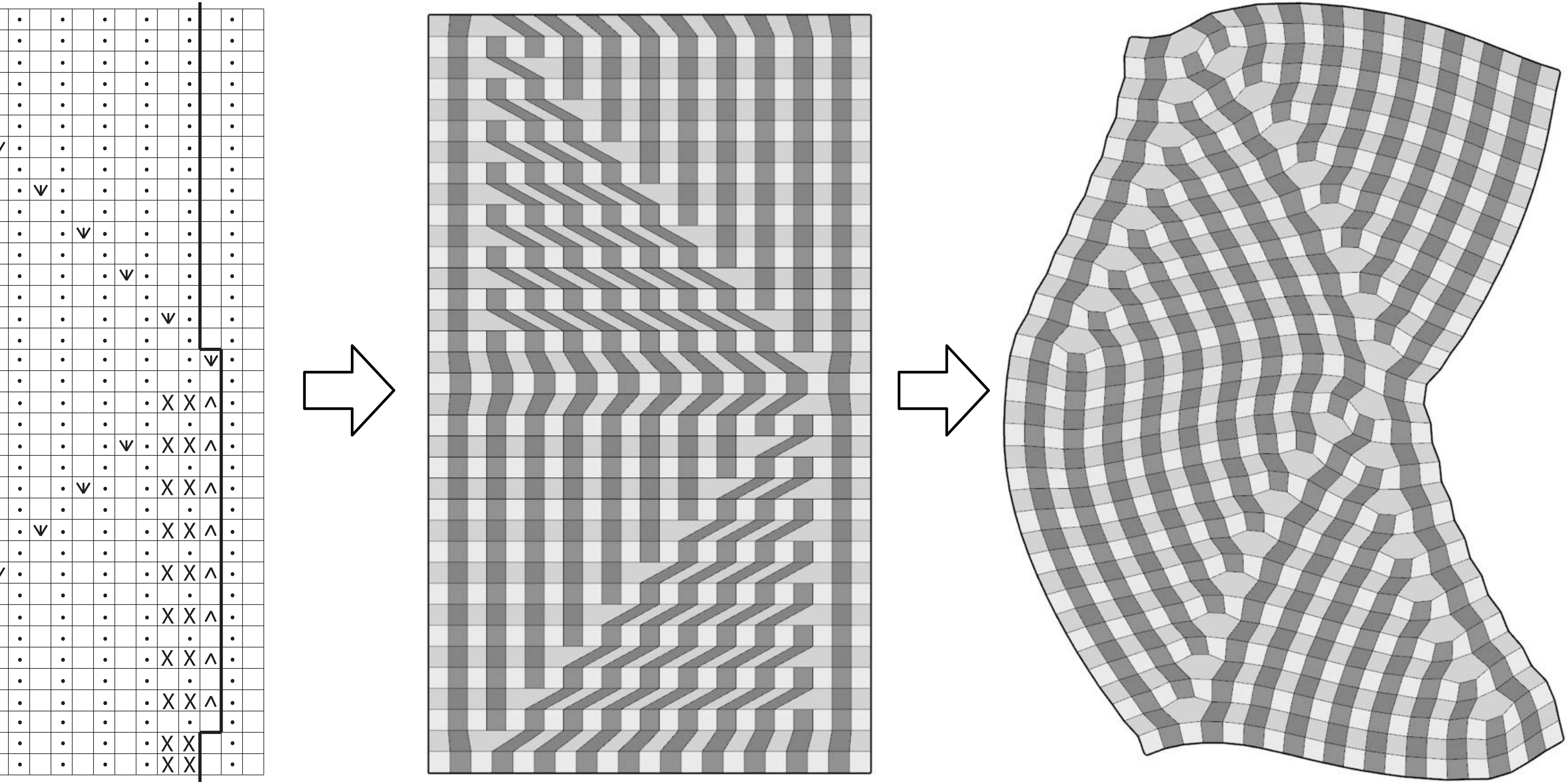
Stitch Type Library

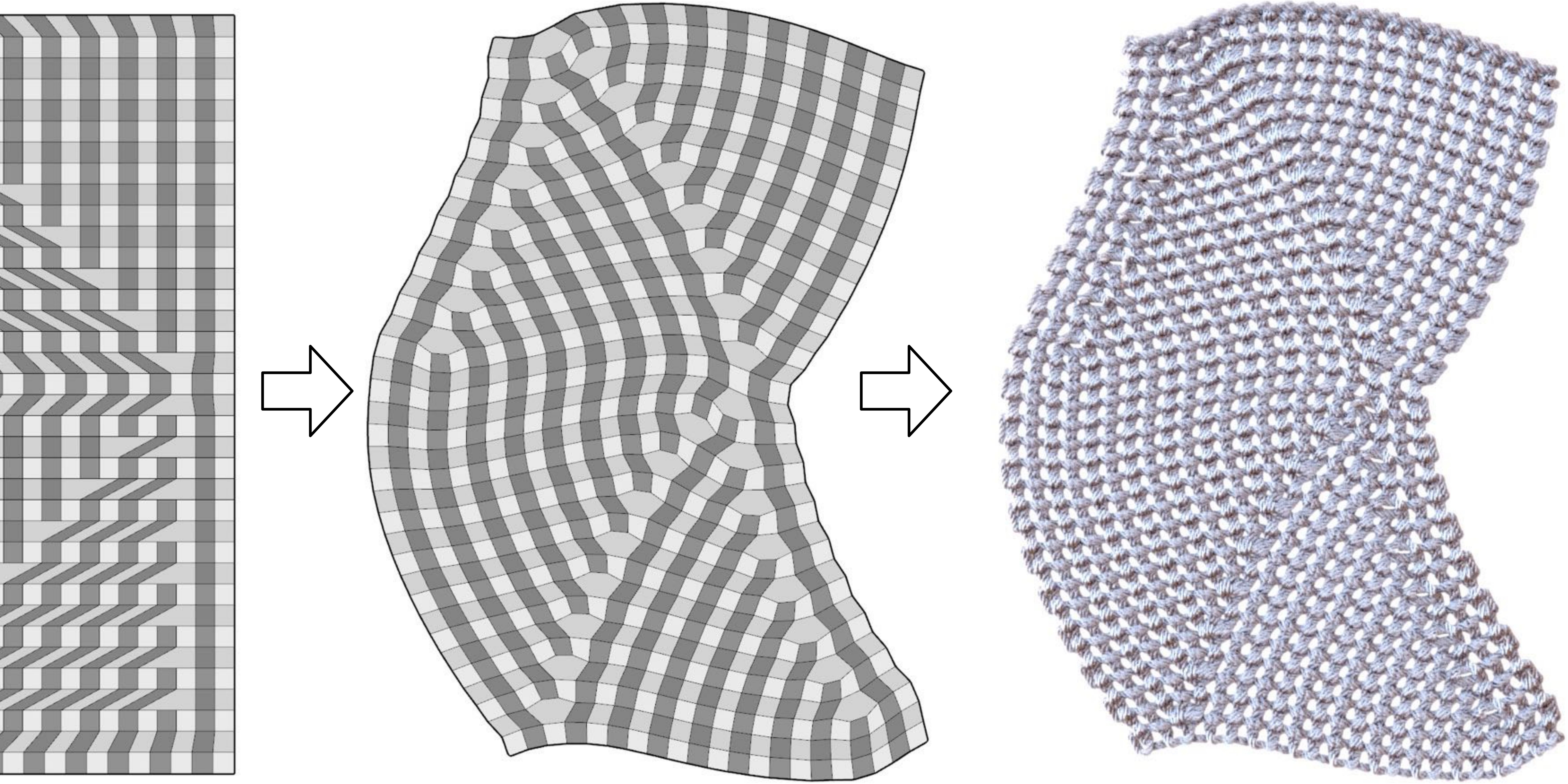


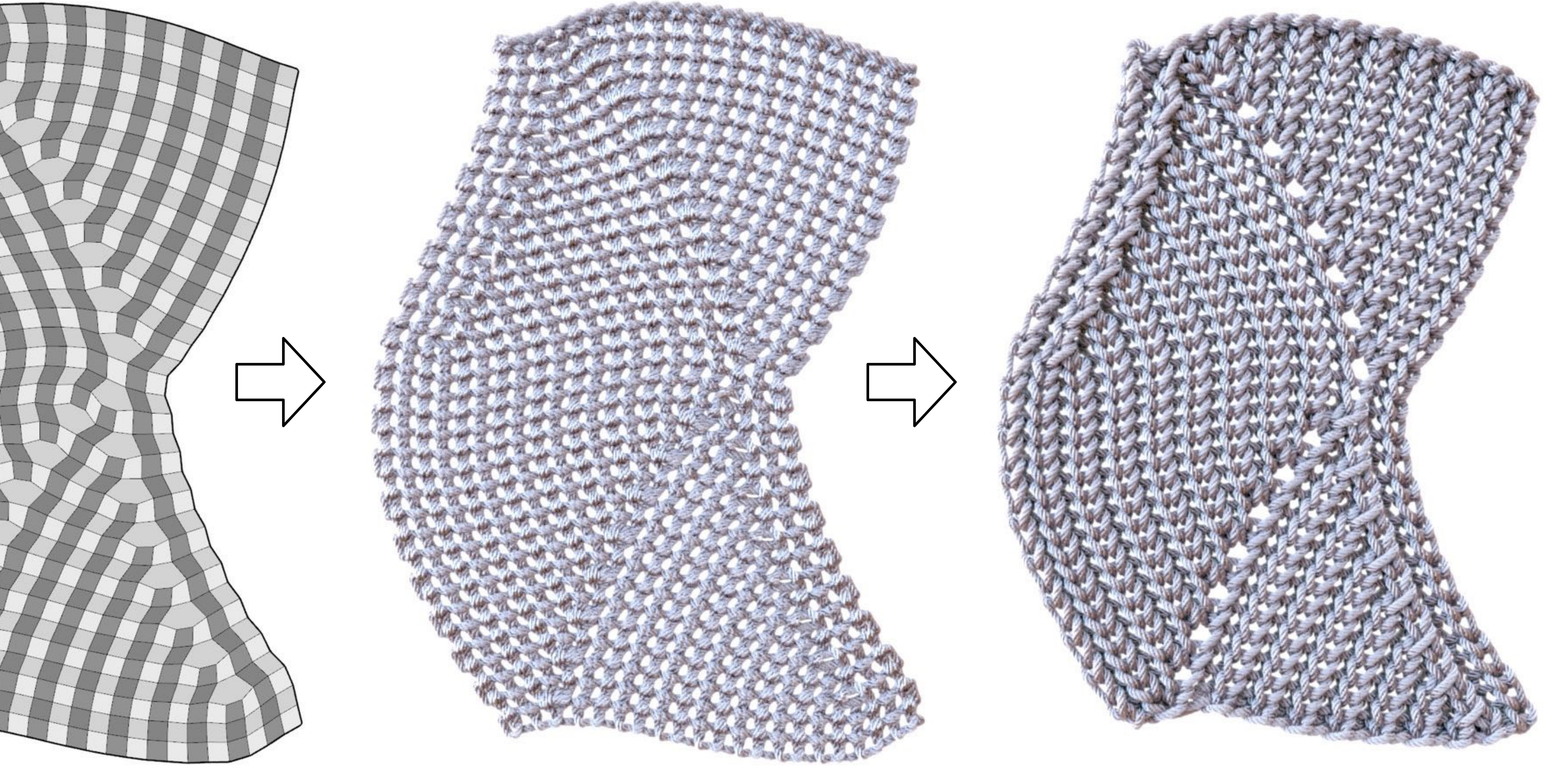


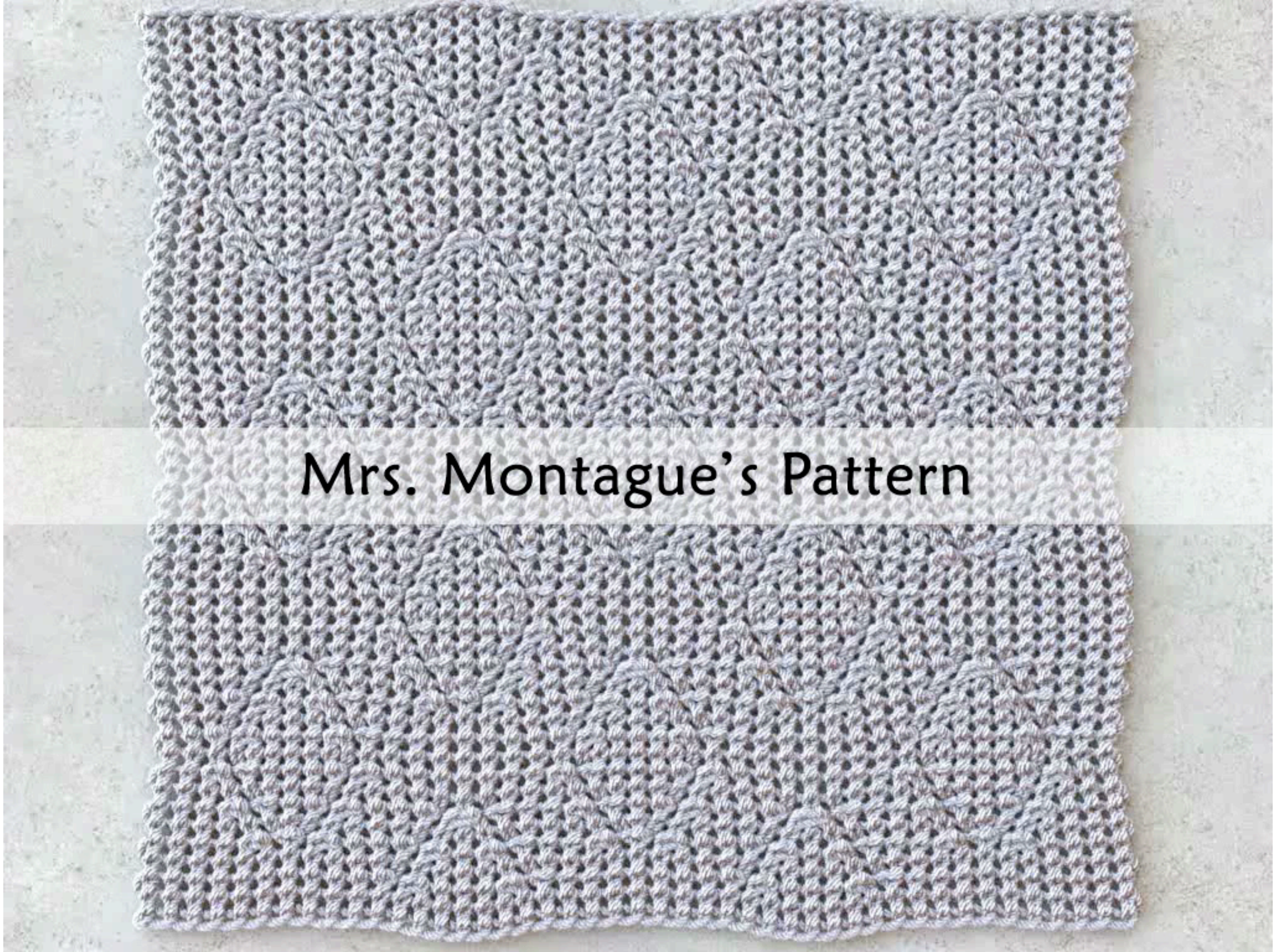
Stitch Mesh

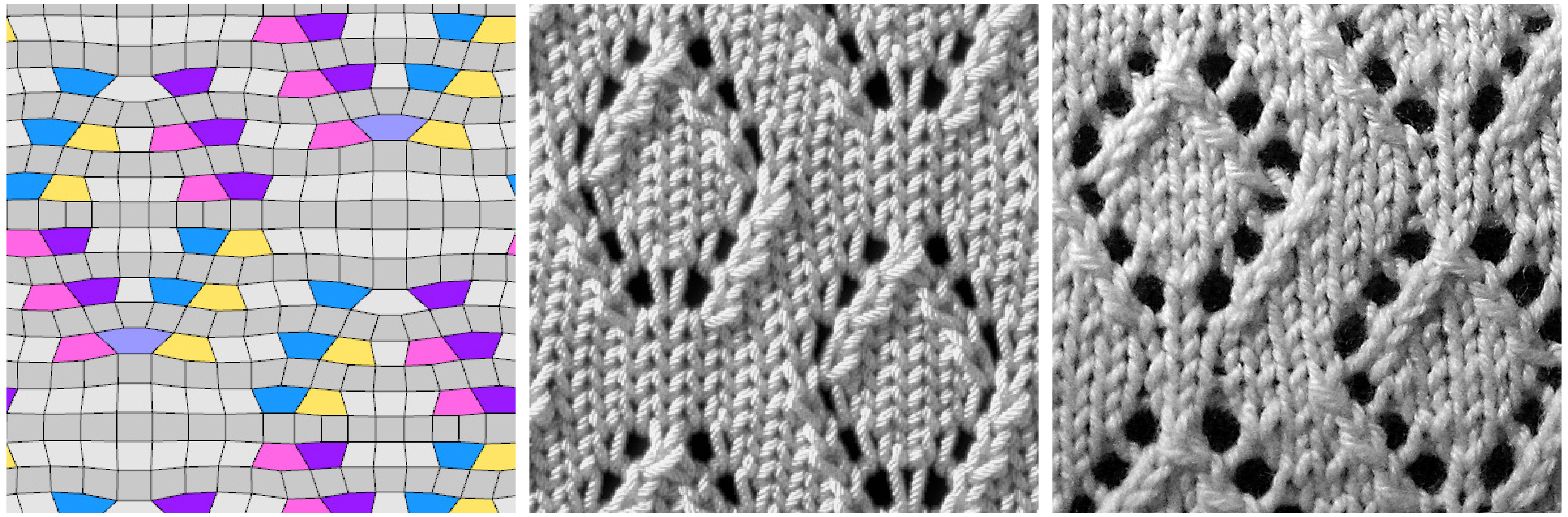




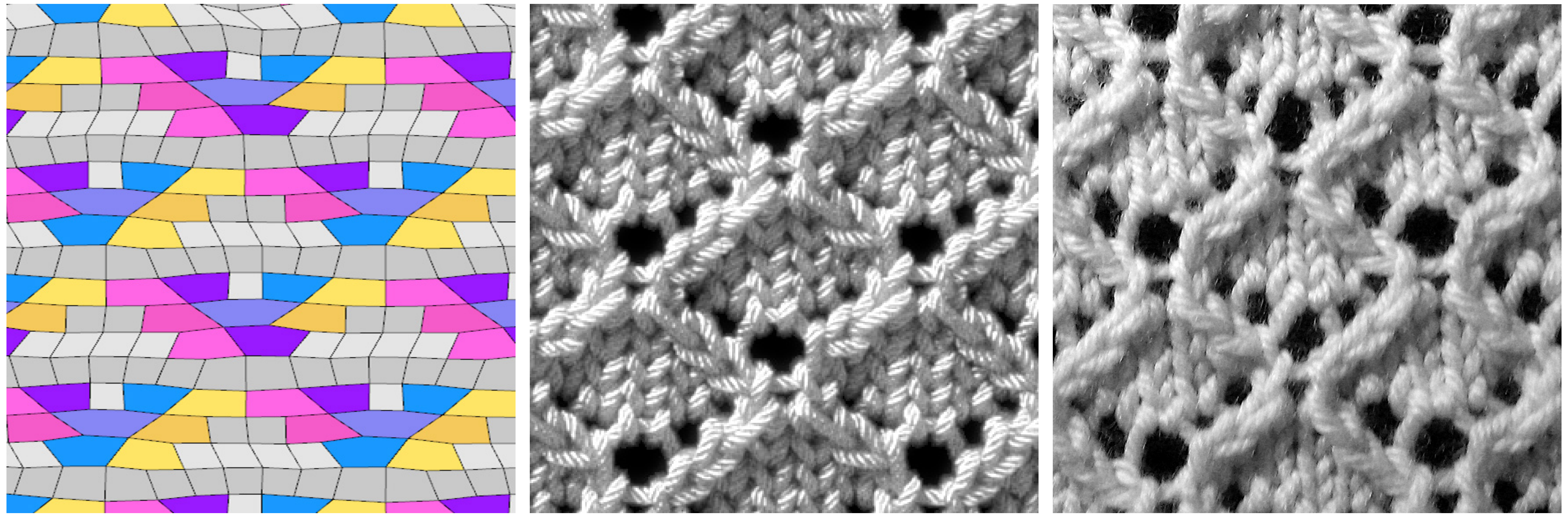




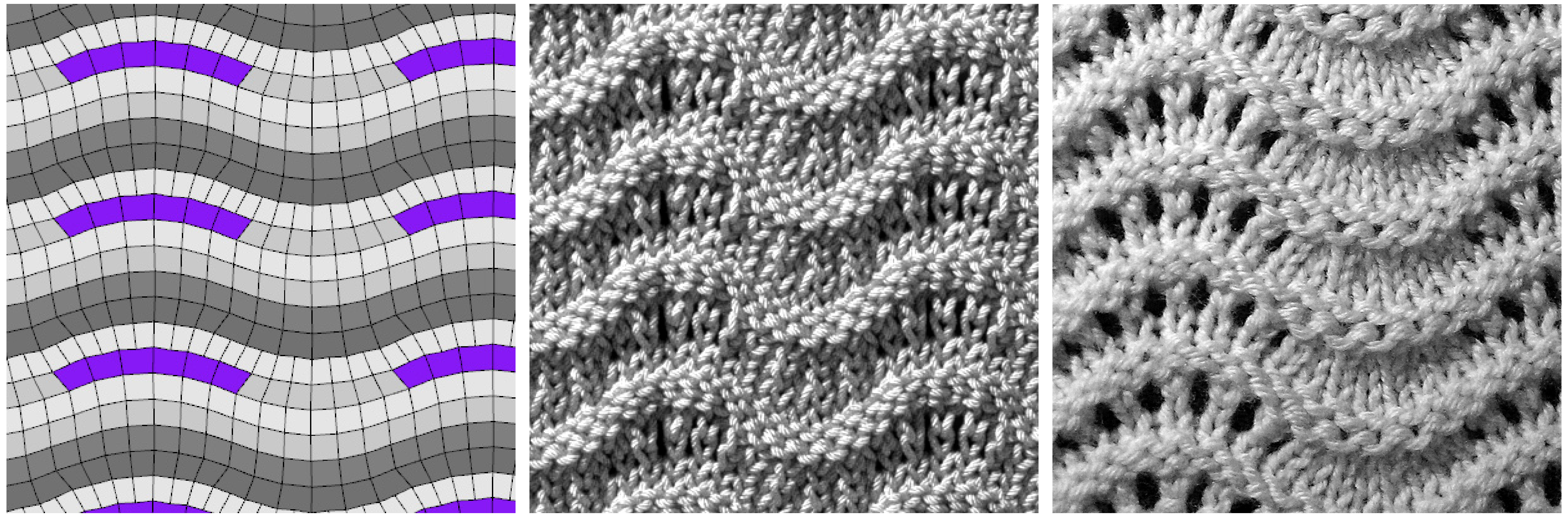

Mrs. Montague's Pattern



Mrs. Montague's Pattern [Matthews 1984]



Openwork Trellis Pattern [Matthews 1984]



Ridged Feather Pattern [Matthews 1984]

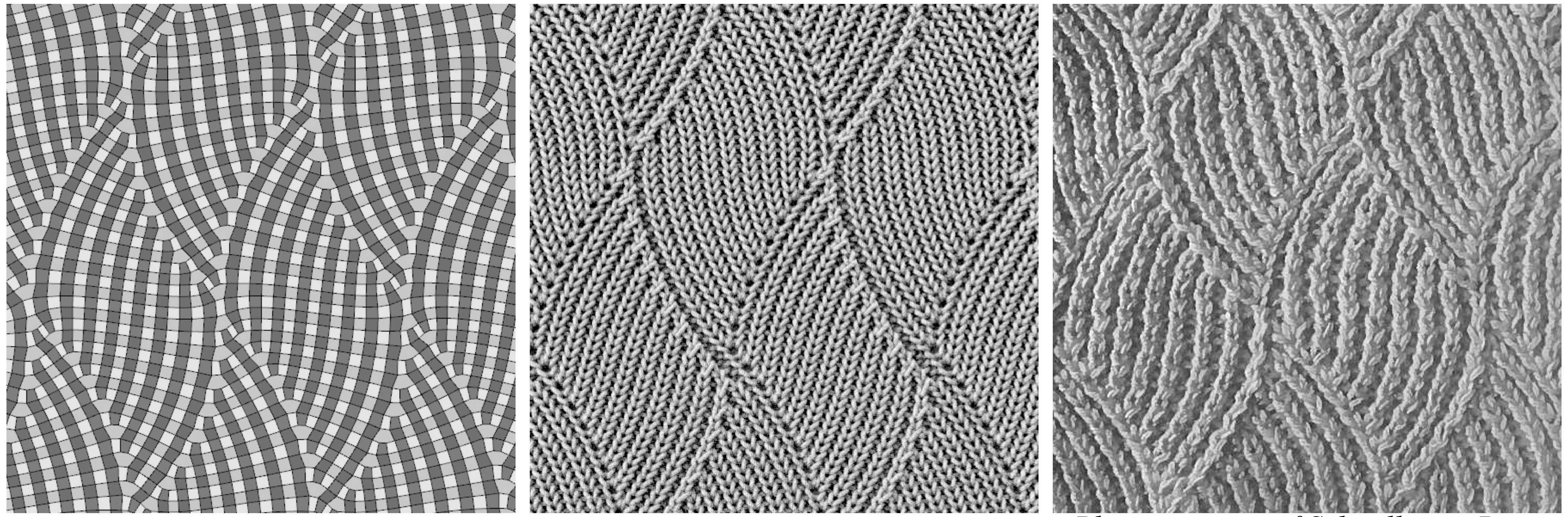
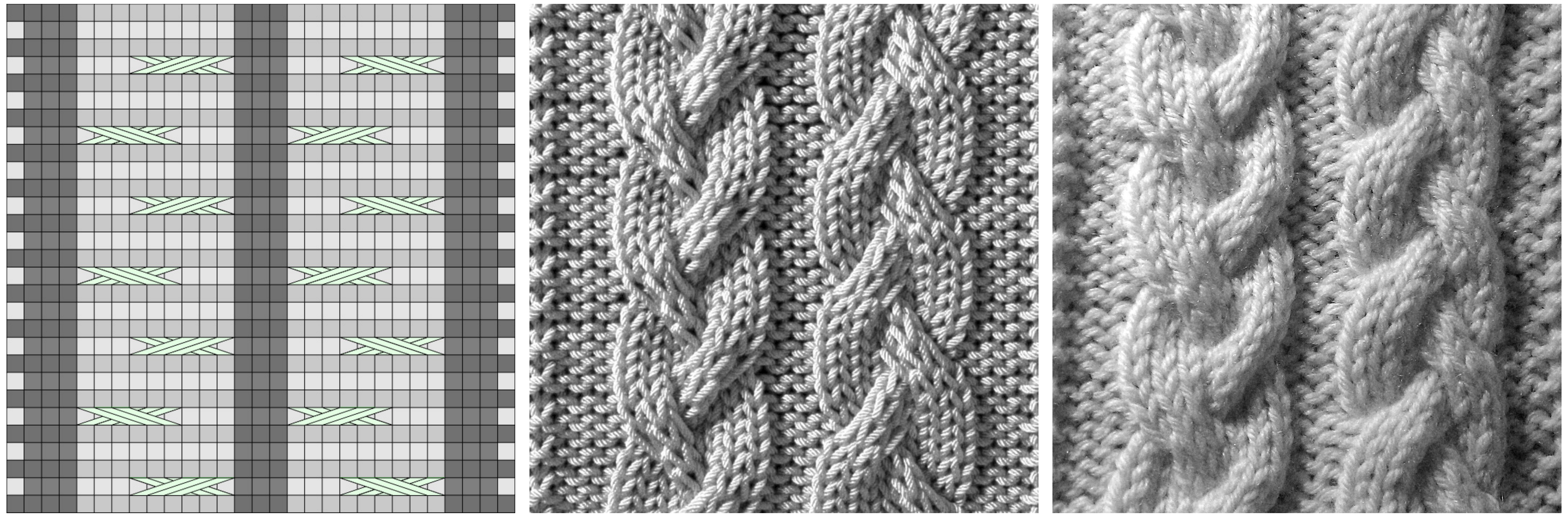


Photo courtesy of Schoolhouse Press

Flame Ribbing Pattern [Walker 2001]



Braid Cables Pattern [Allen et al. 2008]

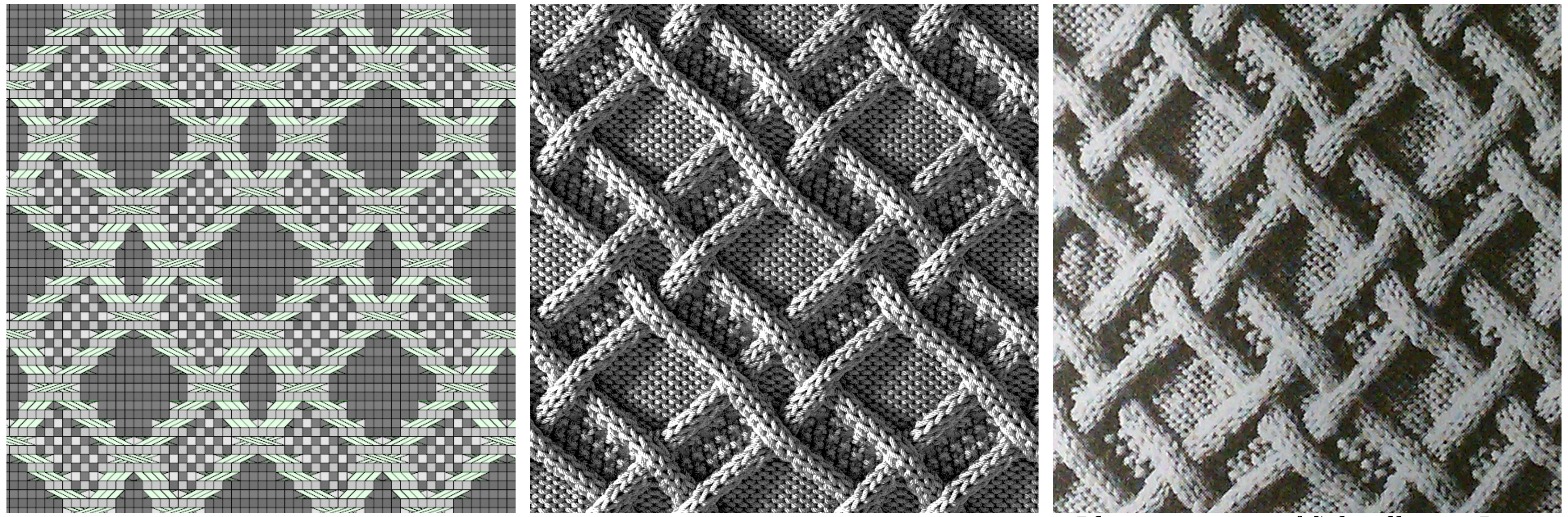


Photo courtesy of Schoolhouse Press

Cable Work Pattern [Walker 2001]

Taichi

Taichi Lang

Origins

- Dissertation work of Yuanming Hu at MIT
- Introduced in a series of SIGGRAPH papers in 2019–2021
- Now maintained (sporadically) by the company Taichi Graphics

What it is

- A language that looks a lot like Python
- A set of data structures for dense and sparse grids
- A just-in-time compiler targeting CPU and GPUs

What it does for us

- Lets us write simple simulation methods with simple code and without C++
- Generates parallel code without a lot of extra effort on our part

Taichi Newton fractal demo

Implemented as a loop in Python

- the Python interpreter has to execute Python code for every pixel
- it does this serially on one core, so it is pretty slow

Implemented as a vectorized NumPy program

- the Python interpreter just executes code with a few calls to Numpy matrix ops
- the Numpy kernels are in fast C code but they still run single-core

Implemented as a Taichi kernel

- the Taichi compiler generates parallel code that runs on many CPU or GPU cores
- the Python interpreter just makes one call
- in many cases it is a lot faster than the other two options