Lecture 1:

Introduction

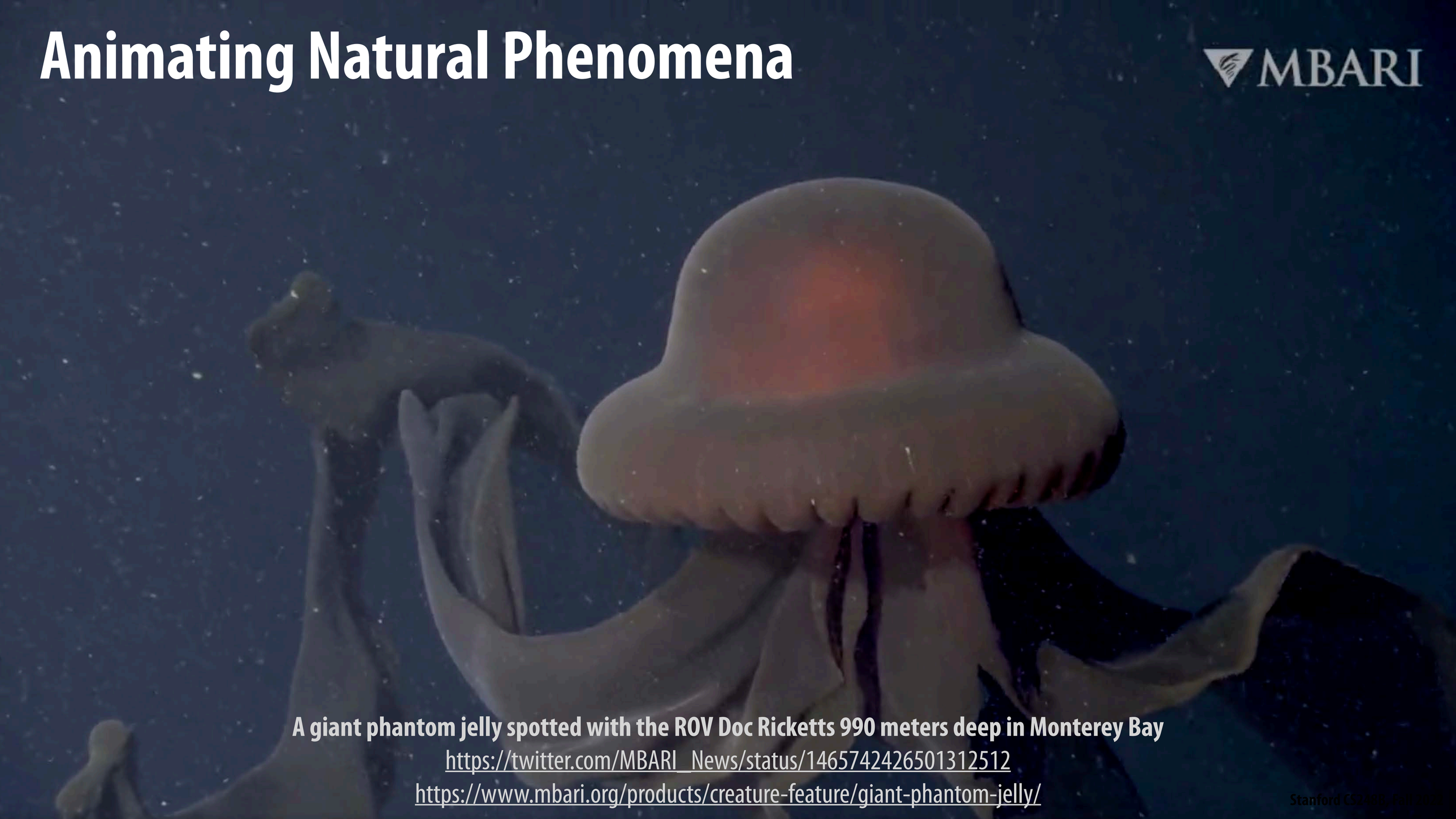
FUNDAMENTALS OF COMPUTER GRAPHICS

Animation & Simulation

Stanford CS248B, Fall 2022

PROFS. KAREN LIU & DOUG JAMES

Animating Natural Phenomena



A giant phantom jelly spotted with the ROV Doc Ricketts 990 meters deep in Monterey Bay

https://twitter.com/MBARI_News/status/1465742426501312512

https://www.mbari.org/products/creature-feature/giant-phantom-jelly/


MBARI

Course Logistics

About Doug

Affiliations:
- Stanford CS (ICME, CCRMA)
- NVIDIA Research (Simulation)

Expertise:
- Applied mathematics
- Computer graphics & animation
- Simulation & modeling
- Computational physics
- Acoustics
- Fabrication

About Doug

Doug L. James is a Full Professor of Computer Science at Stanford University (since June 2015), and a member of Stanford's Center for Computer Research in Music and Acoustics (CCRMA) and the Institute for Computational and Mathematical Engineering (ICME). He holds three degrees in applied mathematics, including a Ph.D. in 2001 from the University of British Columbia. In 2002 he joined the School of Computer Science at Carnegie Mellon University as an Assistant Professor, and later became an Associate Professor of Computer Science at Cornell University (2006-2015). His research interests include computer graphics, computer sound, physically based modeling and animation, and reduced-order physics models. Doug is a recipient of a National Science Foundation CAREER award, and a fellow of both the Alfred P. Sloan Foundation and the Guggenheim Foundation. He received the ACM SIGGRAPH 2021 Computer Graphics Achievement Award, a 2012 Technical Achievement Award from The Academy of Motion Picture Arts and Sciences for "Wavelet Turbulence," and the 2013 Katayanagi Emerging Leadership Prize from Carnegie Mellon University and Tokyo University of Technology. He was the Technical Papers Program Chair of ACM SIGGRAPH 2015, and a consulting Senior Research Scientist at Pixar Animation Studios from 2015-2020. Since 2022 he has been a consulting Senior Research Scientist at NVIDIA.

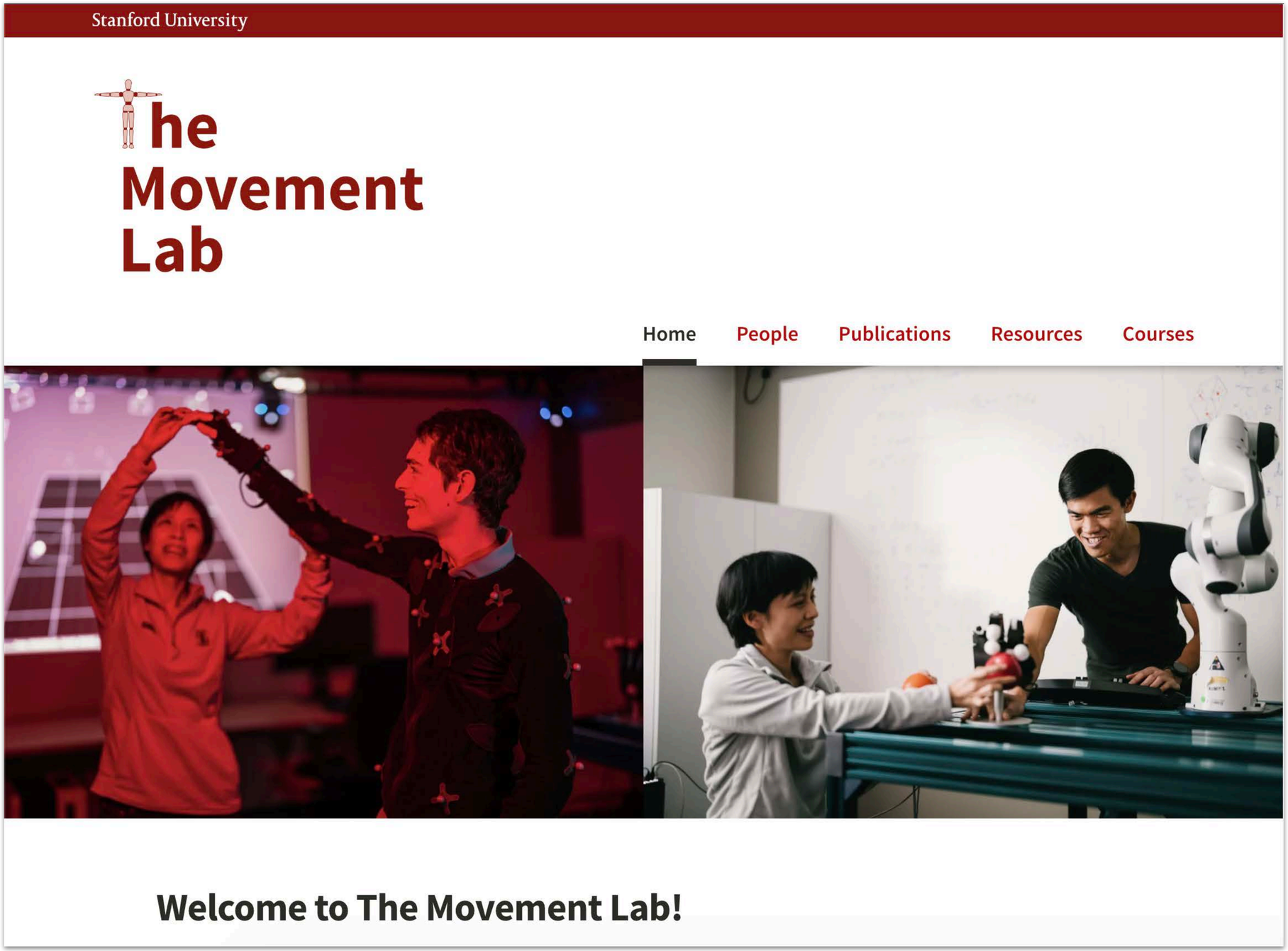
Doug L. James
Professor of Computer Science and, by Courtesy, of Music

Stanford Profile

About Karen

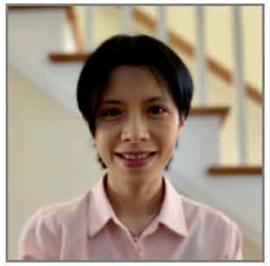
Affiliations:
- Stanford CS (HAI, ...)

Expertise:
- Character animation and robotics
- Computer graphics & animation
- Simulation & modeling
- Optimal control
- Reinforcement learning
- Computational biomechanics



Course Staff

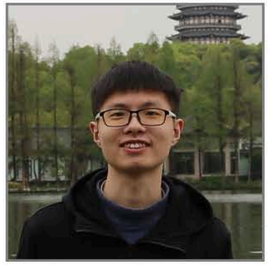
Doug James
[djames at stanford]
Office hours:



Karen Liu
[ckliu38 at stanford]
Office hours:

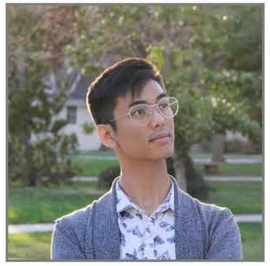
Your fun and helpful TAs:

Avi Goyal
[sagoyal at stanford]
Office hours:



Peng Chen
[pengc at stanford]
Office hours:

Yezhen Cong
[yzcong at stanford]
Office hours:



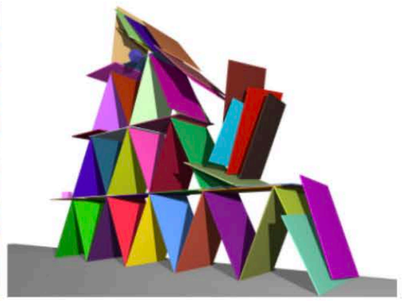
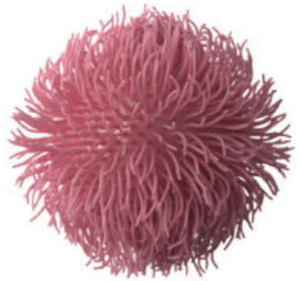
Takara
[takaraet at stanford]
Office hours:

Webpage

[Home] [Course Info] [Lectures/Readings] Welcome, **Doug**  [Logout]

Stanford CS248B, Fall 2022

Fundamentals of Computer Graphics:
Animation and Simulation


This course provides a comprehensive introduction to computer graphics, focusing on fundamental concepts and techniques in Computer Animation and Physics Simulation. Topics include numerical integration, 3D character modeling, keyframe animation, skinning/rigging, inverse kinematics, rigid body dynamics, deformable body simulation, and fluid simulation.

Basic Info

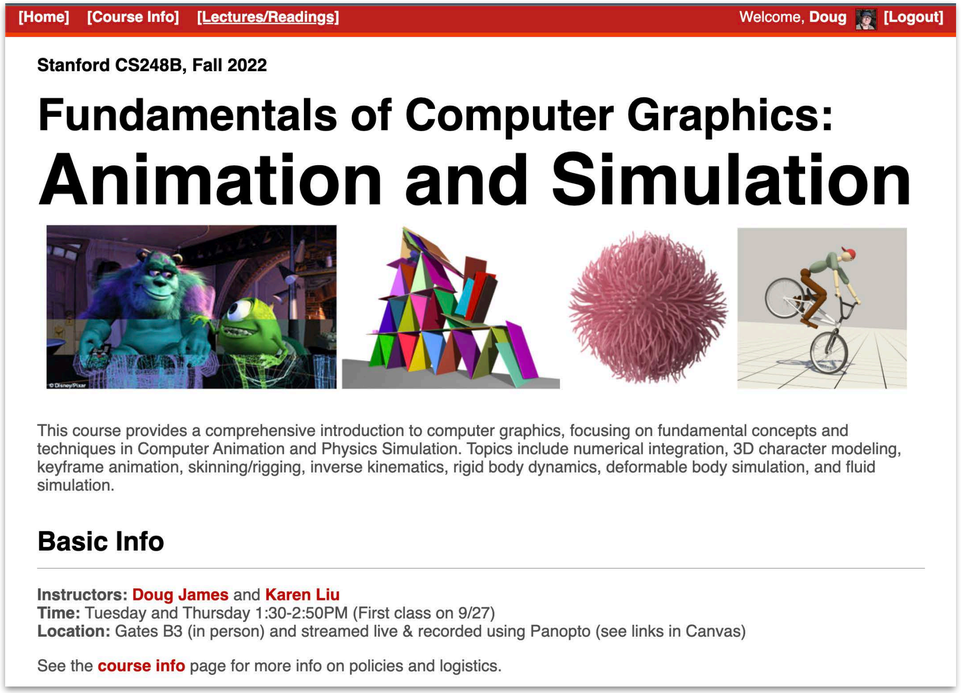
Instructors: **Doug James** and **Karen Liu**
Time: Tuesday and Thursday 1:30-2:50PM (First class on 9/27)
Location: Gates B3 (in person) and streamed live & recorded using Panopto (see links in Canvas)

See the **course info** page for more info on policies and logistics.

https://web.stanford.edu/class/cs248b/cgi-bin/autumn22/index.php/home

Webpage

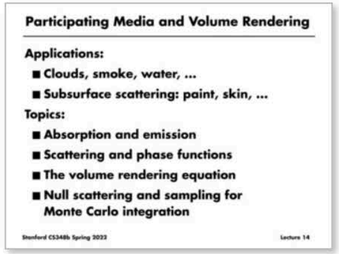
- Main place to get course content, e.g., slides
- Students + staff can login to the webpage
  - TODO: Create an account
  - CODEWORD: (see Ed)
- Comment feature:
  - Login to post + read slide comments
  - A great way to ask and answer slide-specific questions
  - Example from CS348B

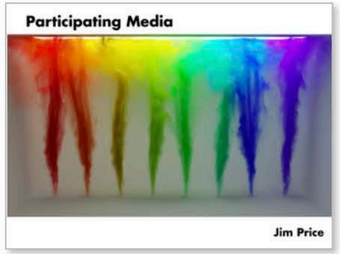


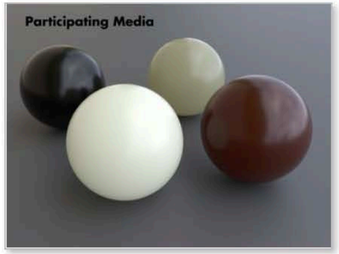
Comment Example: CS348B-S22 Image Synthesis

Lecture 14: Participating Media and Volume Rendering

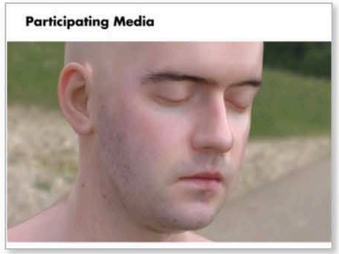
Download slides as PDF



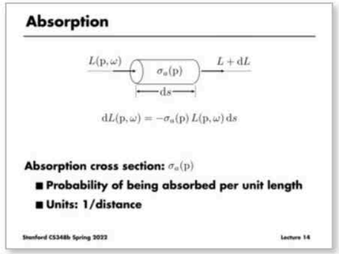




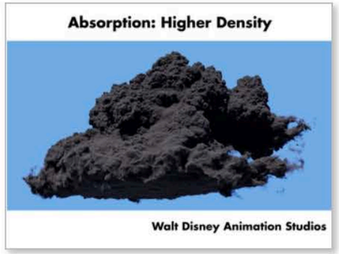
1 comment

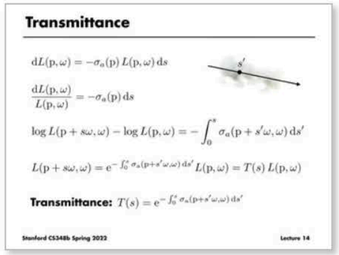


4 comments

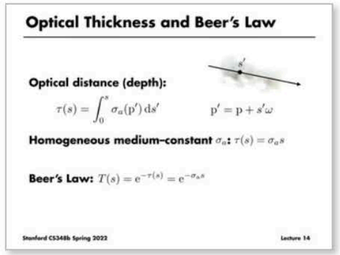


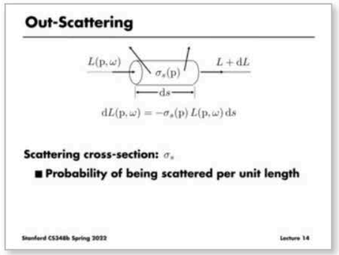
4 comments

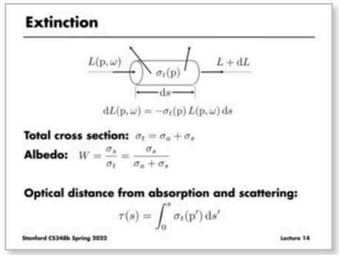




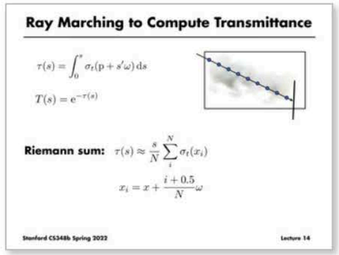
3 comments

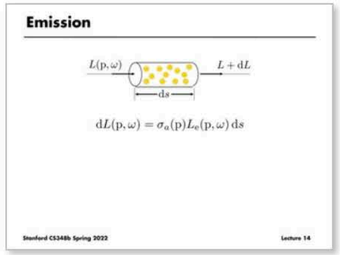




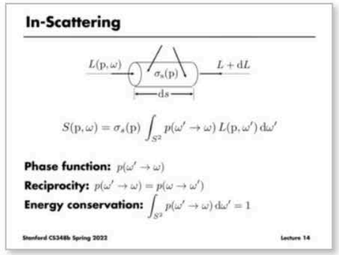


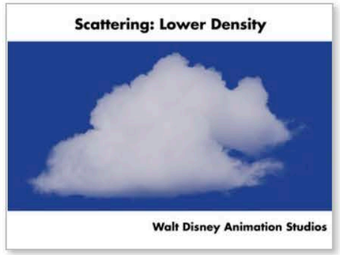
2 comments

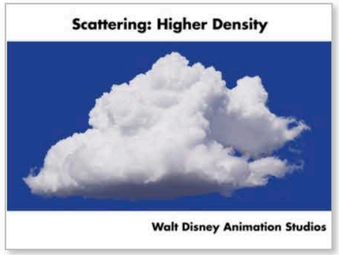


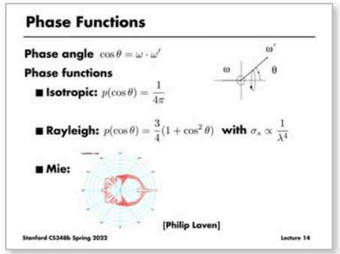


1 comment

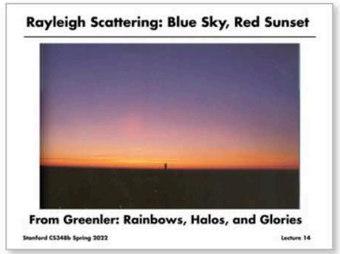




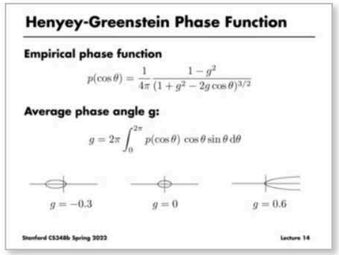




3 comments



6 comments



1 comment

Comment Example: CS348B-S22 Image Synthesis

[Home] [Feed] [Course Info] [Lectures/Readings] [Admin Console]

Participating Media and Volume Rendering

Absorption: Lower Density

Walt Disney Animation Studios

Previous | Next --- Slide 6 of 47 Back to Lecture Thumbnails

syhunter How is the shape of the cloud generated? A noise function?

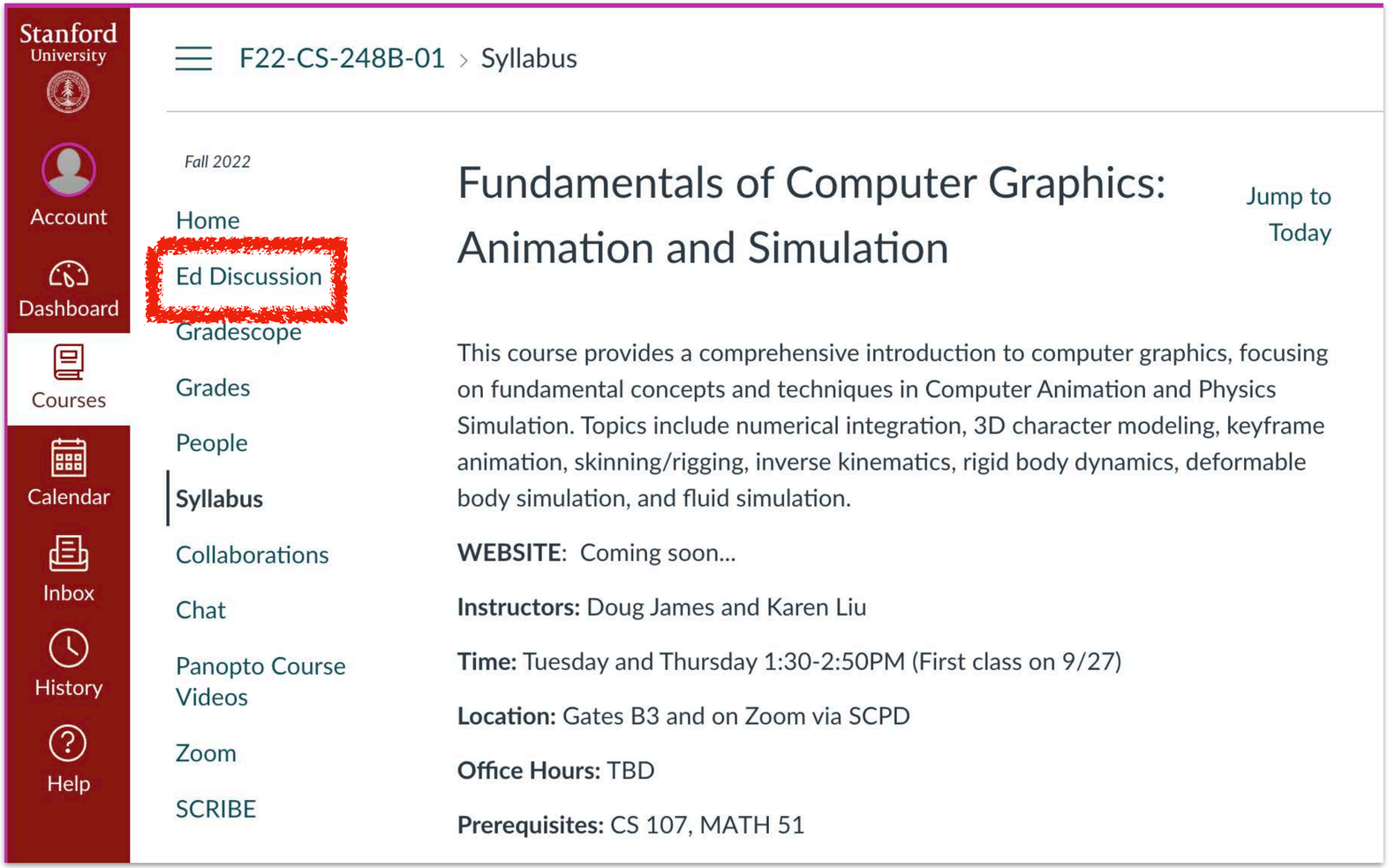
djames It could be procedural noise with artistic weighting. FYI, the dataset webpage is here and does not say:

https://www.disneyanimation.com/resources/clouds/

A google search led me to this survey of atmospheric cloud generation:

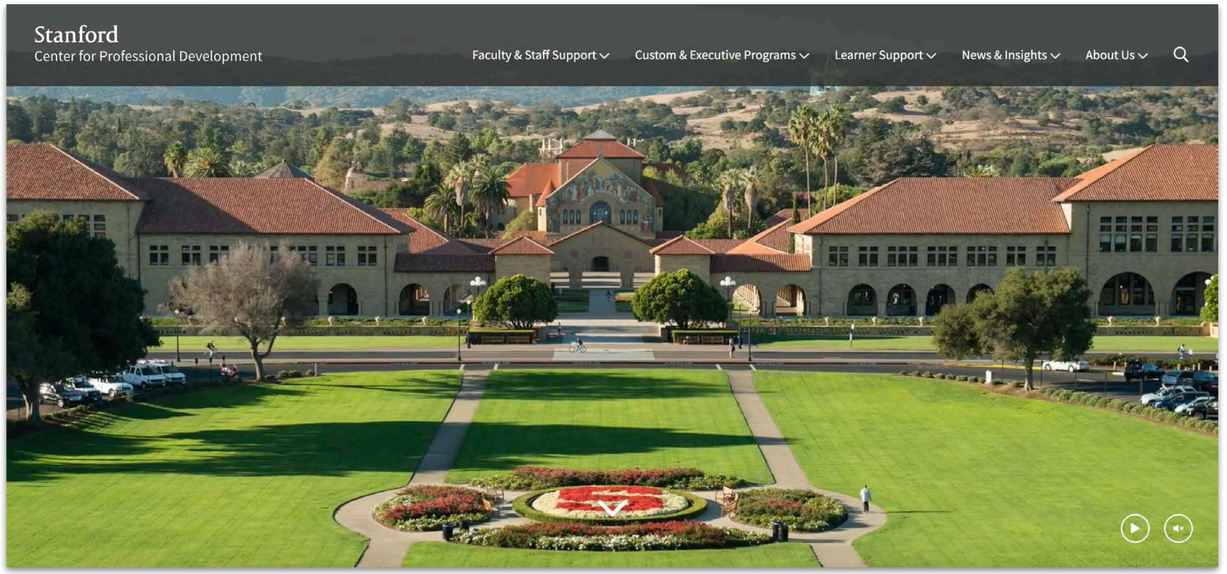
Ed

- Online discussion forum
- Access link through Canvas
- Main place to ask questions



SCPD Students

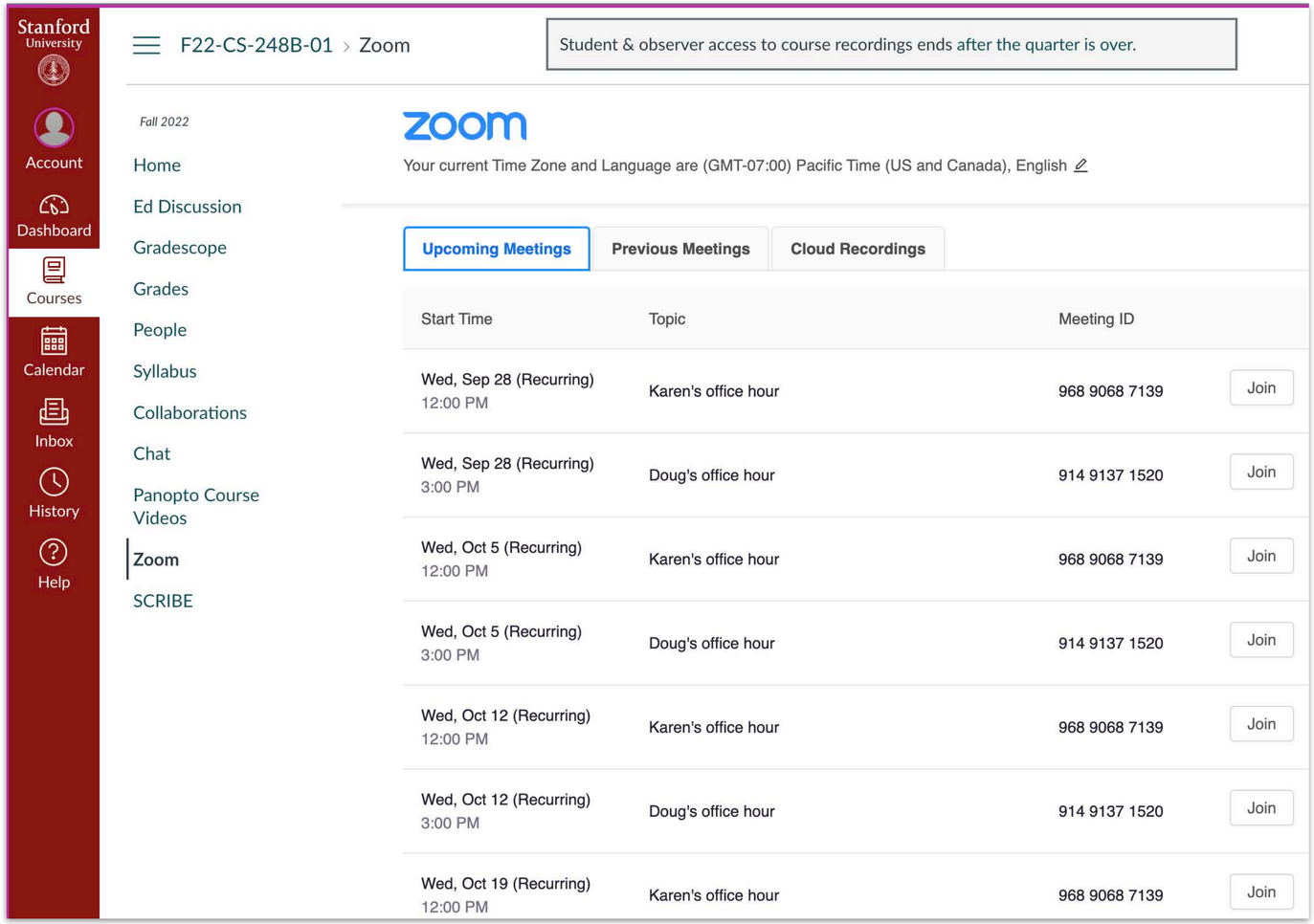
Stanford Center for Professional Development (SCPD)



- This is a hybrid SCPD class
  - Both in-person and remote attendees.
- All lectures are recorded and available to all students
  - Available in Canvas (see "Panopto Course Videos")
- All lectures are live streamed with live chat
  - Please feel free to ask questions
  - Course staff will answer them

Office Hours

- Staff OH will be available over Zoom
  - Convenient and safe
  - Zoom links can be found in Canvas
- Some staff OHs will be hybrid
- Updated office hours on the webpage
- Can ask me questions after class



COVID-19 Considerations

- Face coverings are required in the classroom at Stanford University until further notice
- We care about your health & safety
  - We want everyone to complete their studies safely
  - We want everyone to graduate and have productive careers
- We will provide opportunities for you to reduce your exposure and stay safe
- It is OK to skip class if you do not feel well, or have been exposed, etc.
  - You do not need a note
- Please take advantage of our numerous accommodations:
  - Lecture recordings
  - Lecture live stream + chat
  - Slide and Ed discussions
  - Virtual office hours
- If you have any concerns, please contact us

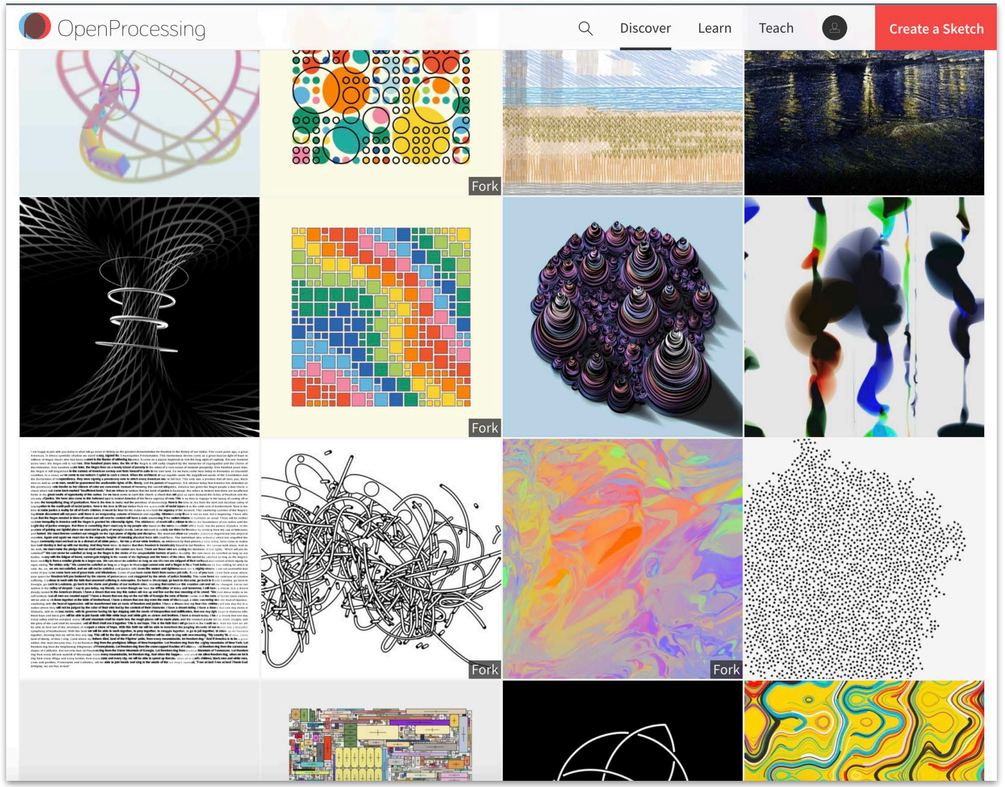
Tentative Syllabus

Lectures + Projects

- 20 Lectures
- 5 Written Homeworks
  - HW1: Math foundations
  - HW2: Numerical integration
  - HW3: Rigid and deformable bodies
  - HW4: Keyframe animation
  - HW5: Articulated bodies
- 4 Programming Assignments
  - The fun part :)
  - Details in a moment
- Final Exam
  - Comprehensive

Week 1: Introduction

- Math Foundations
  - Recap of mathematical and geometrical concepts
  - Leads to first written homework (HW1)
- Programming Warm-up
  - p5.js is a JavaScript library for creative coding
    - https://p5js.org
  - OpenProcessing (OP)
    - https://openprocessing.org
    - Everyone in the course will get an OP course account
    - Enables easy group collaboration and sharing of your results
      - With or without sharing code
      - So… we can try each other's simulations and … play each other's games



Week 2: Particle Systems

- **Particle Systems**
  - Dynamical systems
  - First- vs second-order motion models
  - Forces
  - Numerical integration schemes
- **Collision Resolution Basics**
  - Collision examples from animation (hair, cloth, rigid bodies, etc.)
  - Contact models; Position- vs velocity-level contact resolution
  - Contact conditions
  - Coefficient of restitution
  - Planar and nonplanar contact geometry
  - Signed distance fields (SDFs)

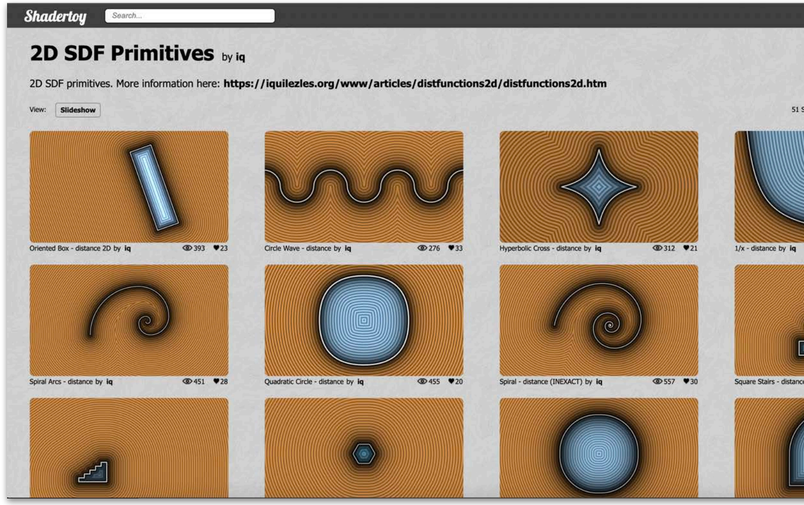
[Reeves 1983]

Lagoa

Nomination for best supporting actor:

Signed distance fields (SDFs)

- Useful primitive for proximity queries (used in rendering, simulation, etc.)
- Definition: Let d(x) be the signed distance to a closed boundary/surface, where
  - d(x)<0 : x is inside the shape
  - d(x)=0 : x is on the shape
  - d(x)>0 : x is outside the shape
  - |d(x)| is the closest distance to the shape's boundary from x
- Analytical SDFs are know for many simple geometric shapes
  - 2D SDFs: https://iquilezles.org/articles/distfunctions2d
- Efficient evaluation for simulation and HW-accel. rendering
  - https://www.shadertoy.com/playlist/MXdSRf



Programming Assignment #1: Simulating Pinball!

- Modify starter code to design your own virtual pinball game
- Robust collision processing using 2D SDFs
  - Ray-march SDFs for robust continuous collision checks
- Work alone or with a partner. Full creative control.
- Play each other's games on OpenProcessing (w/o sharing code)

Groovy Starter Code

Animating Deformable Models: Hair

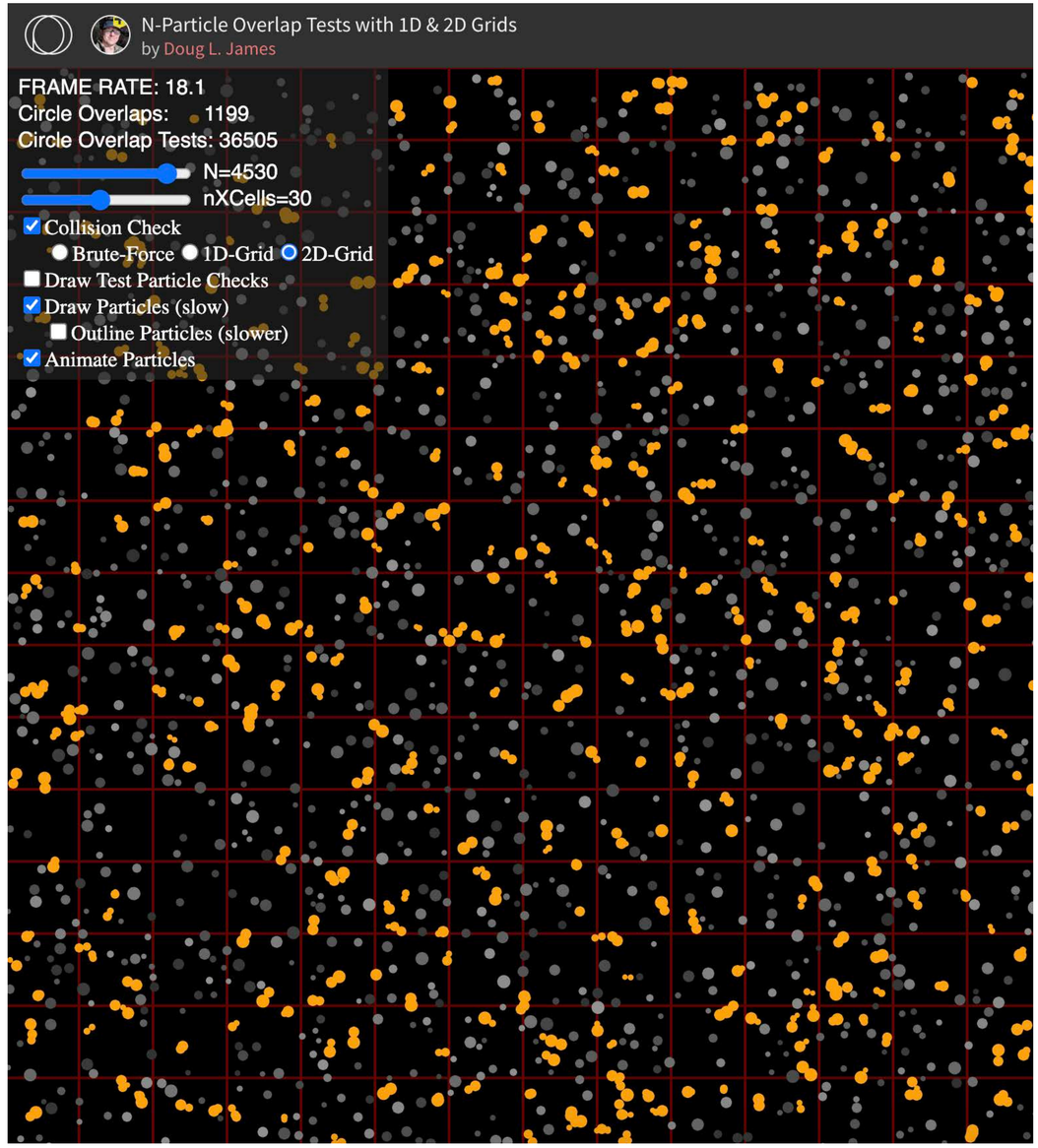
Tangled [Disney]

Brave [Pixar]

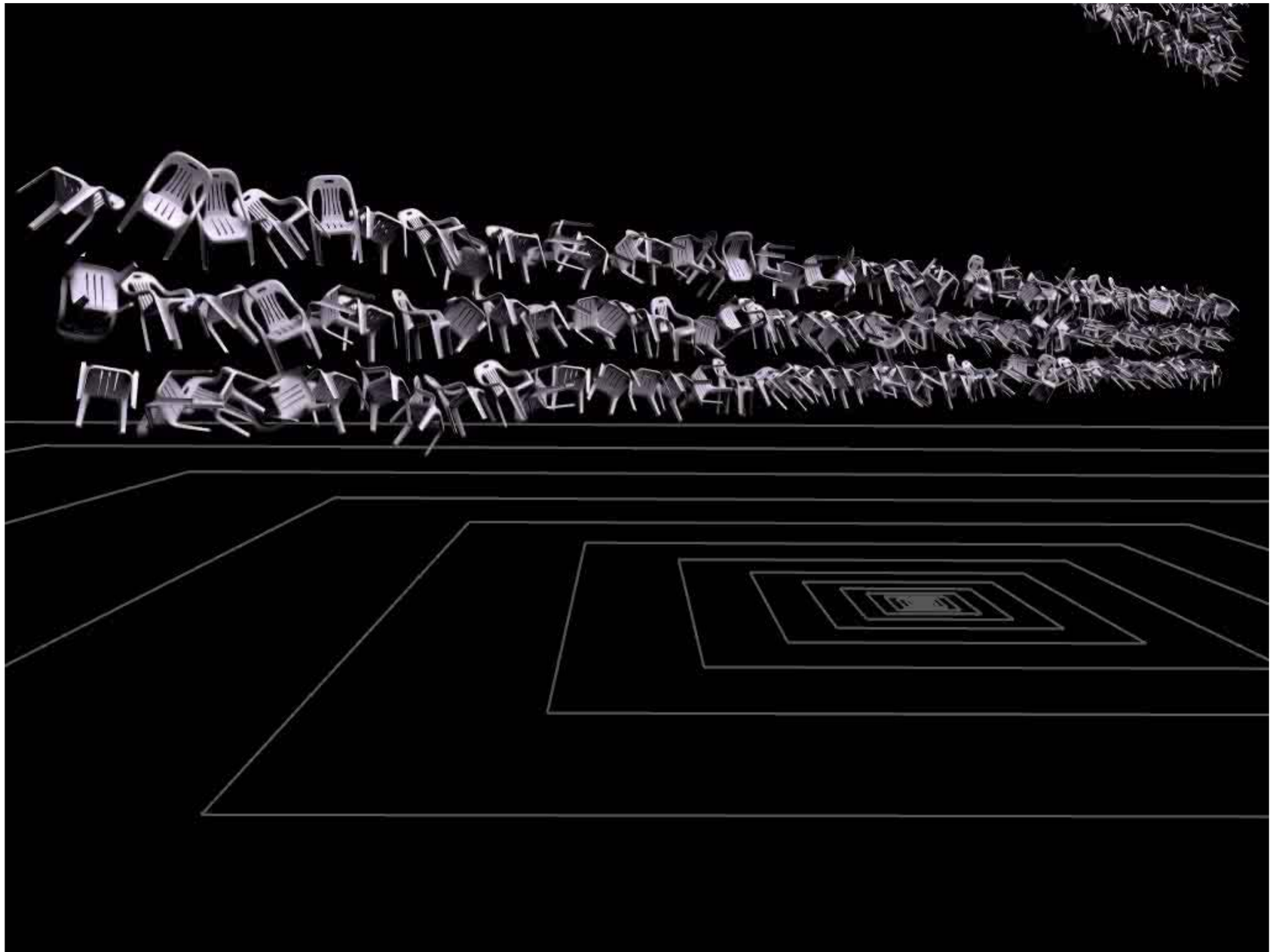
Adaptive Nonlinearity for Collisions in Complex Rod Assemblies [Kaufman et al. 2014]

Week 3: Collisions

- Collision Detection (CD) Methods
  - Broad vs narrow phase CD
  - All-pairs culling strategies
  - Primitive-level computations
  - Bounding volumes (fitting and evaluation)
  - Broad phase CD data structures
- Continuous collision detection methods
  - Discrete vs Continuous CD
  - CCD methods (2D, 3D)
  - Root finding schemes
    - Ray-marching SDFs (for pinball, of course)
  - Collision resolution strategies (conservative advancement, ...)

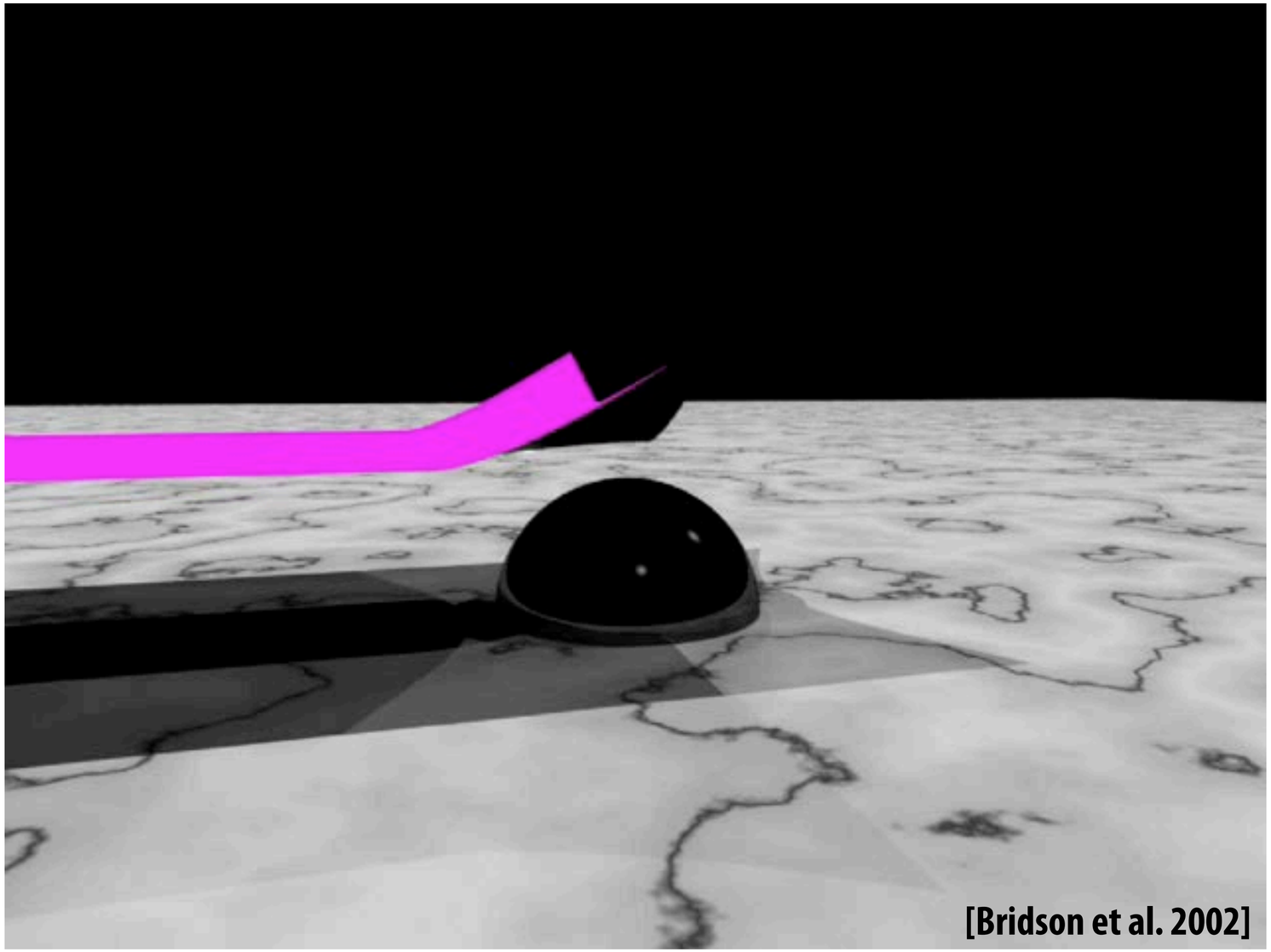


Animating Deformable Models: Plastic Chairs

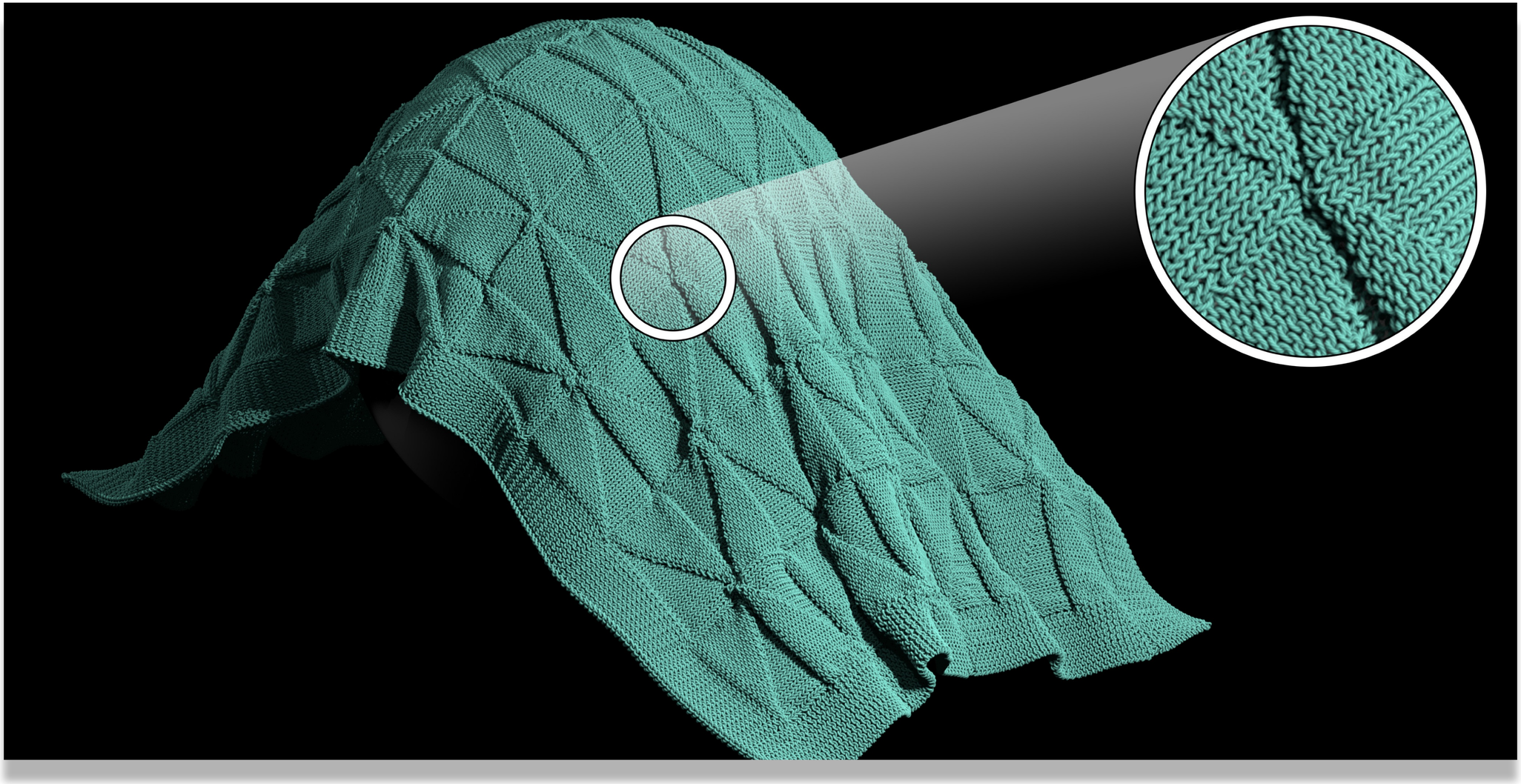


[James and Pai 2004]

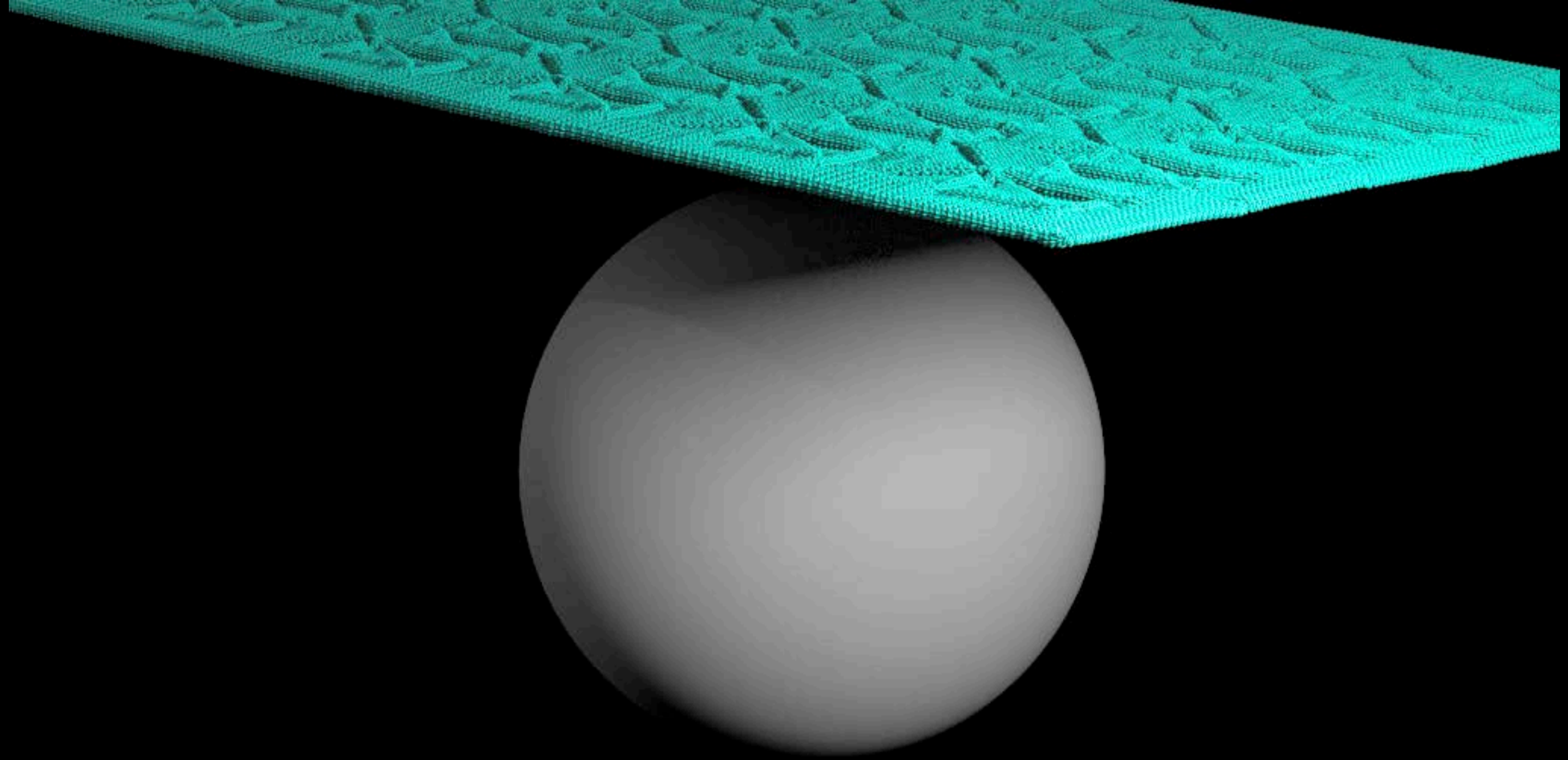
Animating Deformable Models: Sheet-based Cloth



Animating Deformable Models: Yarn-level Cloth



[Kaldor et al. 2010]

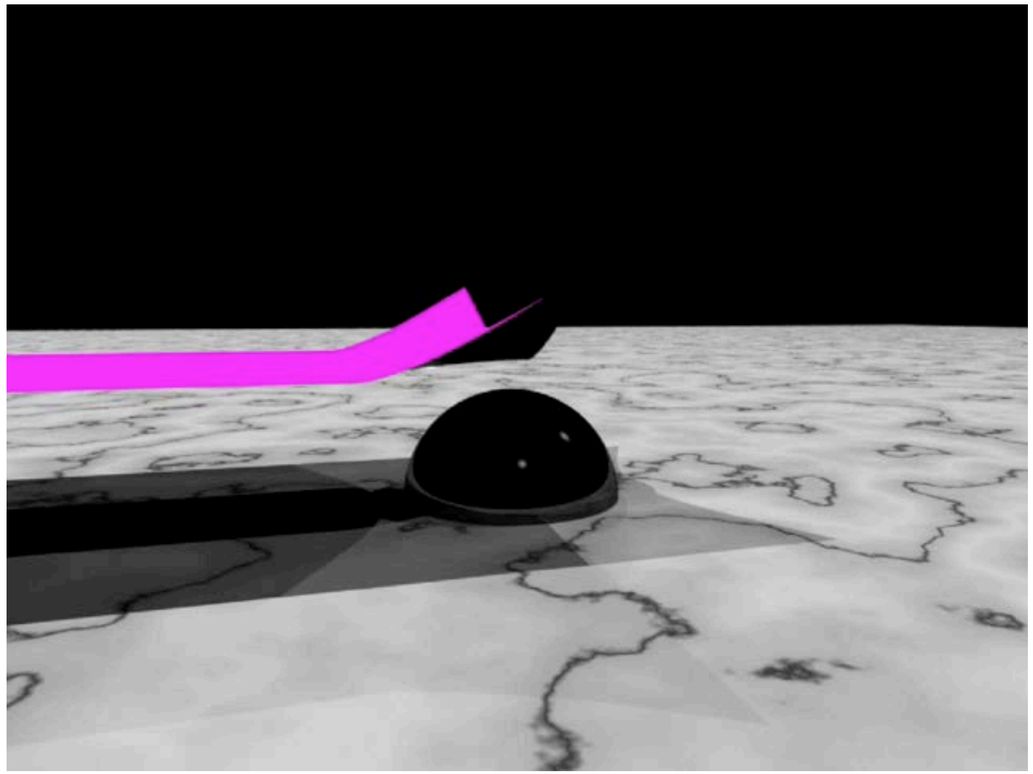


[Kaldor et al. 2010]

[Zheng & James 2012]

Week 4: Deformable Models

- Deformable Models in Animation
  - Examples from animation (hair, cloth, flesh, etc.)
  - Kinematic models
  - Constitutive force models
  - Numerical integration (stability and robustness)
  - Element inversion
- Robust Deformable Collision Processing
  - Detection vs resolution
  - Examples from cloth and hair simulation
  - Collision resolution schemes (impulses, penalty, etc.)
  - Untangling methods

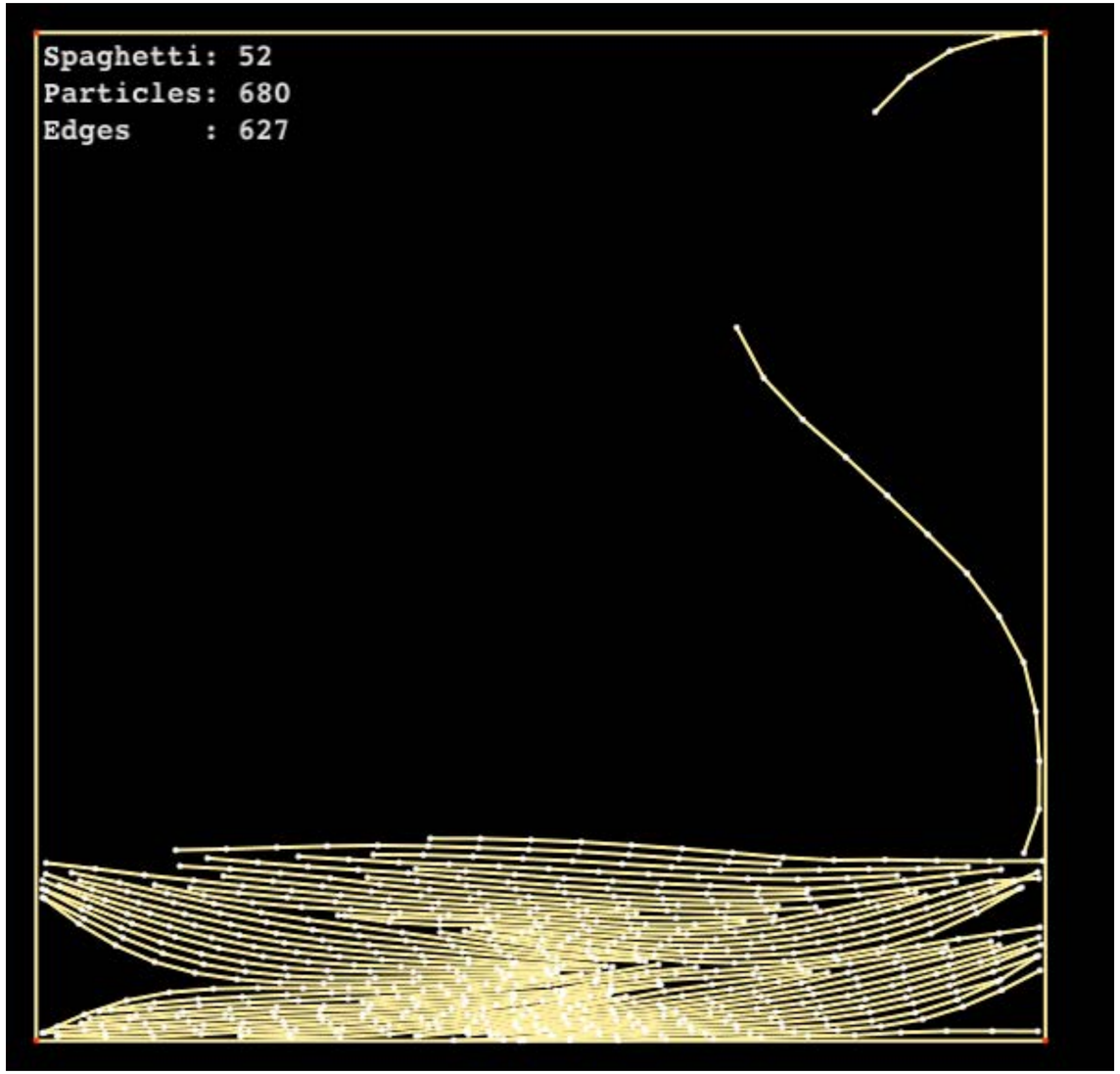


[Bridson et al. 2002]

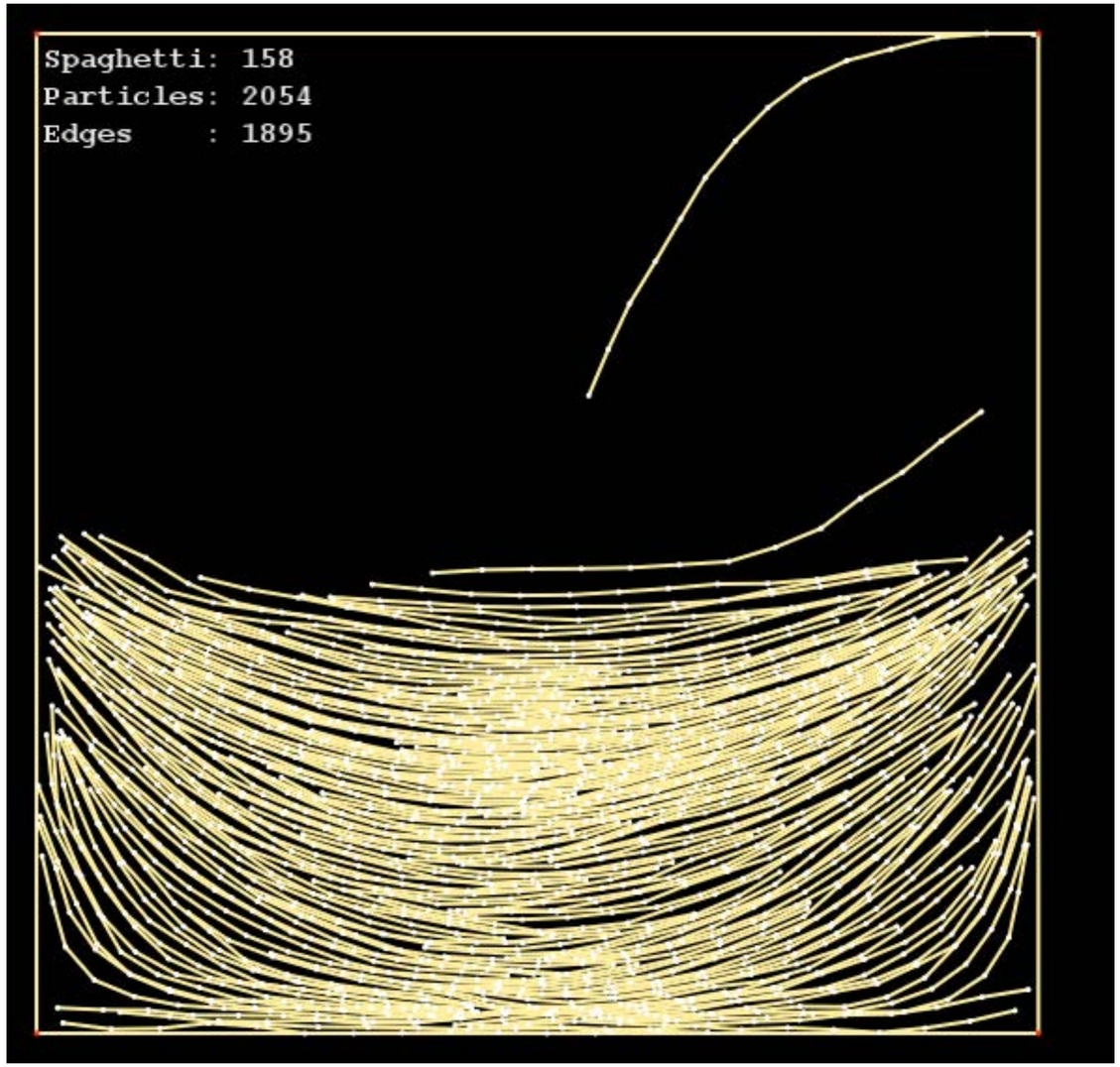
OLD Animation Homework: Robust Collision Processing

- Code a 2D particle-based collision processor
- Make it robust to interpenetrations … *at all cost!*

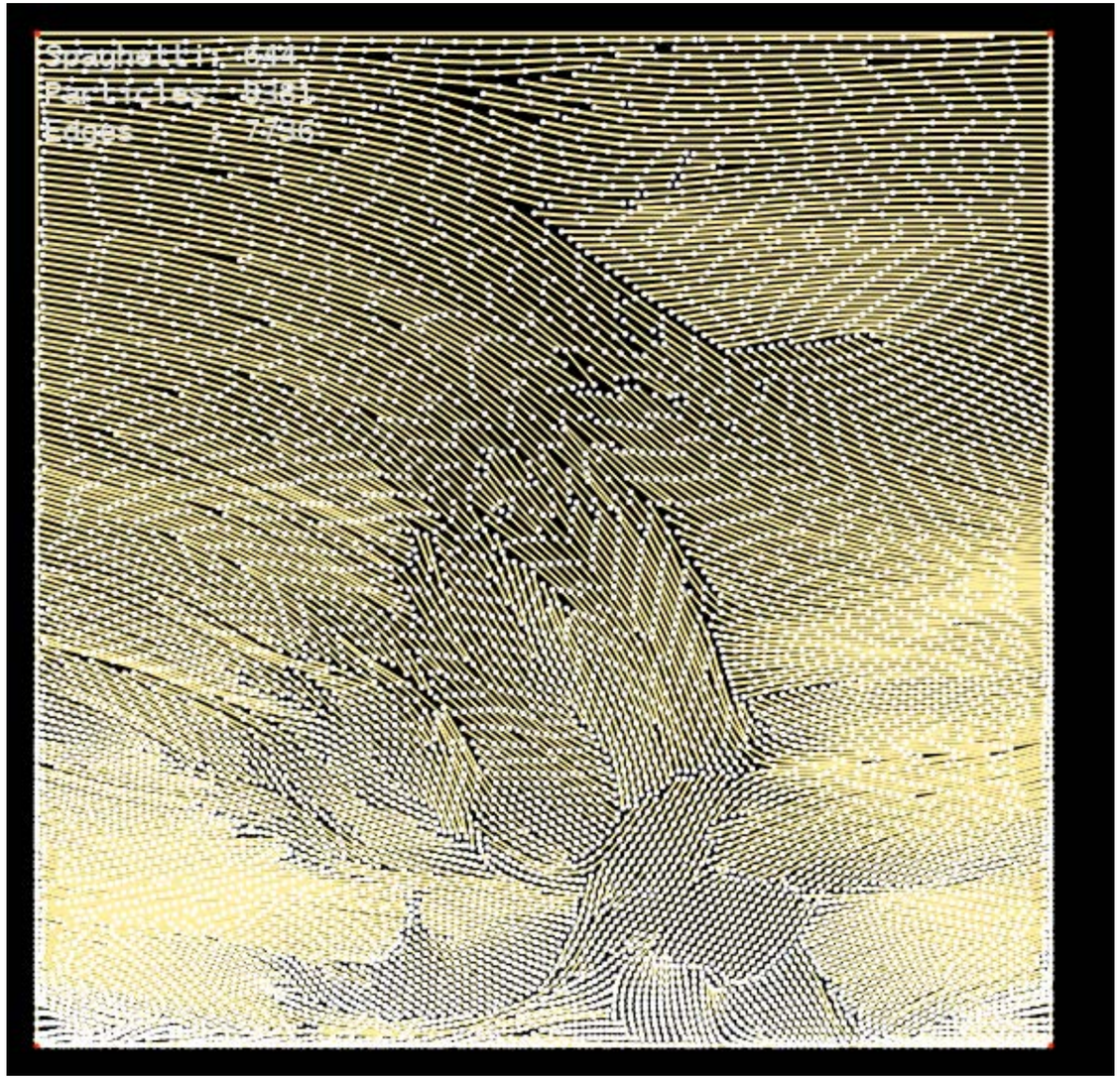
OLD Animation Homework: Robust Collision Processing



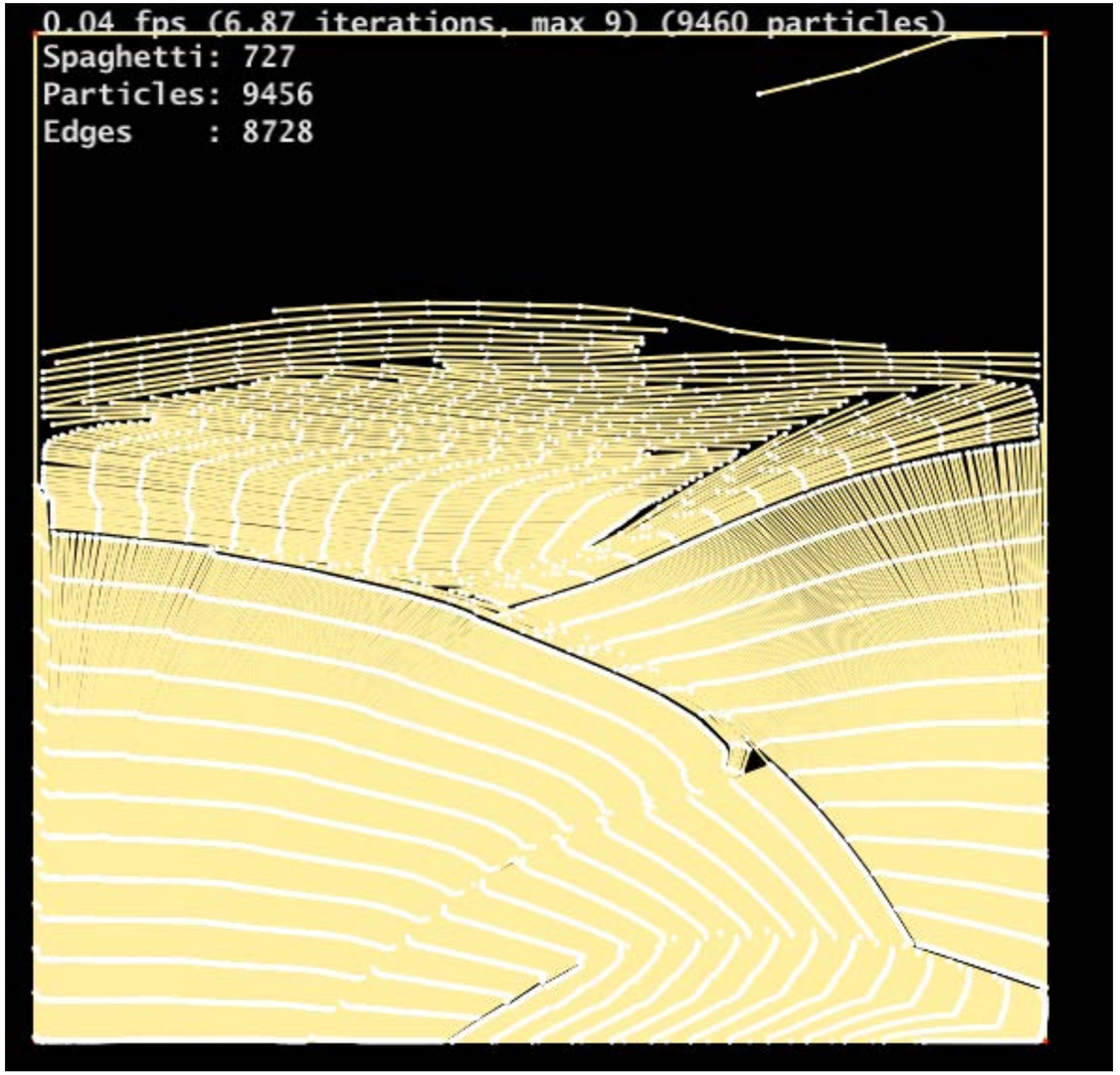
OLD Animation Homework: Robust Collision Processing



OLD Animation Homework: Robust Collision Processing



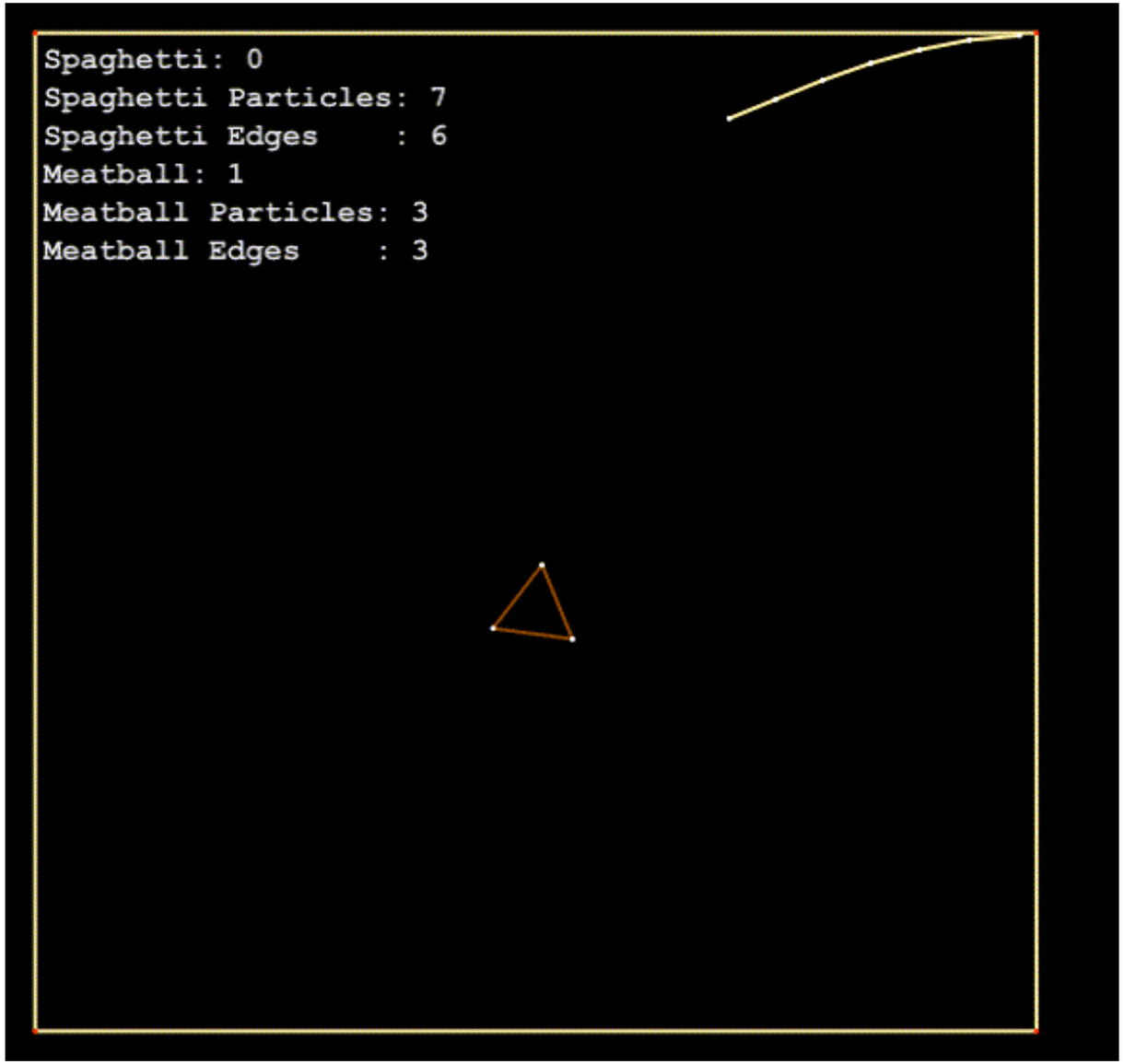
OLD Animation Homework: Robust Collision Processing

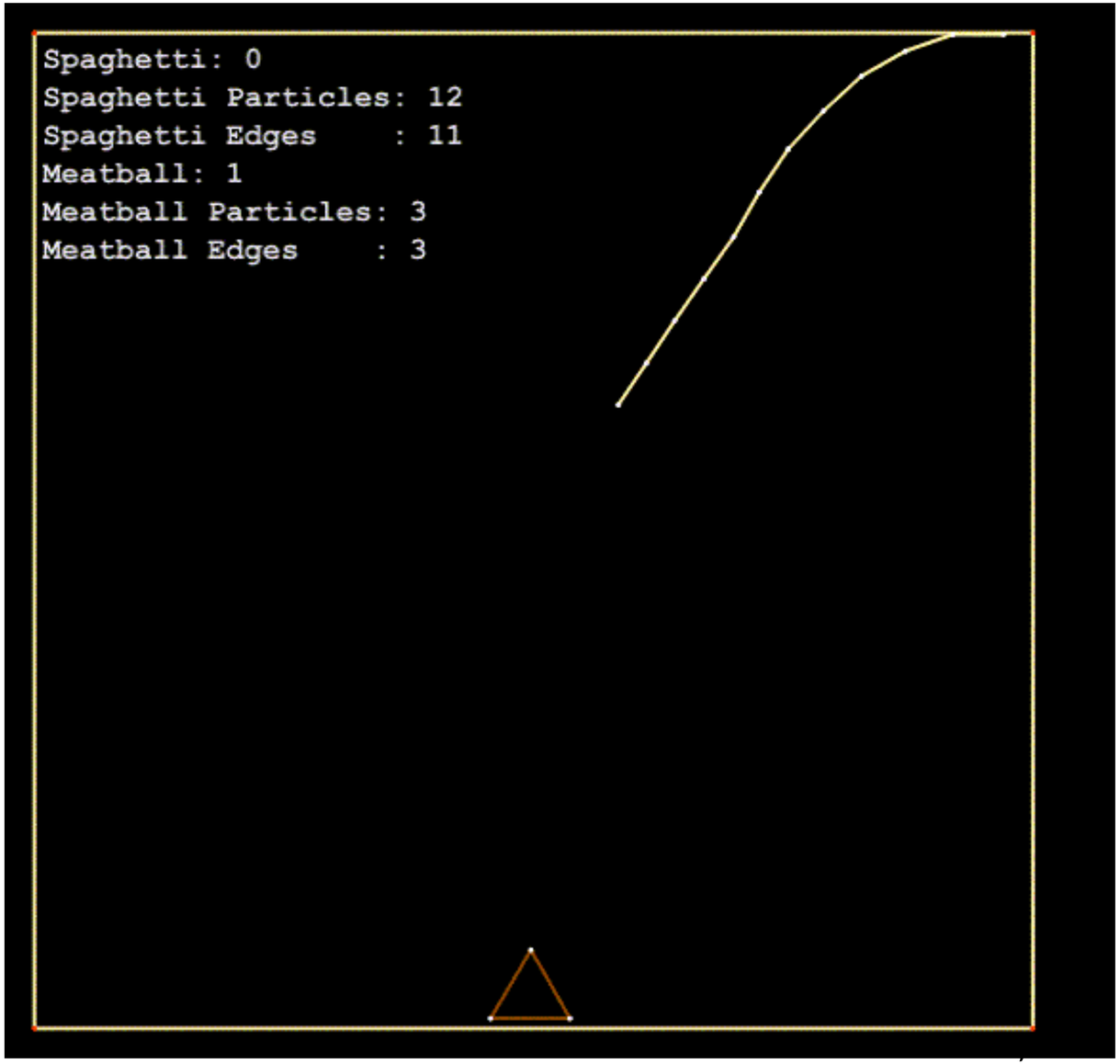


Wait… real spaghetti has…

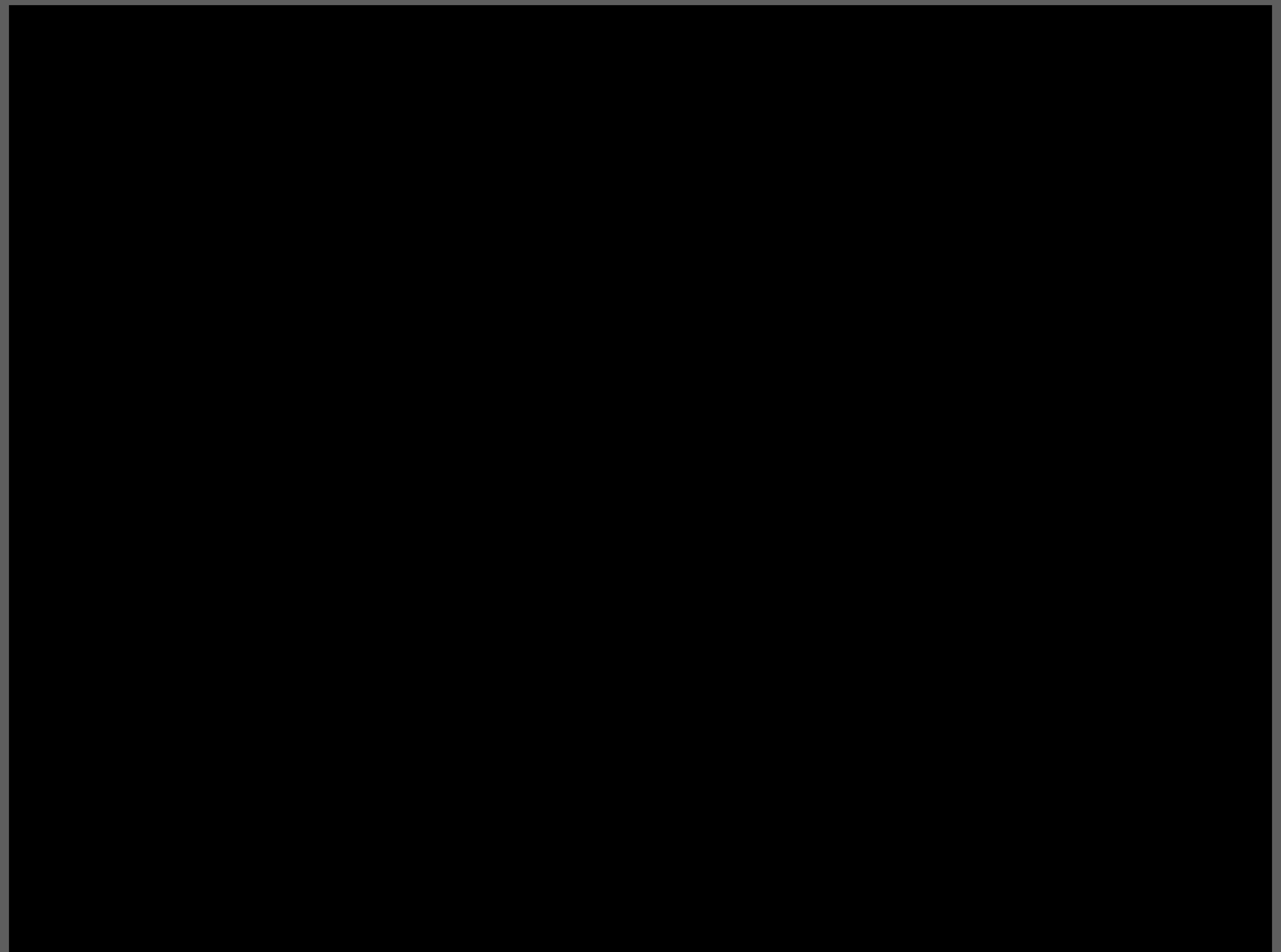
More spaghetti

By Paul Liu (TA)





CS348C-W19 HW3: Spaghetti Factory Results

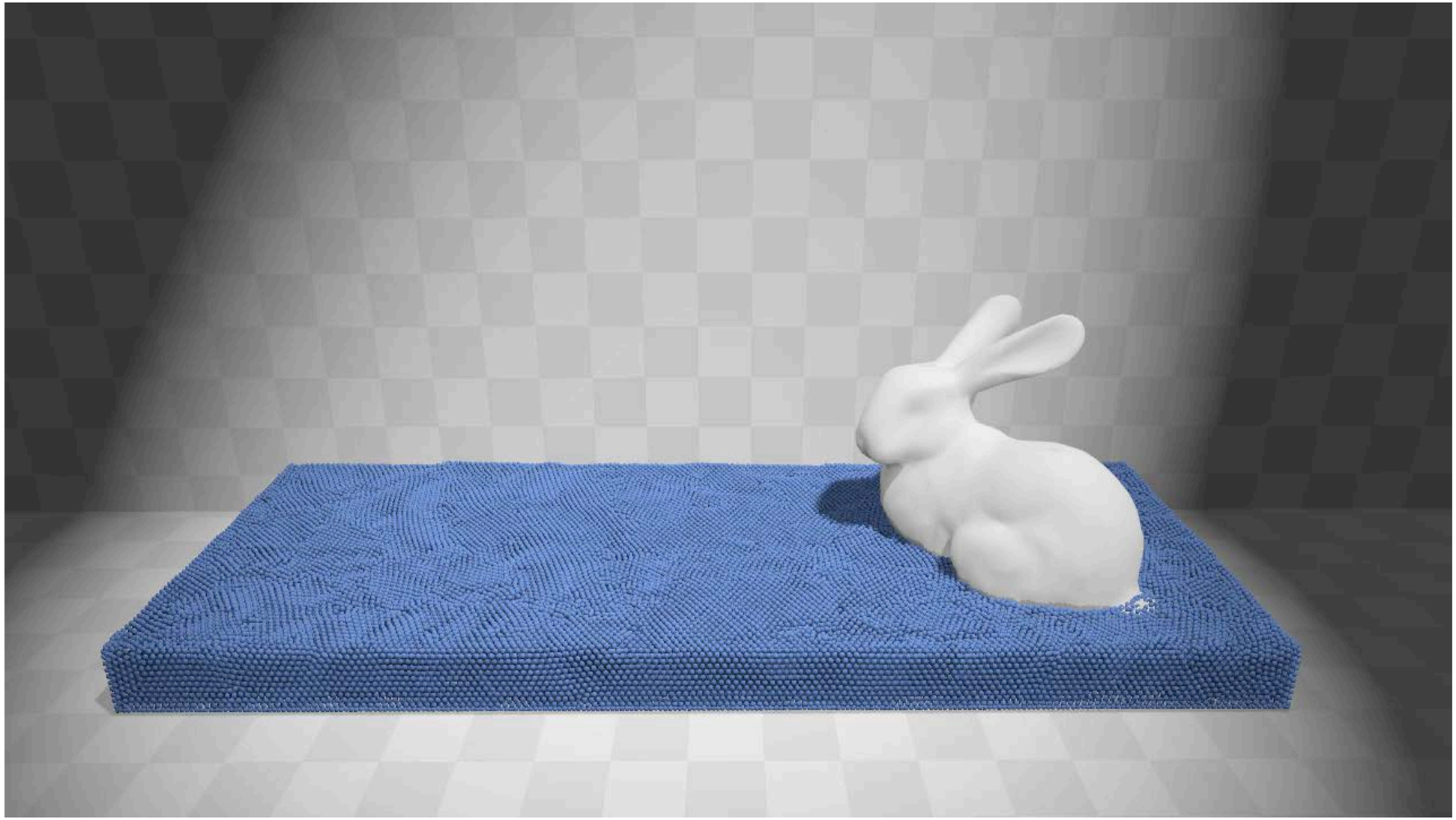


Paul Liu (TA)

Programming Assignment #2: "Blob Factory"

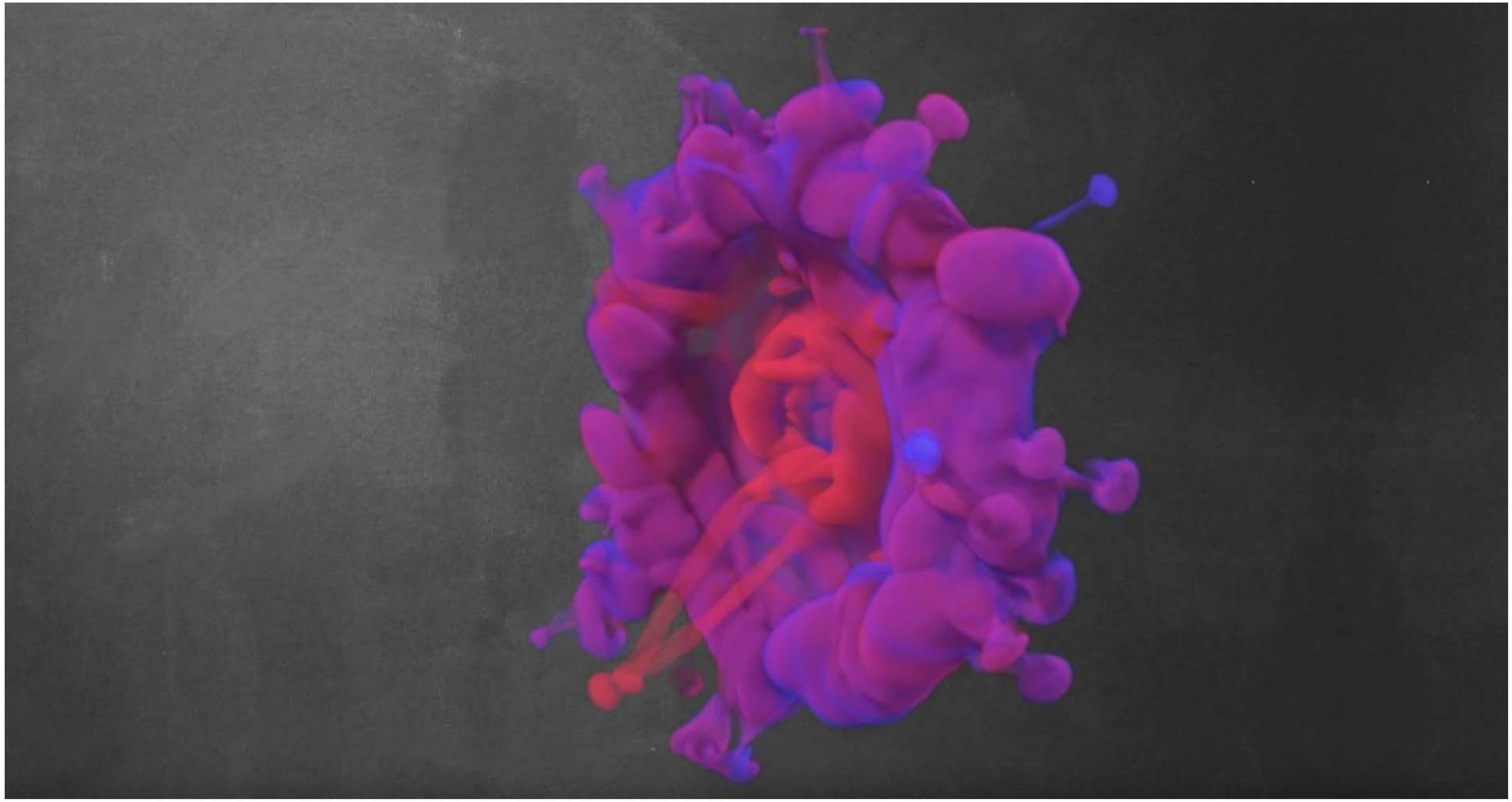
- Robust collision processing with 2D deformable models
  - Builds on P1's 2D SDF collision environment to now simulate 2D deformable bodies
- Components:
  - Implicit integration of 2D deformable models ("blobs")
  - Robust collision handling (both defo-rigid and defo-defo collisions)
- Theme: *Blob Factory or Blob Game* (you decide)
  - Mechanically simulate *non-interpenetrating blobs* using moving SDFs
  - Choose your favorite blobs (gummy bears, marshmallows, cells, balloons, etc.)
  - Machinery or game of your choosing

Week 5: Particle-based Fluids



Position Based Fluids [Macklin & Müller 2013]

Week 5: Grid-based Fluids



Schrödinger's Smoke [Chern et al. 2016]

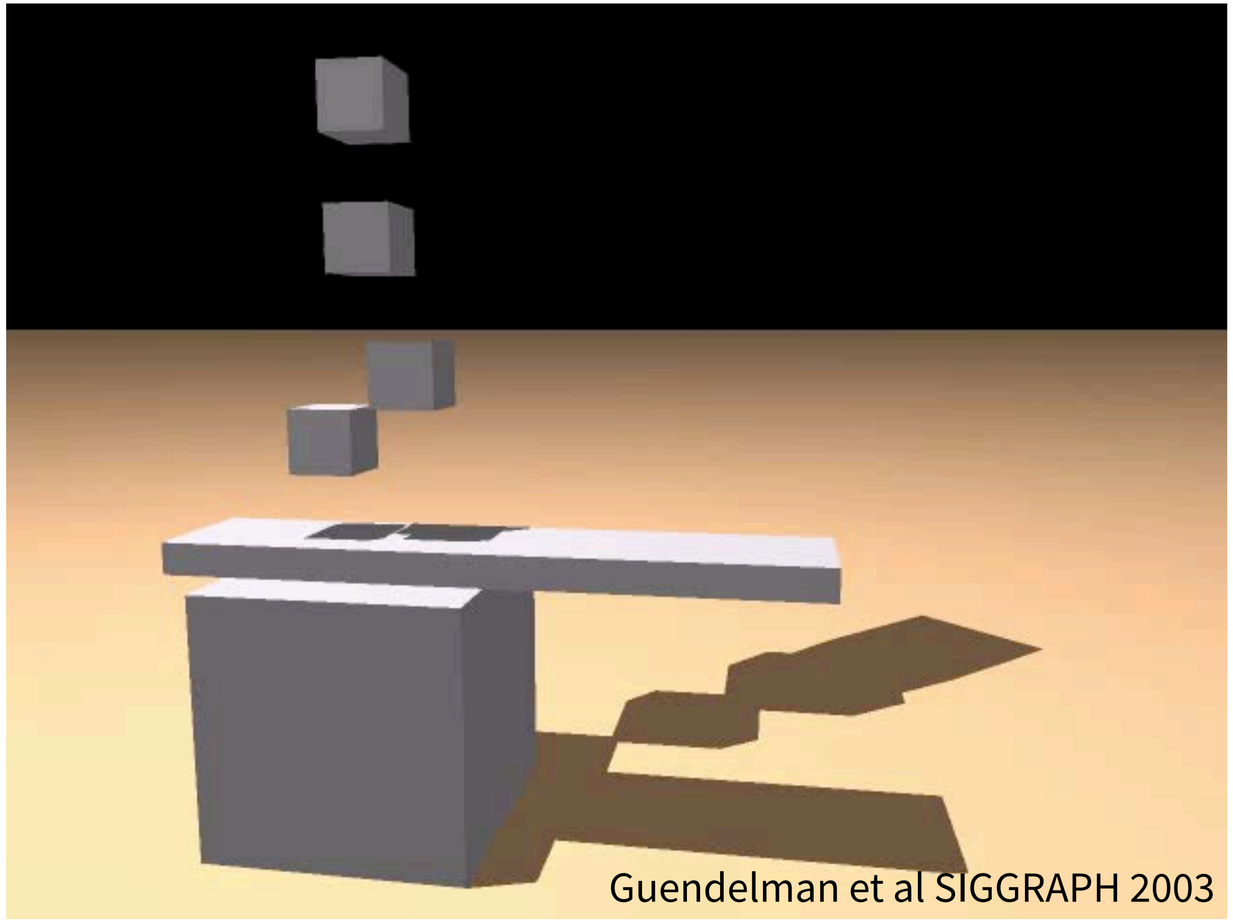
https://youtu.be/5C9BLAXCe1I?t=1m43s

Programming Assignment #3: Water Games

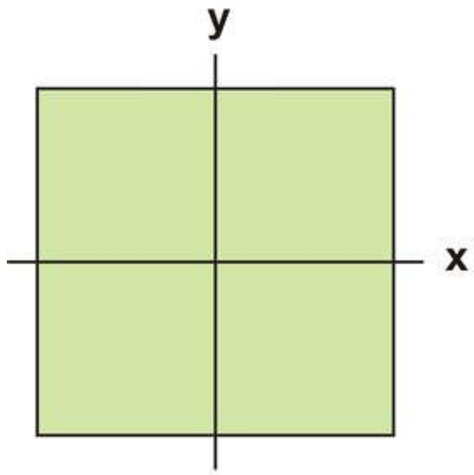
- Build an interactive 2D fluid simulation!
  - Grid-based semi-Lagrangian solver
  - Builds on P1 & P2's 2D SDF collision environment to now simulate 2D fluids
- Theme: A 2D Water-Powered Game:
  - Push objects around with fluid jets
  - Precomputed matrix factorization for real-time performance

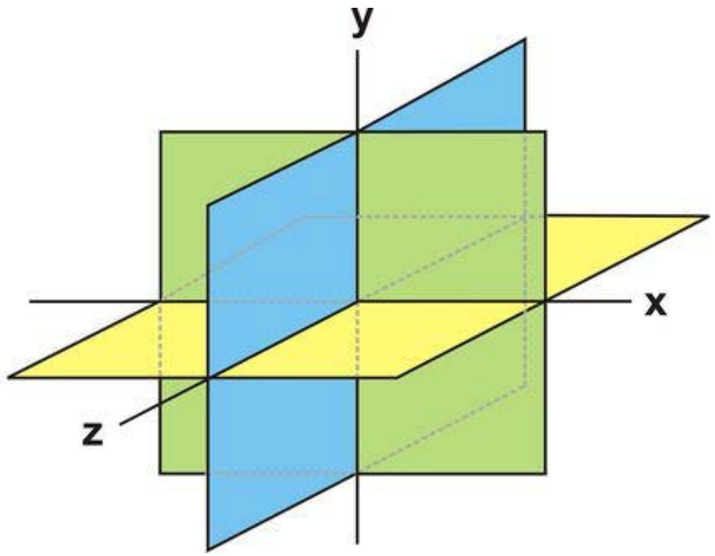
Week 6: Rigid Body Simulation

- 3D position and 3D orientation
- Rigid-body dynamics
- Collisions and constraints
- Articulated rigid bodies

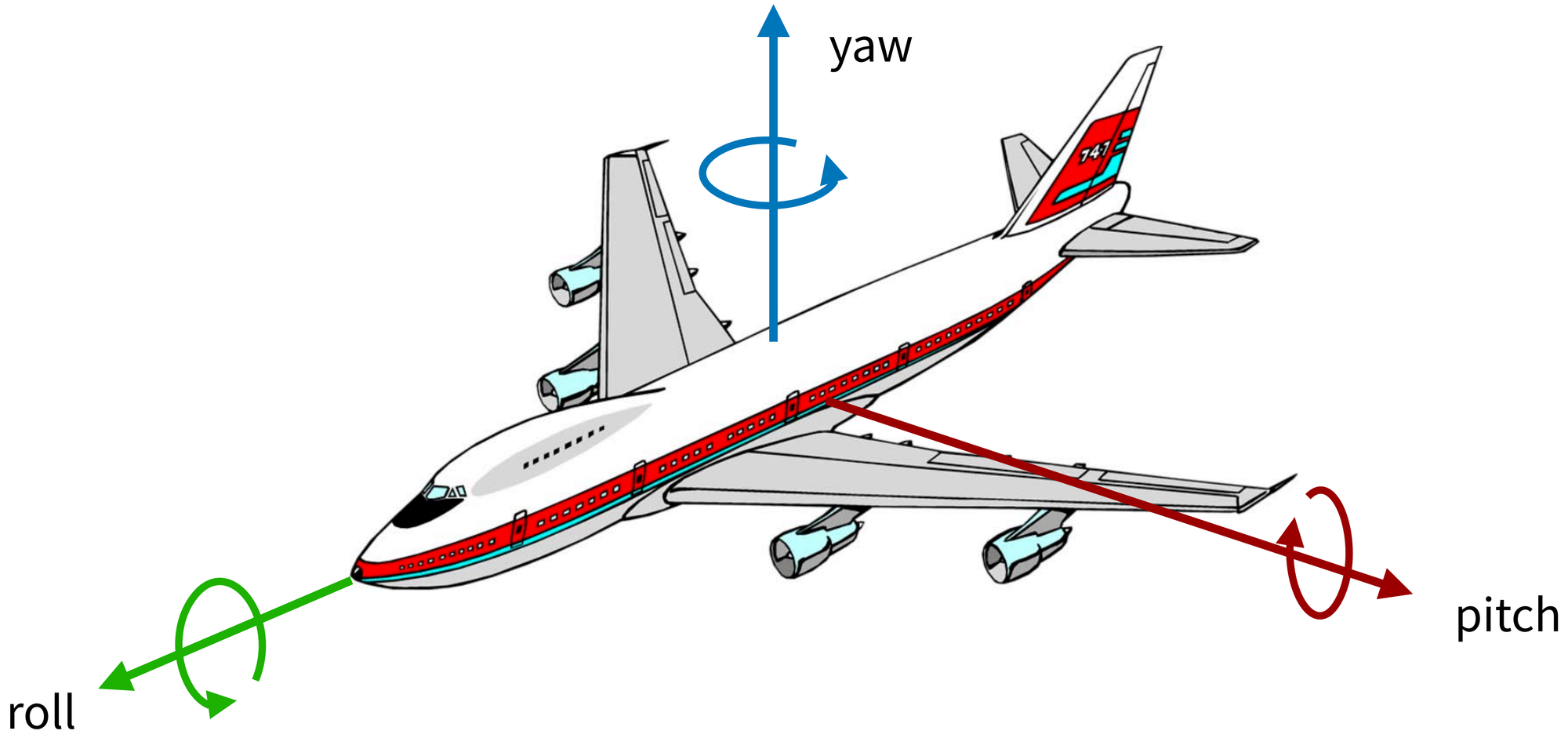


3D position



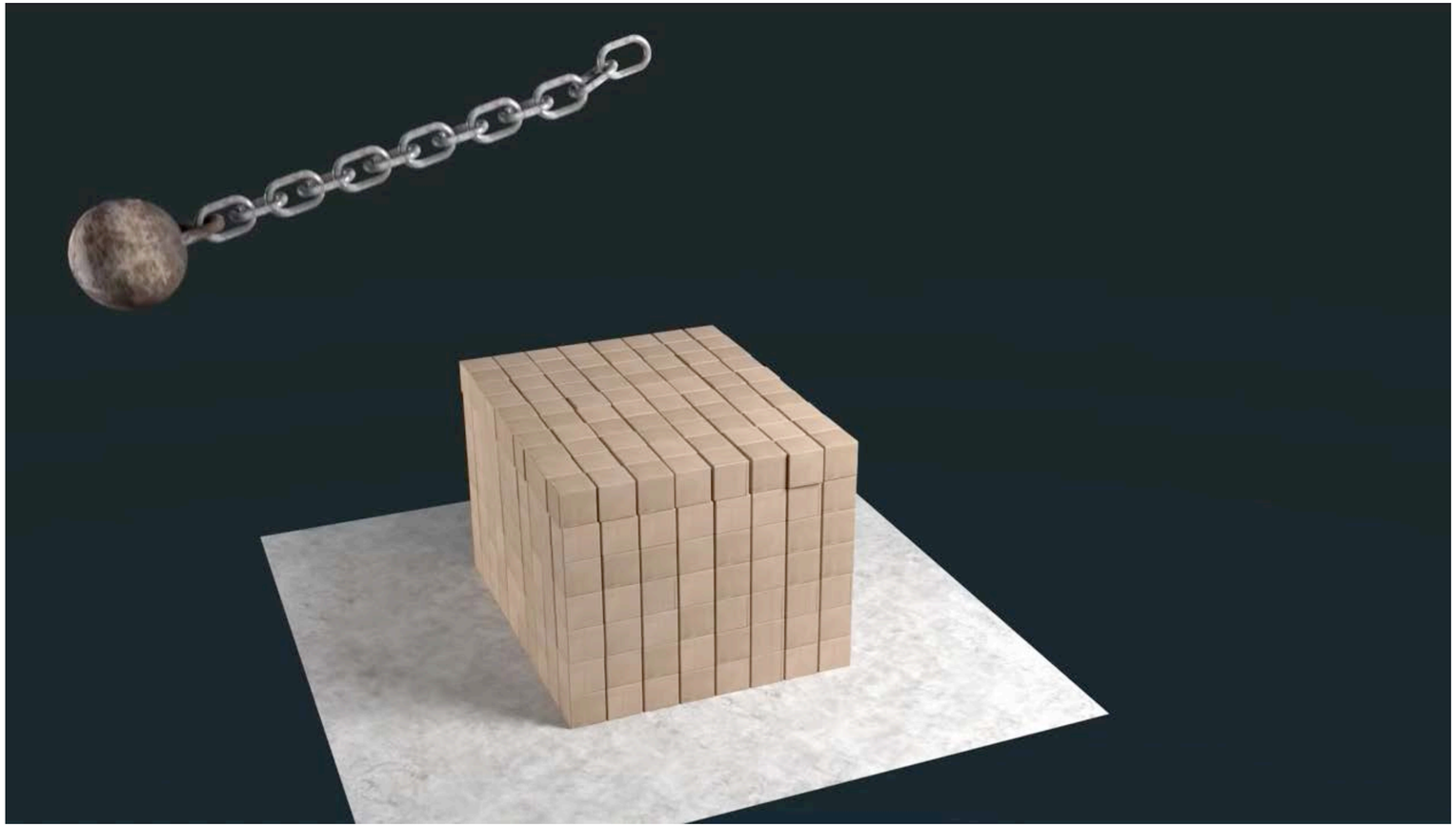


3D orientation

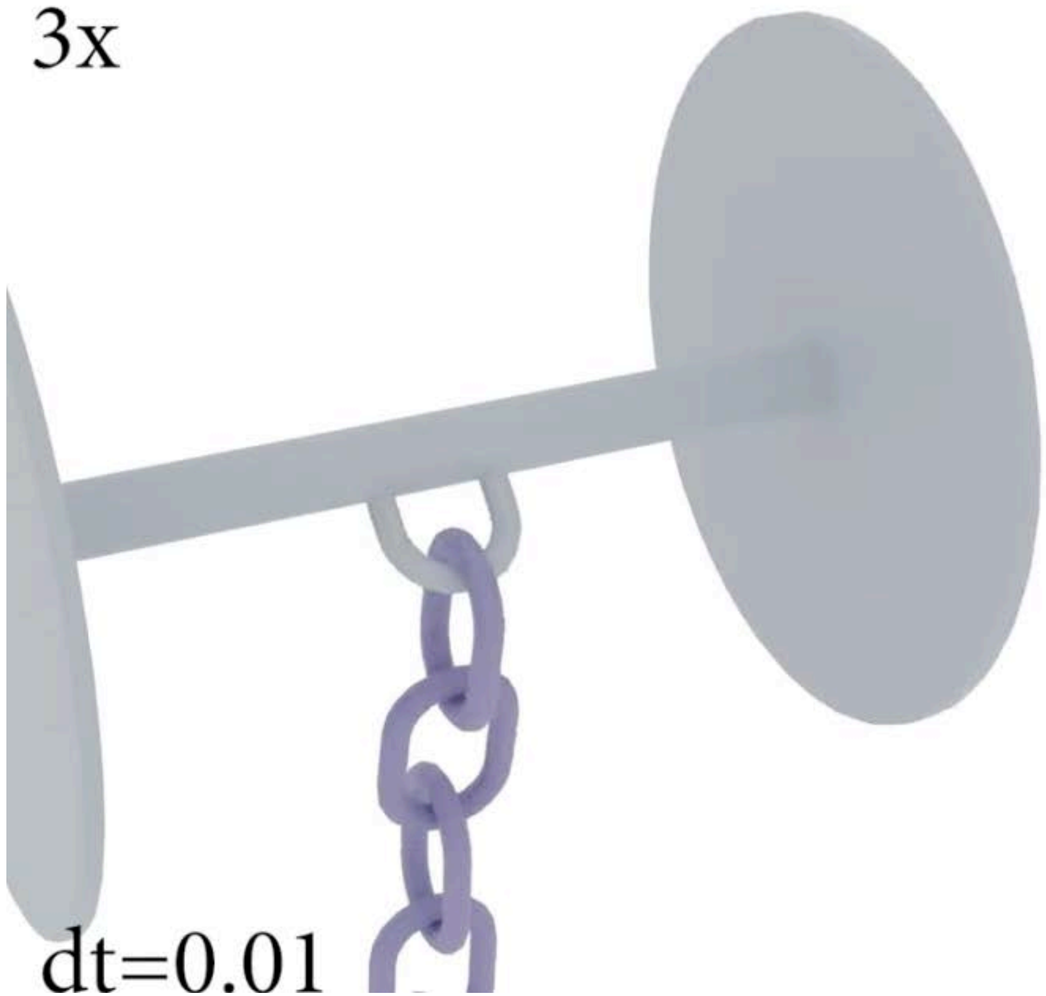


Rigid body simulation

- Objects that translate and rotate freely in 3D without deformation.
- Satisfy Newton-Euler equations.
- Contact and collision need to be handled through forces.



Rigid body simulation

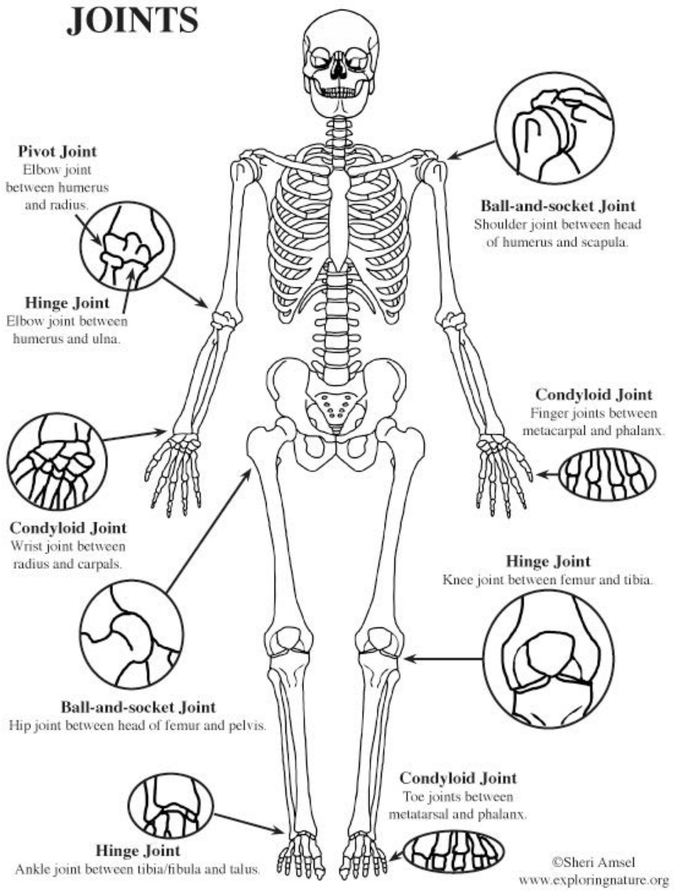


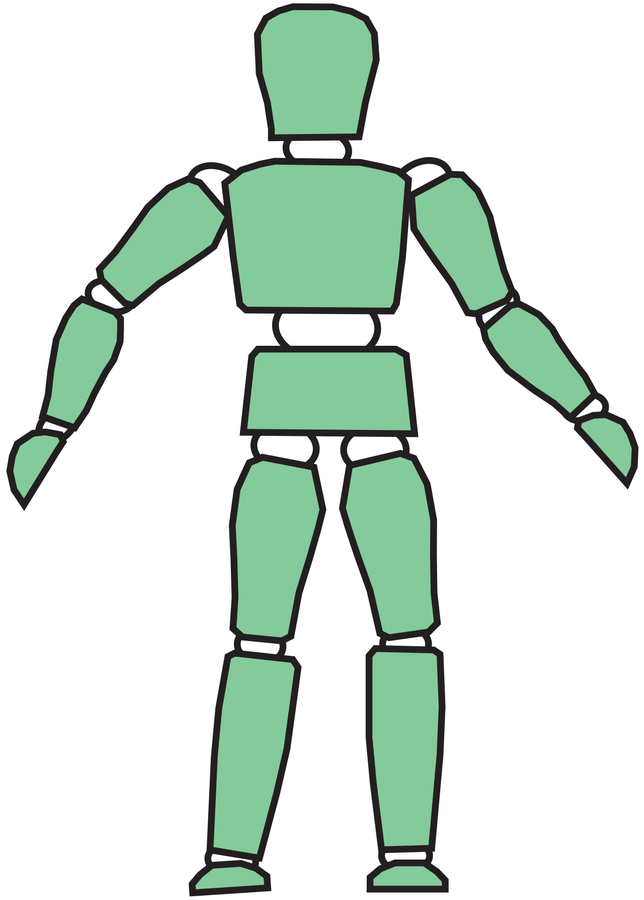
Furguson et al SIGGRAPH 2021

Week 7: Character Modeling

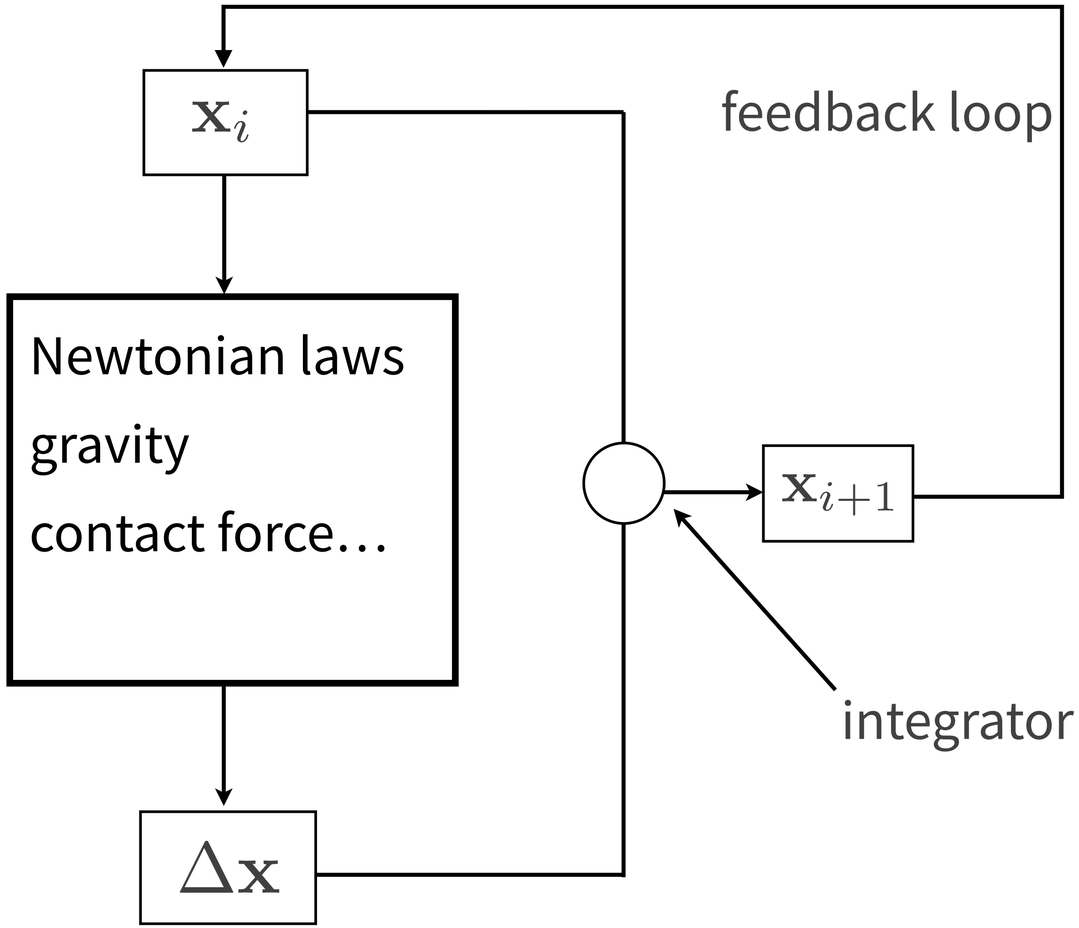
- Human skeletal systems
  - Hierarchical transformations
  - Parameterization of human kinematics
- Skinning
  - Deformation of human body
  - Shape interpolation
  - Skeleton subspace and pose space deformation
  - Data-driven approach

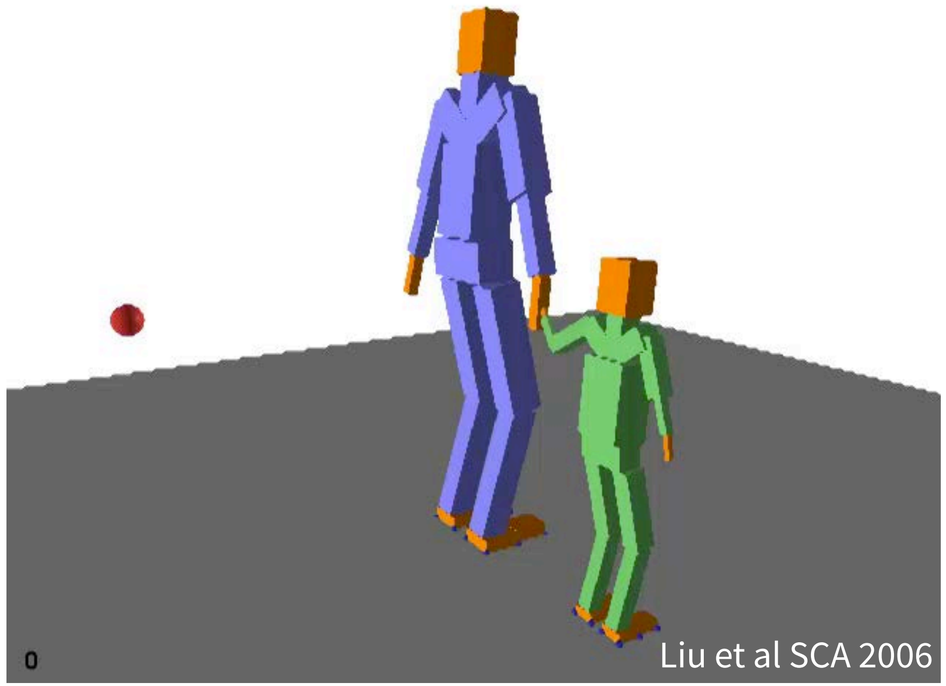
What's a compact representation for human poses?





Simulate human as articulated rigid bodies





Skinning

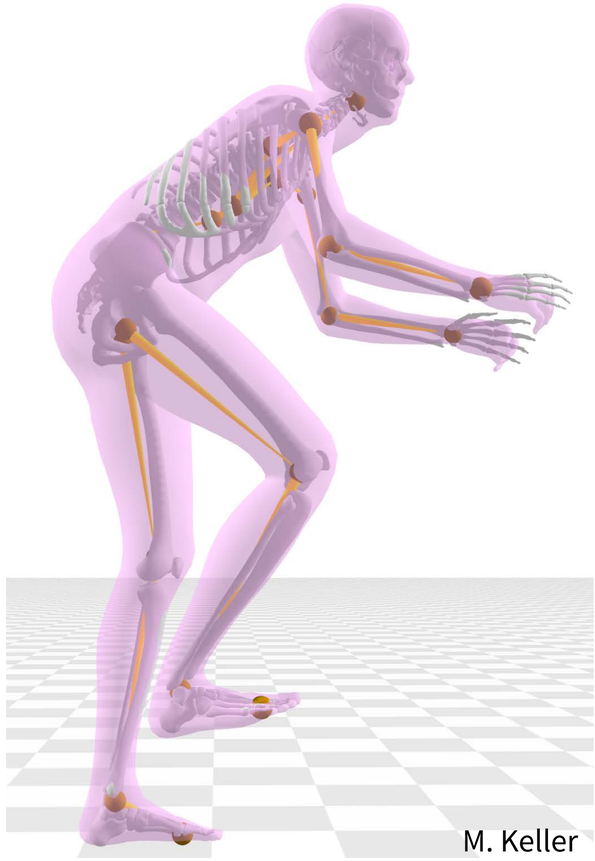
Park et al SIGGRAPH 2006

$$\mathbf{p} = f(\mathbf{q}; \boldsymbol{\theta})$$

skinning parameters

body shape

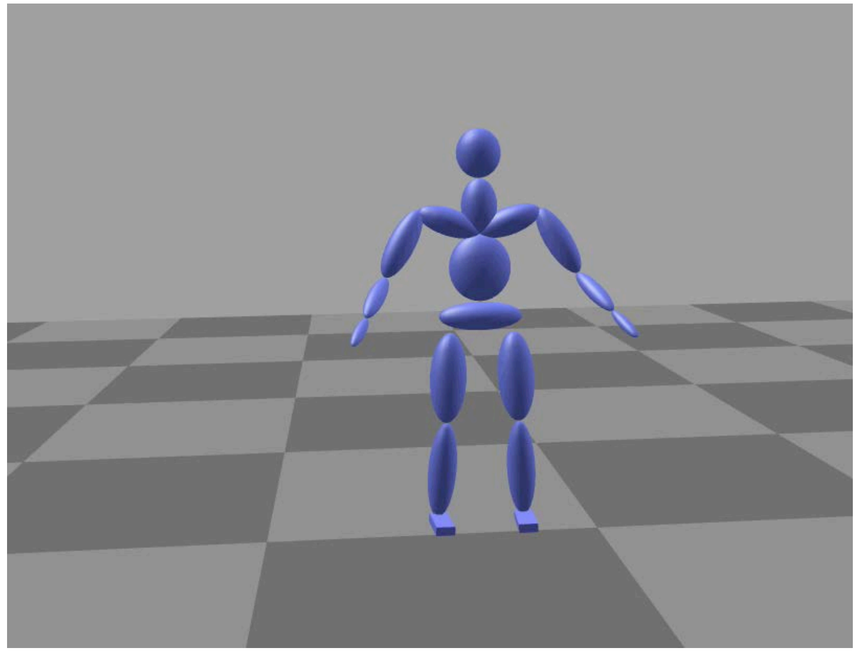
skeleton pose



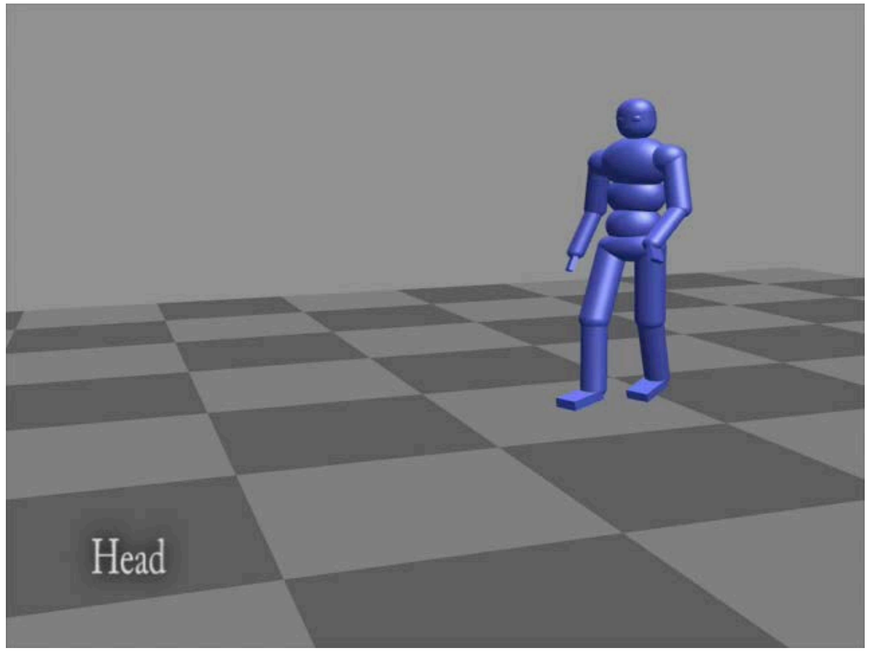
M. Keller

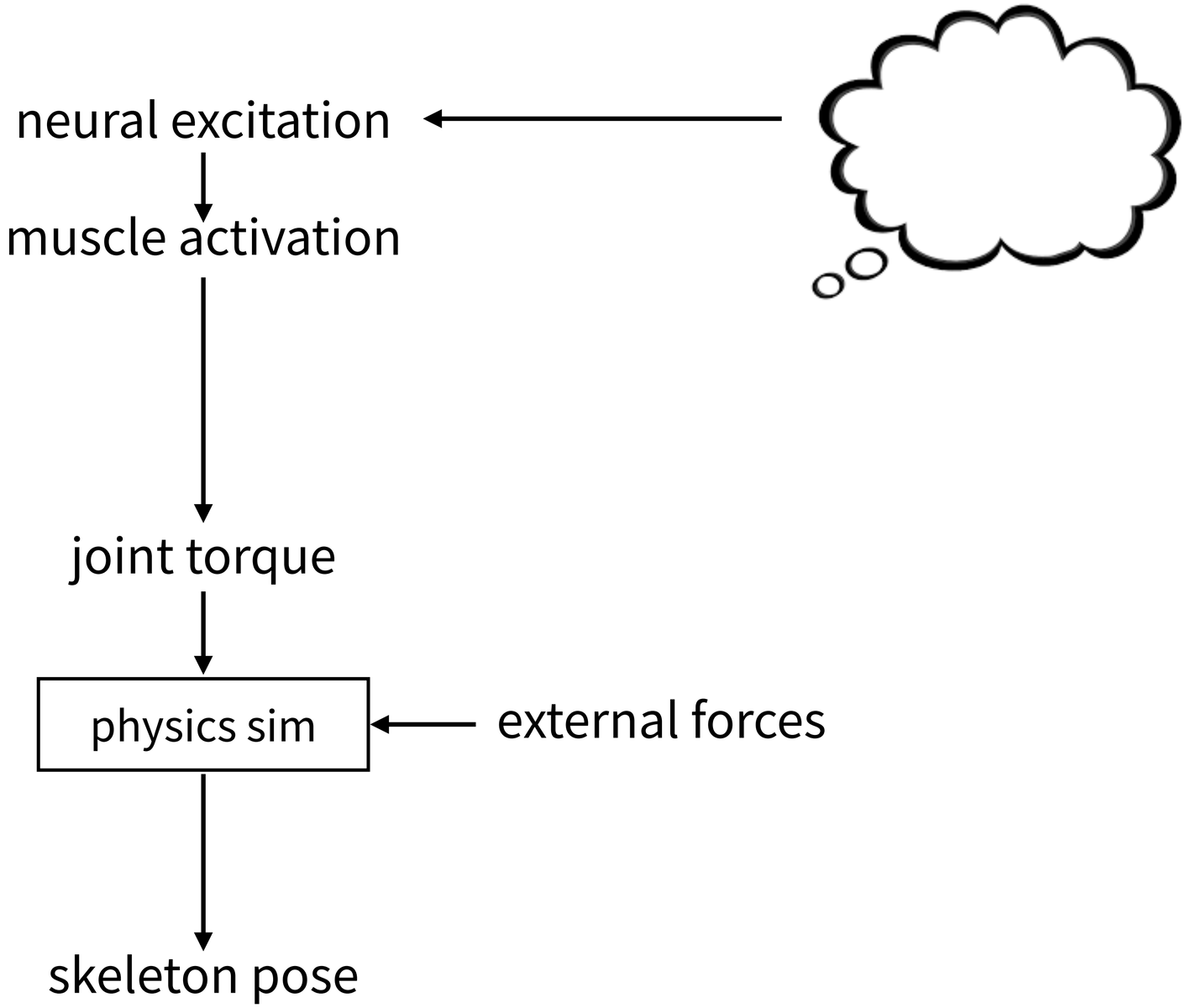
How do we control human motion?

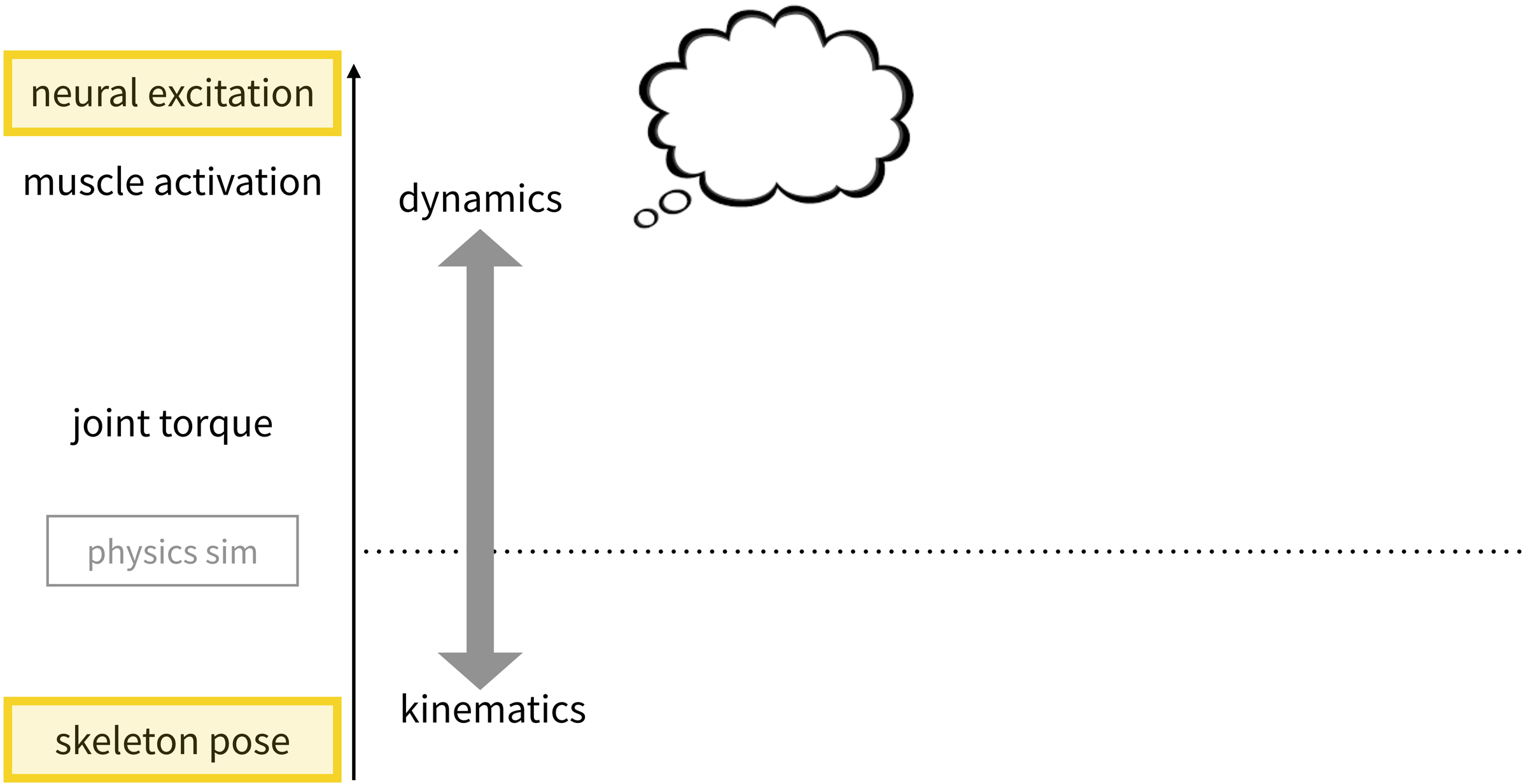
physics without control



physics with control




neural excitation
muscle activation
joint torque
physics sim
external forces
skeleton pose


neural excitation
muscle activation
dynamics
joint torque
physics sim
kinematics
skeleton pose

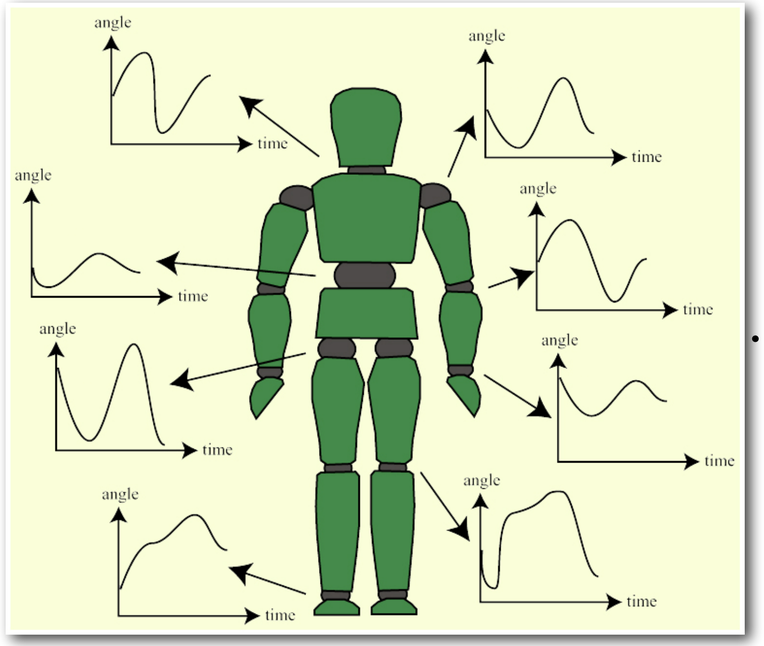
neural excitation

muscle activation

joint torque

physics sim

skeleton pose



trajectory

abstract goal

“walk over there, don’t fall”

neural excitation

muscle activation

joint torque

physics sim

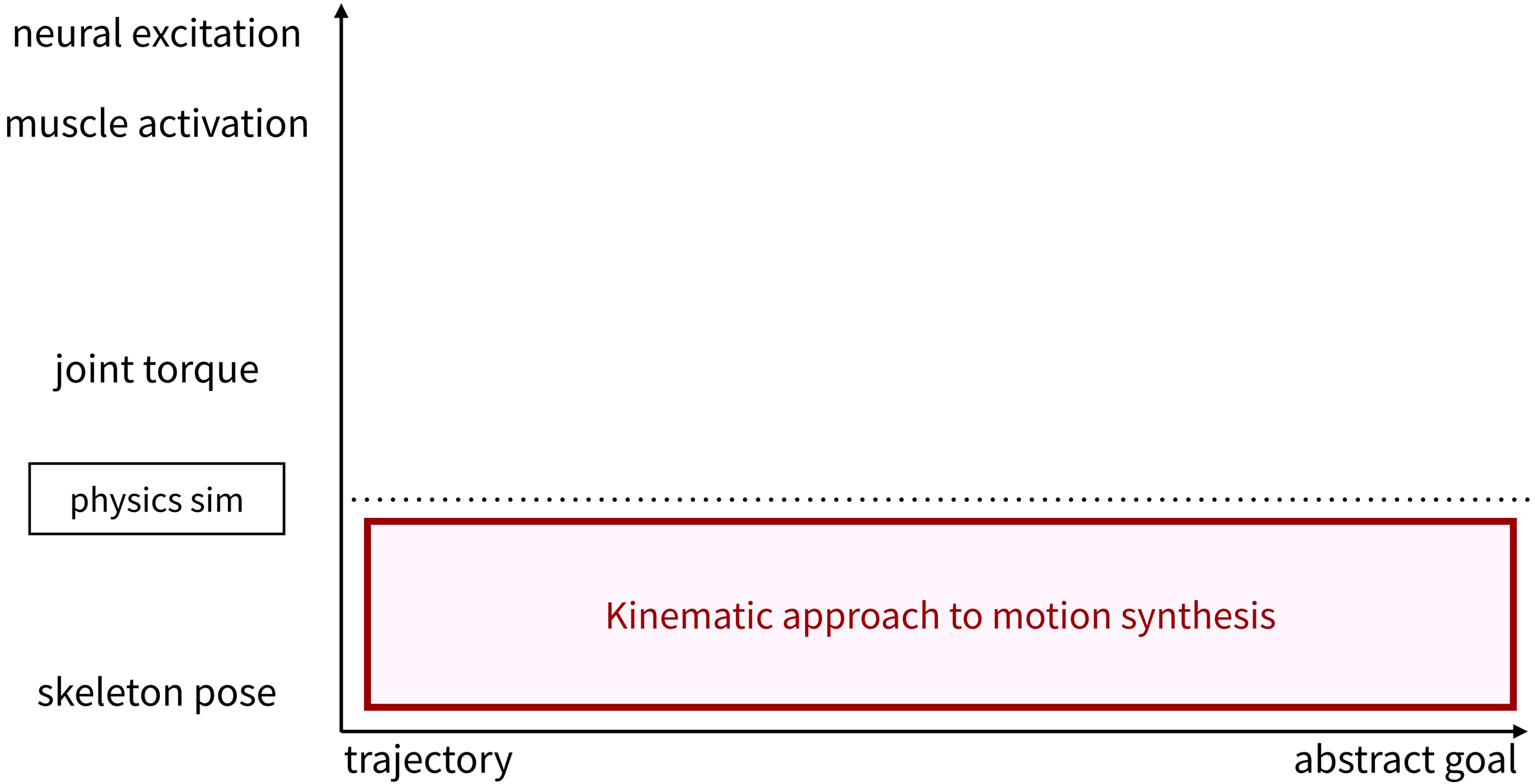
skeleton pose

naturalness

generality

trajectory

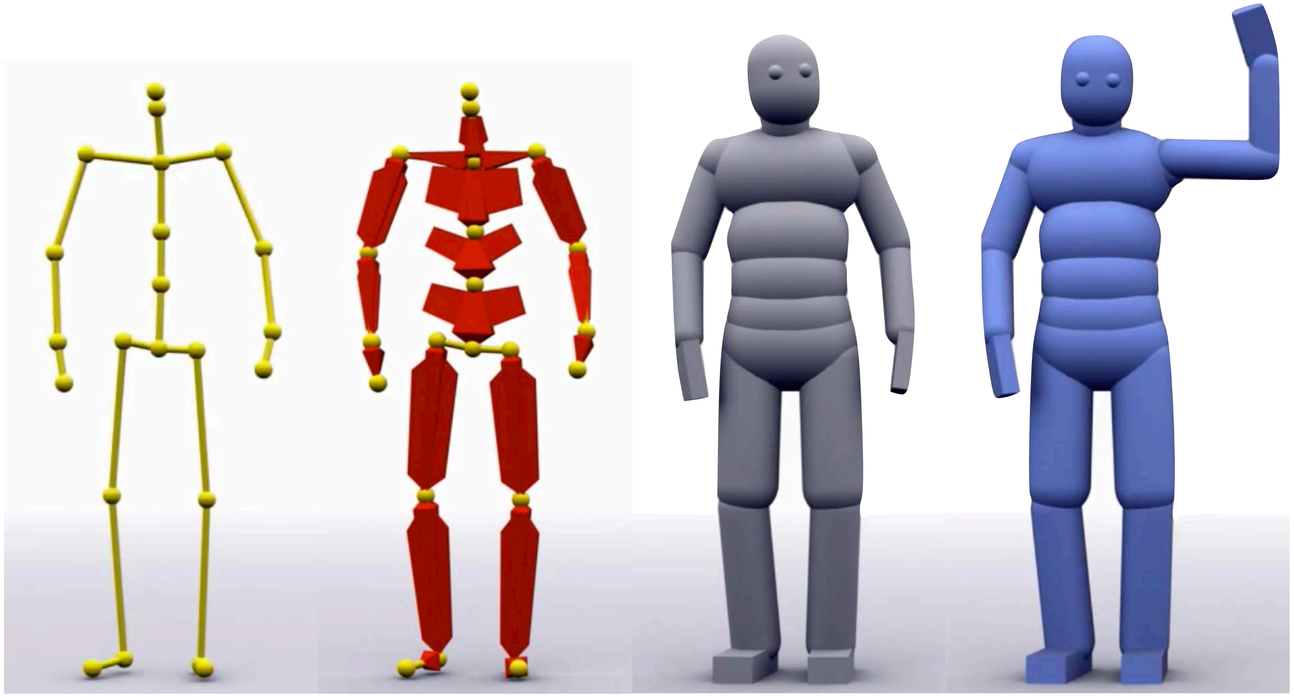
abstract goal


neural excitation
muscle activation
joint torque
physics sim
skeleton pose
Kinematic approach to motion synthesis
trajectory
abstract goal

Week 8: Inverse Kinematics and Keyframes

- Inverse kinematics
  - IK constraints
  - Efficient computation for Jacobian matrix
  - Solving IK as a non-linear optimization
  - Heuristics-based IK methods
- Keyframe animation
  - Interpolation using cubic curves
  - Interpolating multiple keyframes using splines
  - Properties of splines

Human kinematics



kin·e·mat·ics

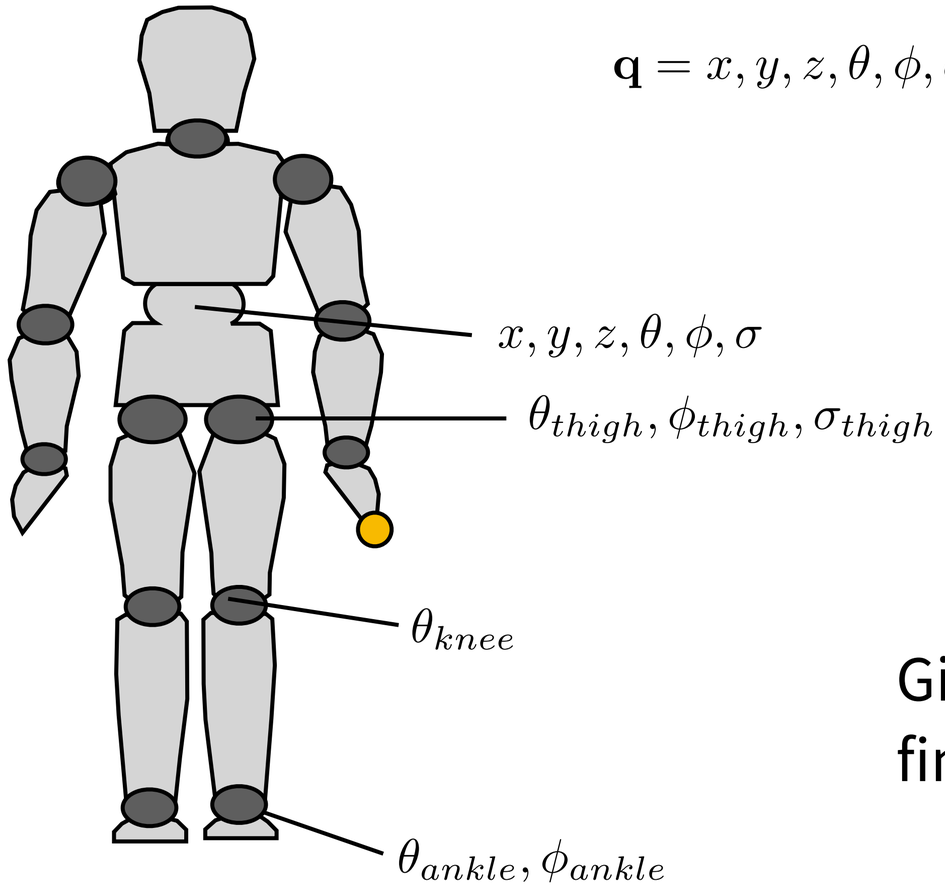
/ˌkinəˈmadiks/

noun

the branch of mechanics concerned with the motion of objects without reference to the forces which cause the motion.

- the features or properties of motion in an object.
  plural noun: **kinematics**

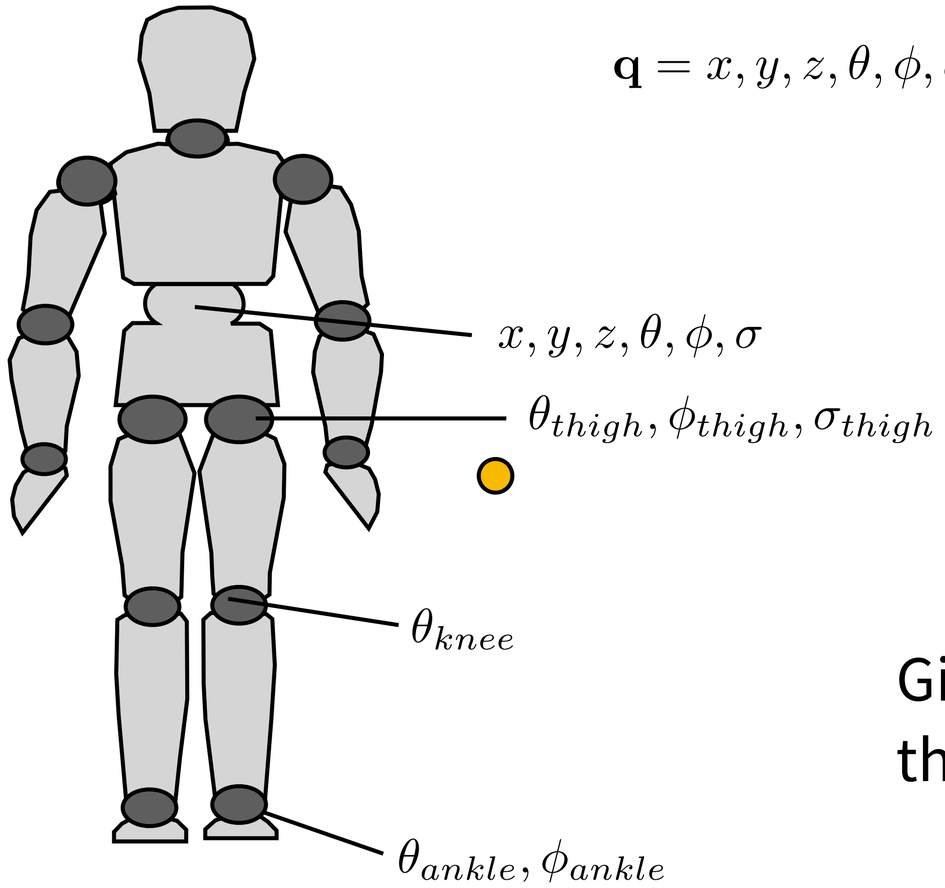
Forward kinematics



$$\mathbf{q} = x, y, z, \theta, \phi, \sigma, \theta_{thigh}, \phi_{thigh}, \sigma_{thigh}, \theta_{knee}, \theta_{ankle}, \phi_{ankle}, \cdots$$

Given a pose **q**, what is the 3D position of the finger tip in the world coordinate frame?

Inverse kinematics

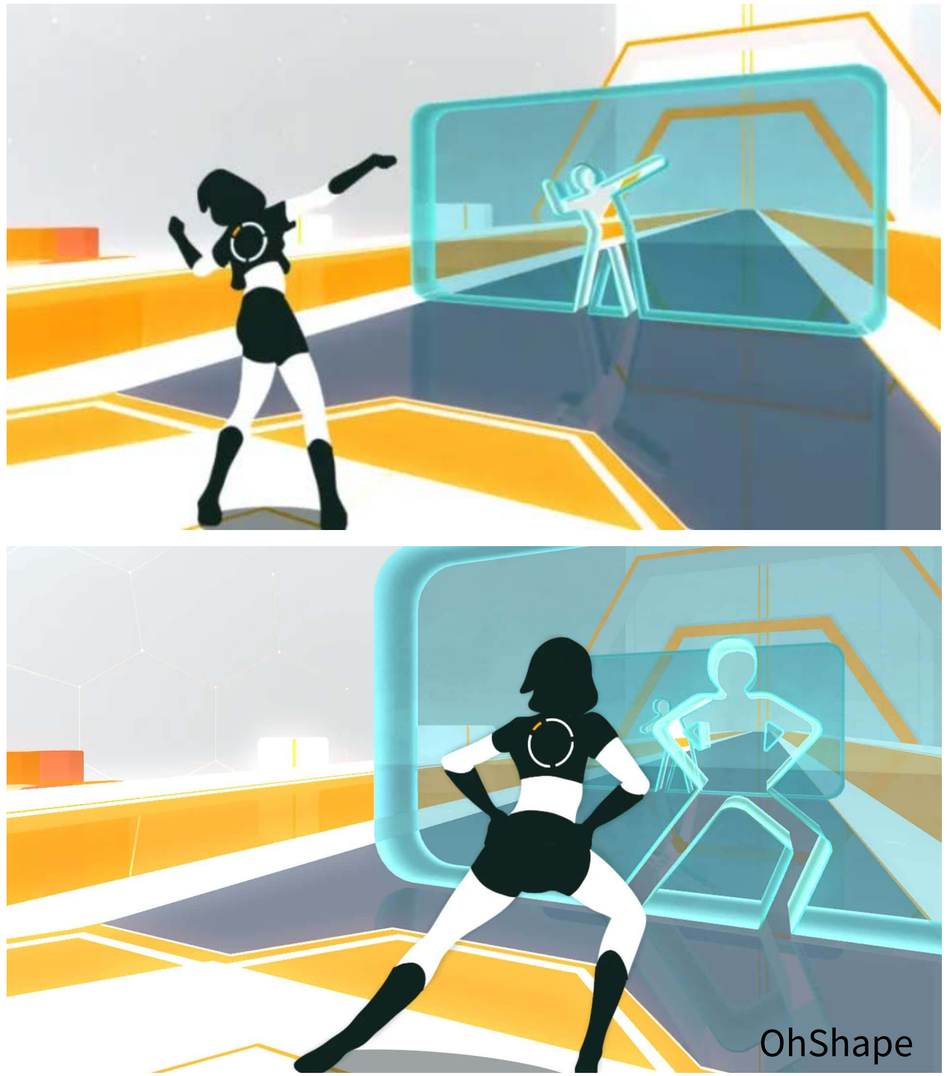


$$\mathbf{q} = x, y, z, \theta, \phi, \sigma, \theta_{thigh}, \phi_{thigh}, \sigma_{thigh}, \theta_{knee}, \theta_{ankle}, \phi_{ankle}, \cdots$$

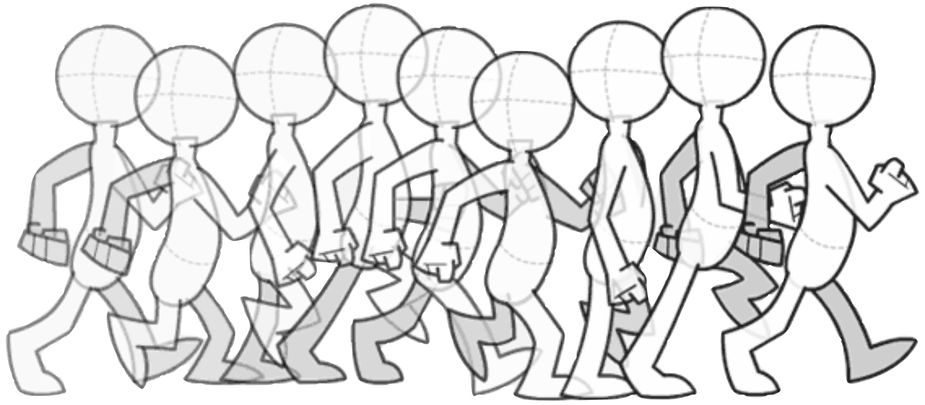
Given the target 3D position of the finger tip in the world coordinate frame, what is the pose **q**?

Programming Assignment #4: Pose Matching Game

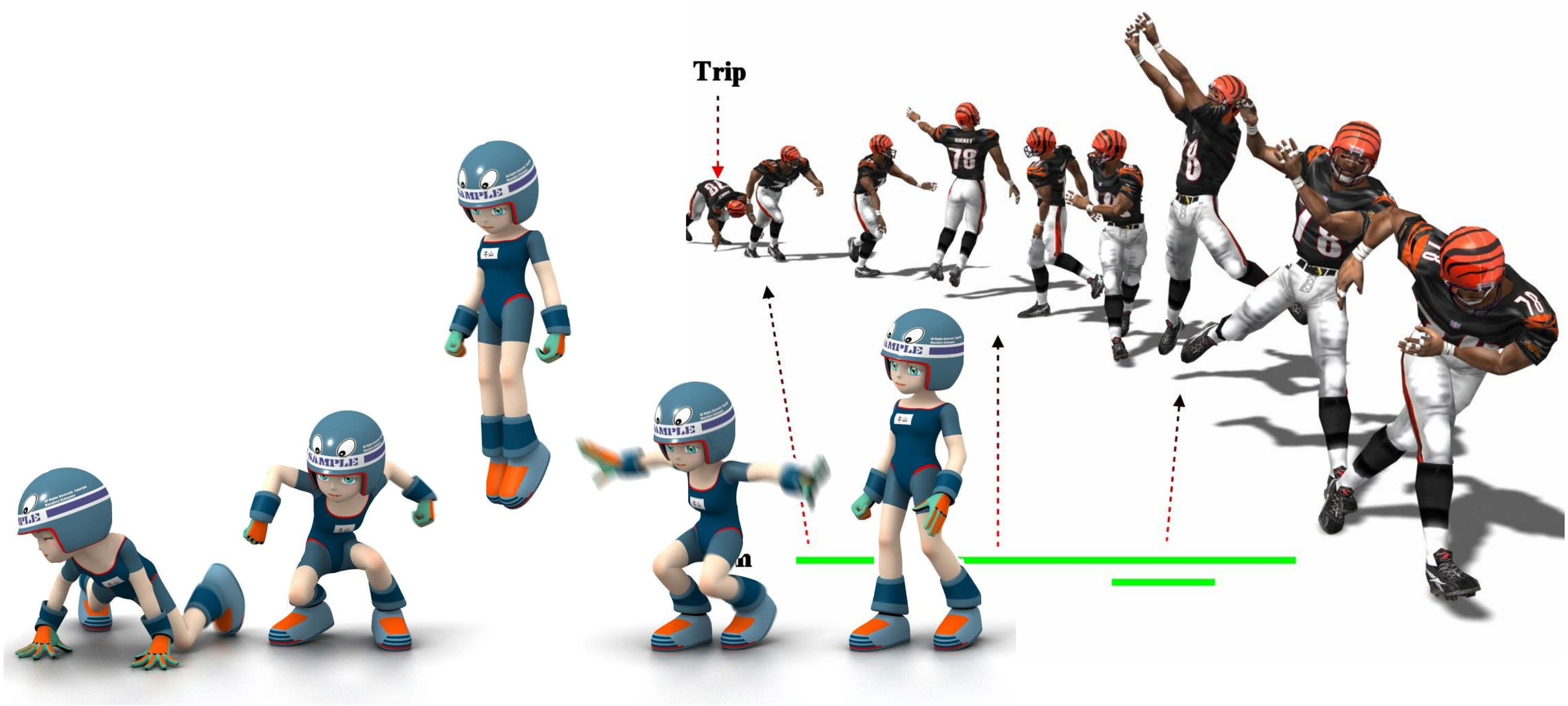
- Build a real-time inverse kinematics (IK) solver
- Fast IK solving process that supports interactive posing of the character
- Build a 3D game using IK: Control the character's pose to match the desired body silhouette



Keyframe animation

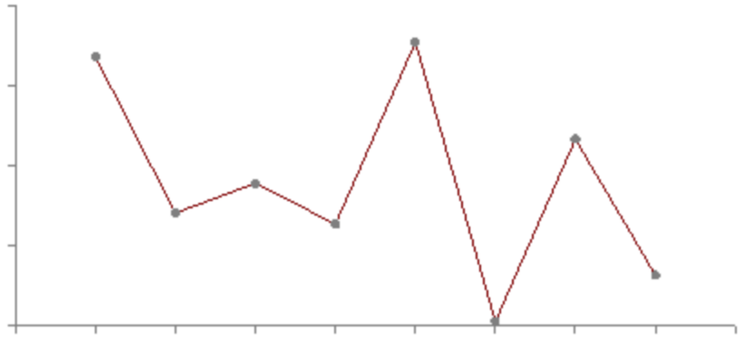


Keyframe animation

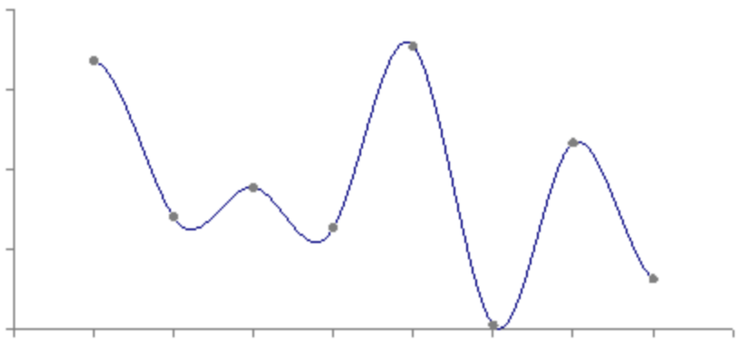


Interpolation

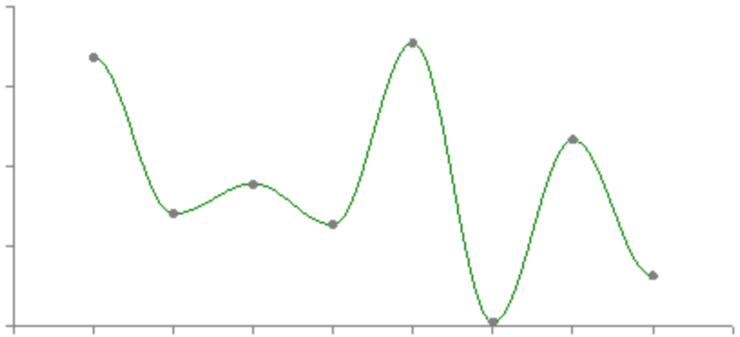
Linear



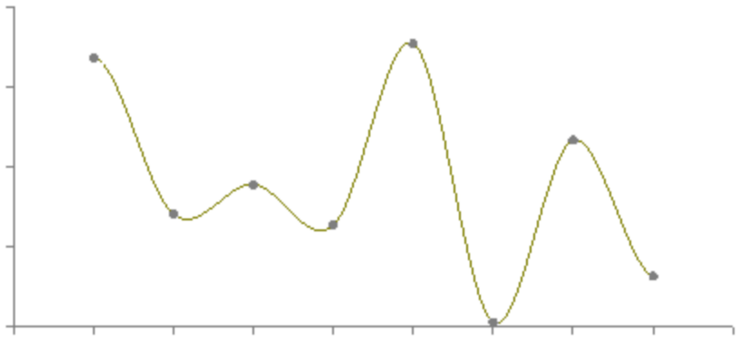
Cubic

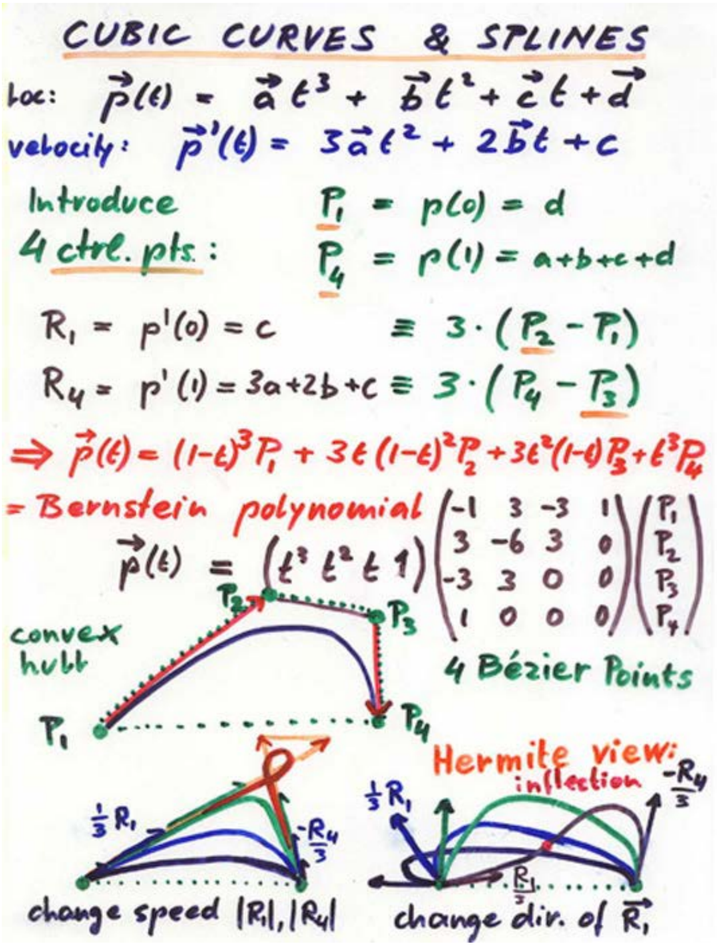


Cosine



Hermite

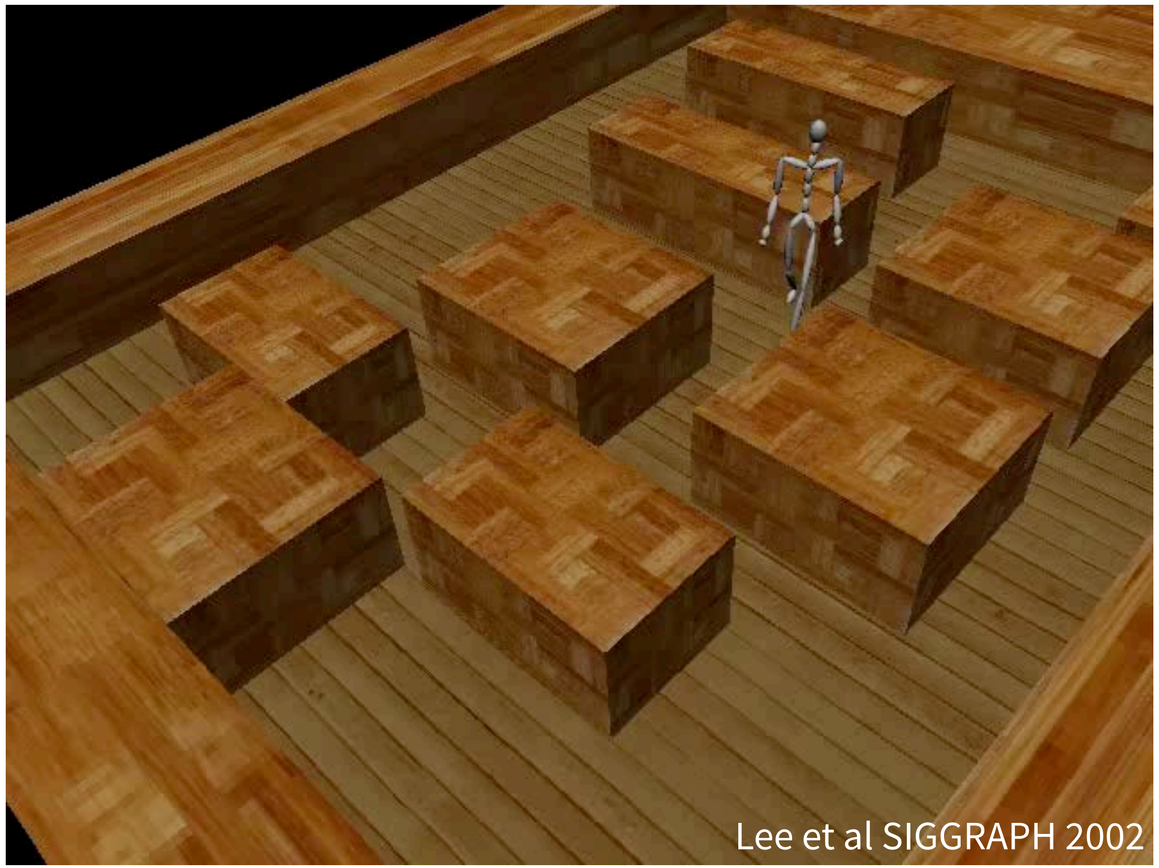




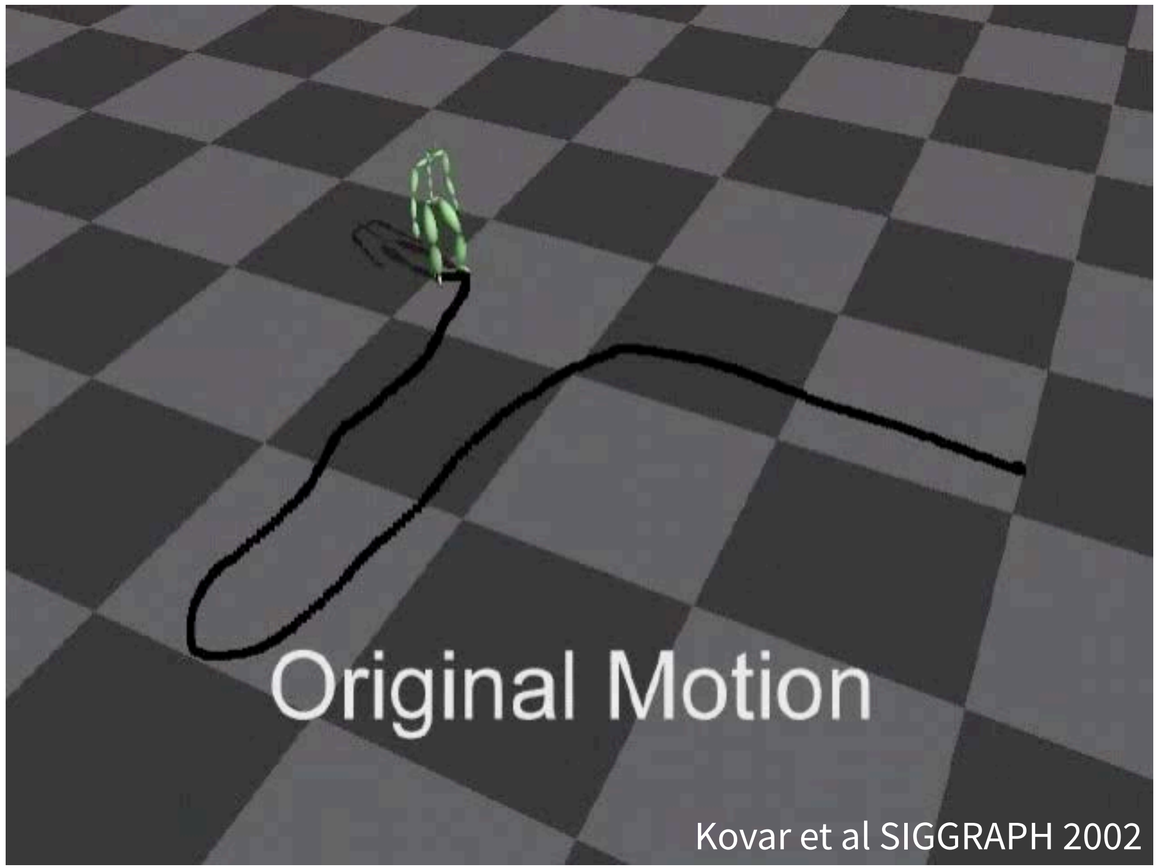
Week 9: Animation Techniques

- Data-driven and ML-based character animation
  - Motion graphs
  - Learning-based motion matching
  - Generative models for kinematic motion
- Physics-based character animation
  - Feedback control
  - Trajectory optimization
  - Reinforcement learning

Motion graphs

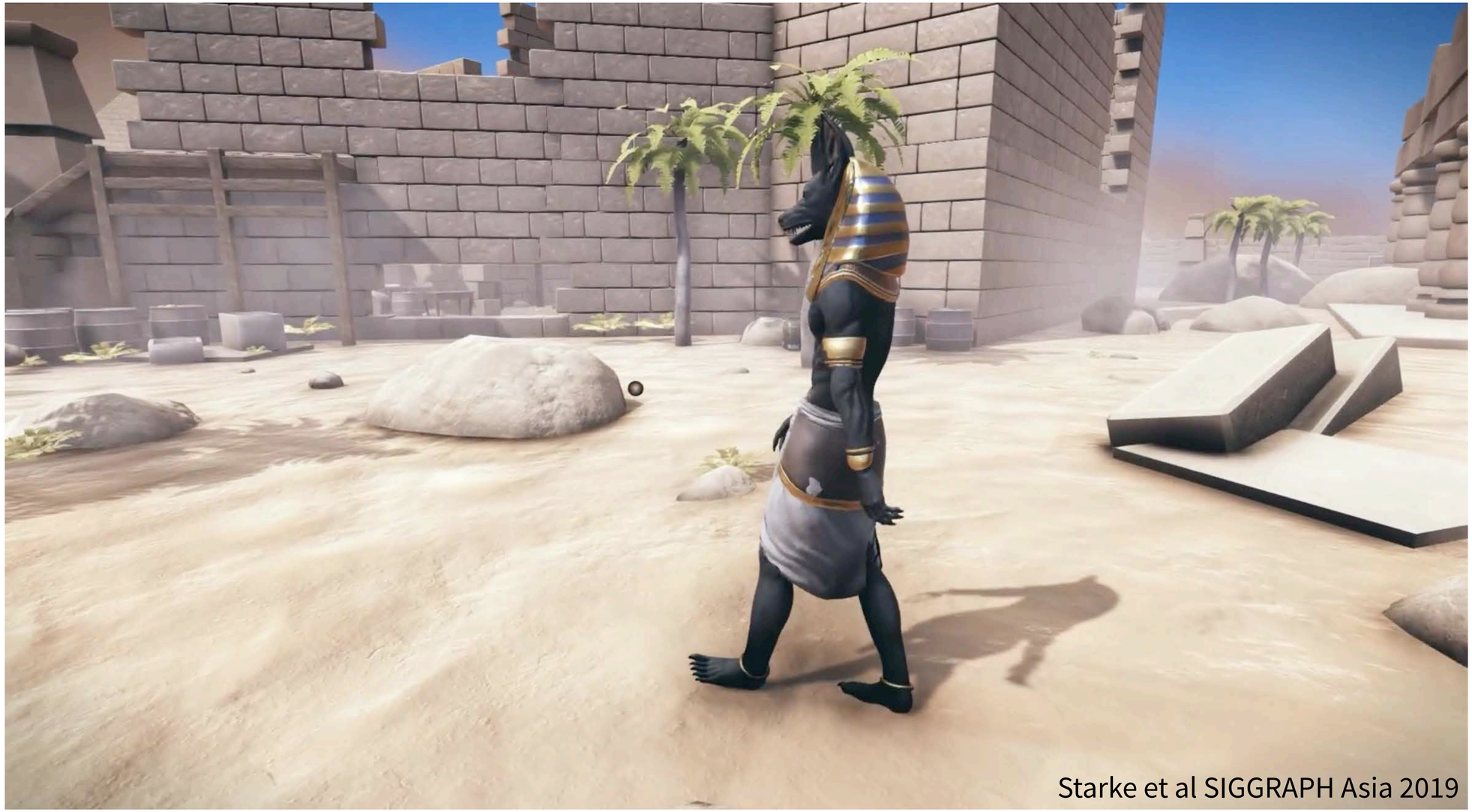


Lee et al SIGGRAPH 2002



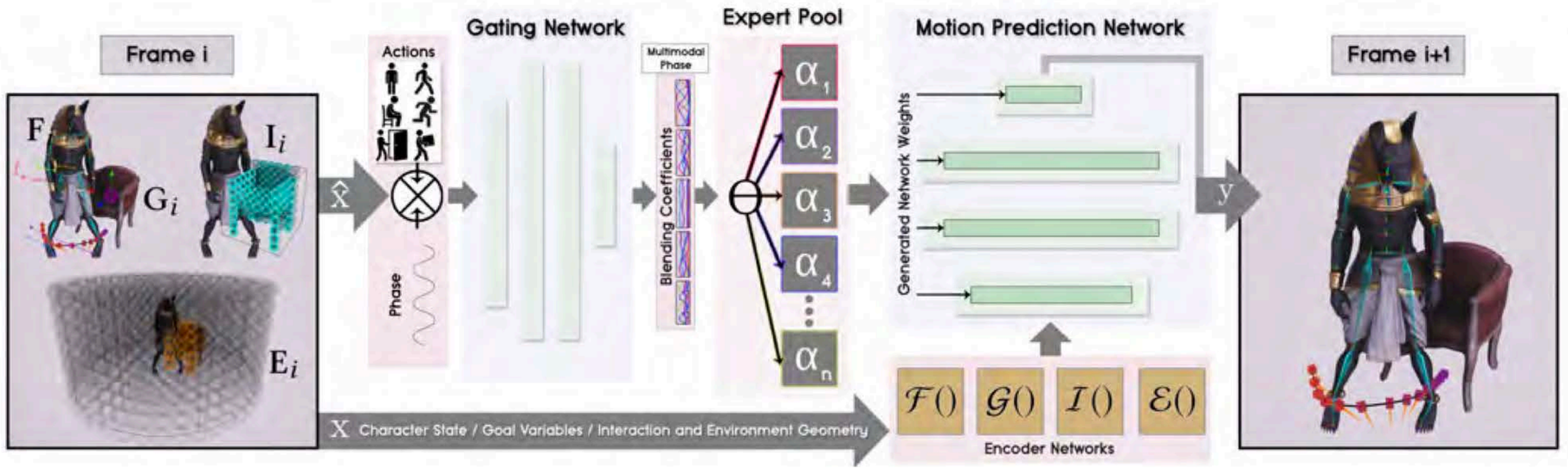
Kovar et al SIGGRAPH 2002

Generative models



Starke et al SIGGRAPH Asia 2019

Generative models



current pose

pass and future states and actions

past and future goals

environment geometry

Motion prediction network

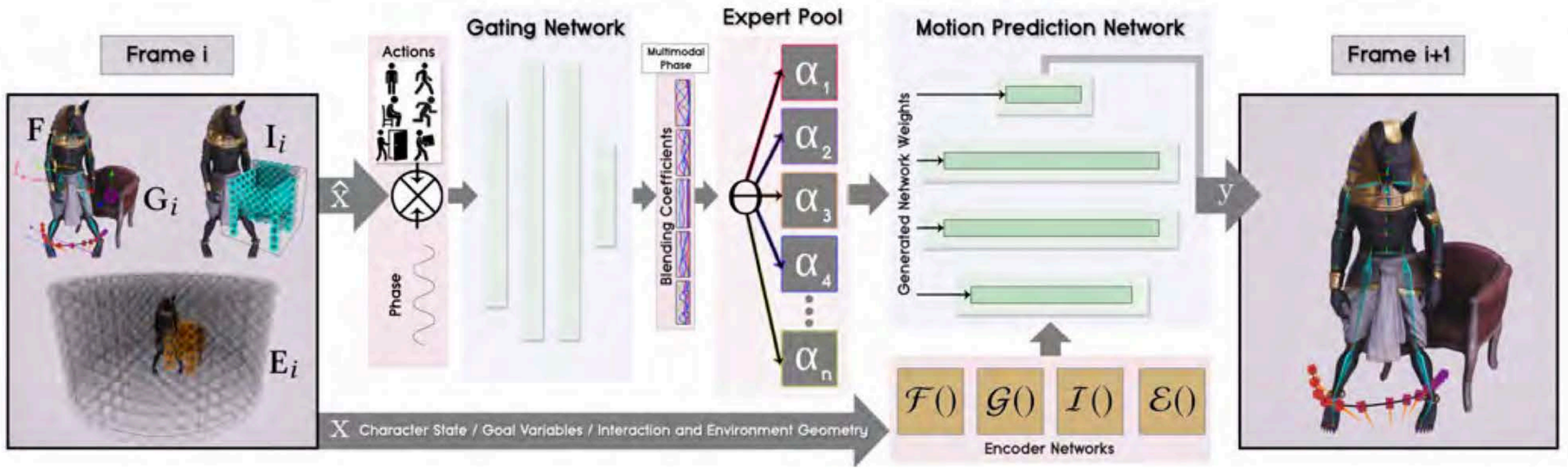
next pose

future states and actions

future goals

phase parameter

Generative models



current pose

pass and future states and actions

past and future goals

environment geometry

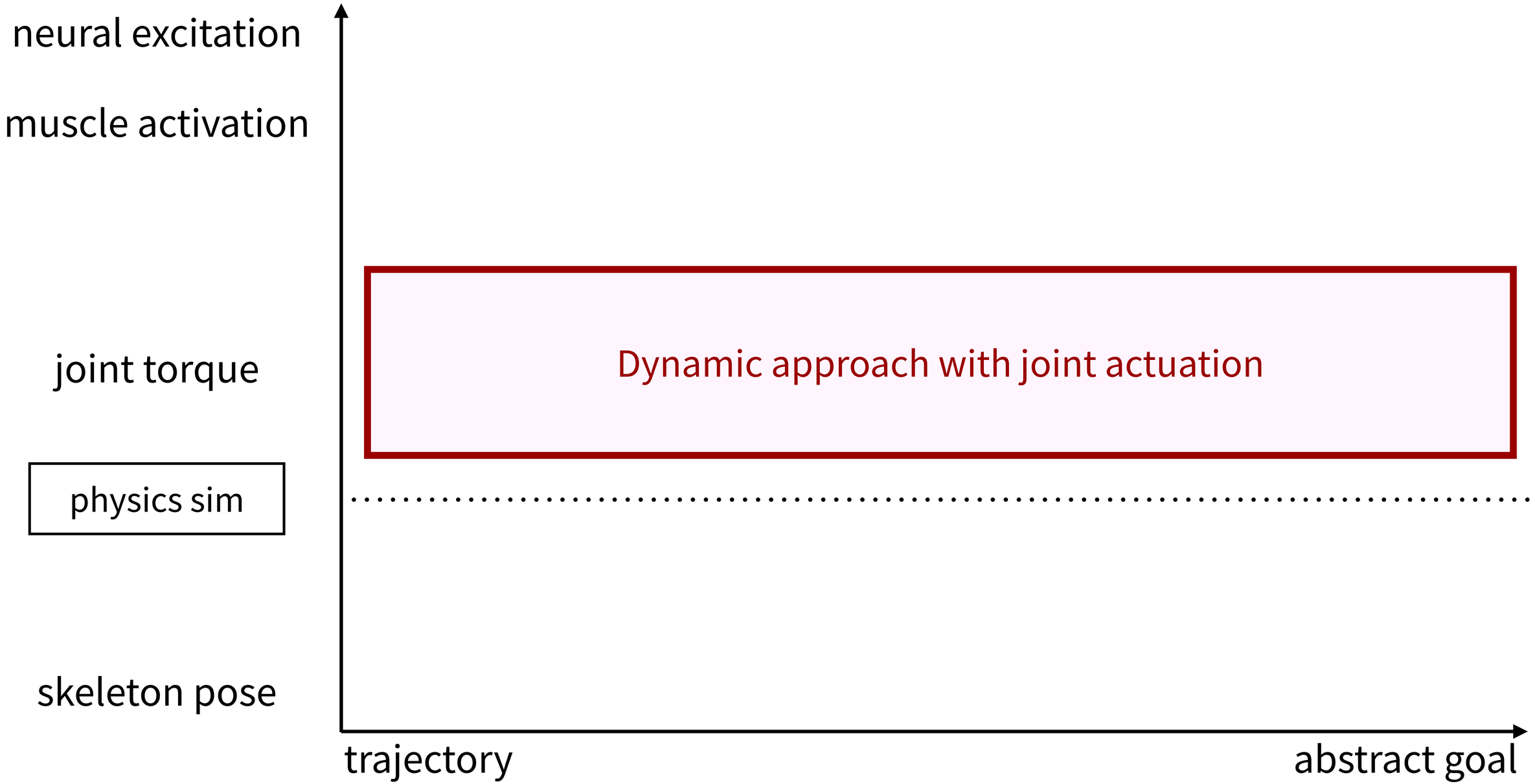
Motion prediction network

next pose

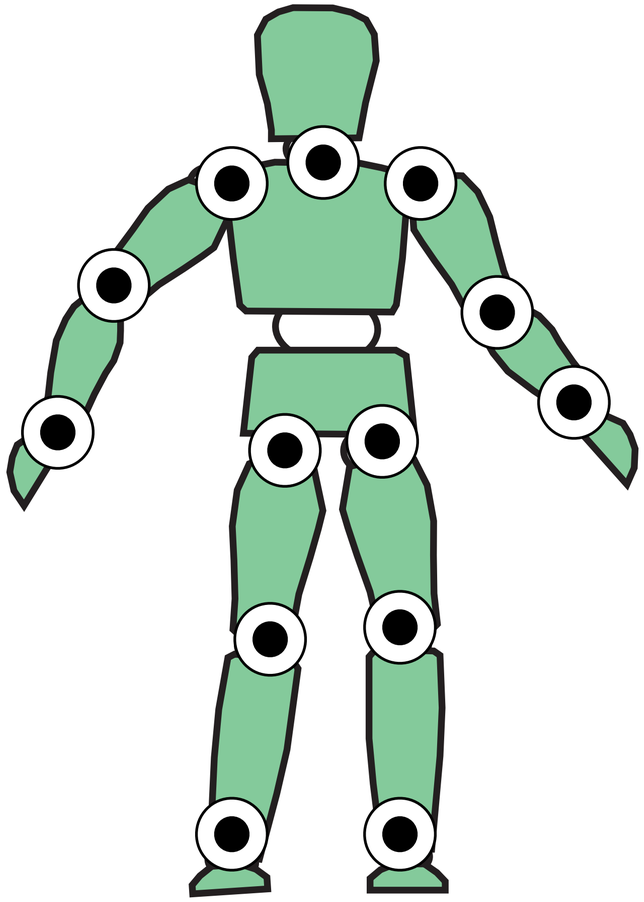
future states and actions

future goals

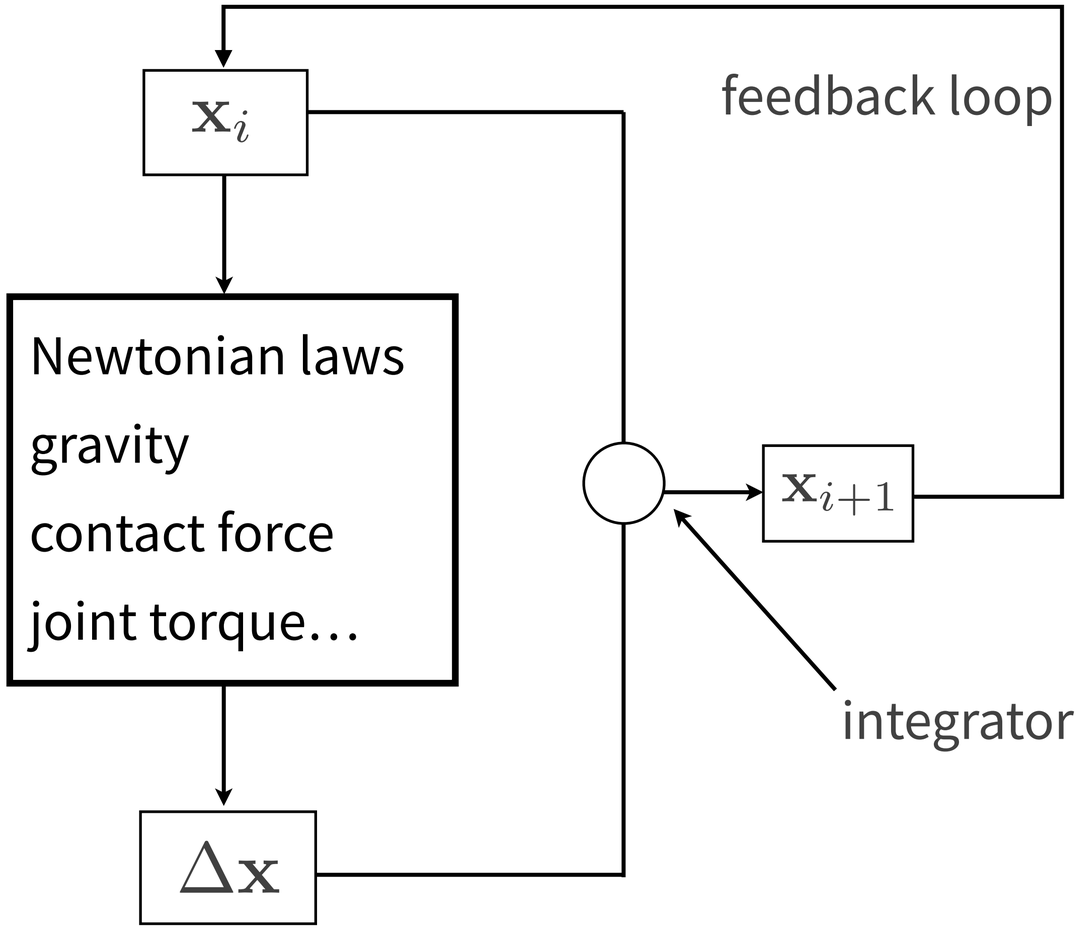
phase parameter

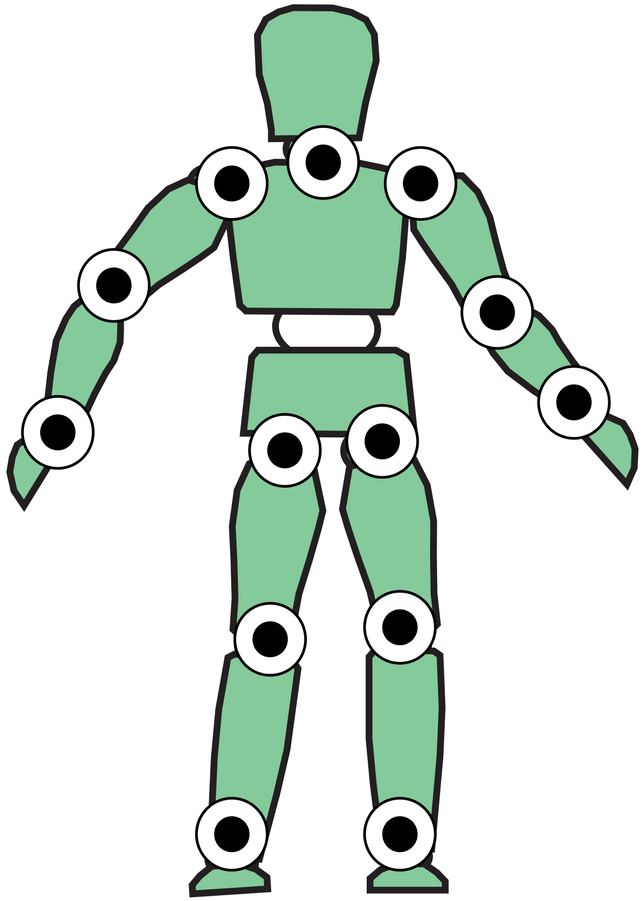

neural excitation
muscle activation
joint torque
physics sim
skeleton pose
Dynamic approach with joint actuation
trajectory
abstract goal

Control human motion

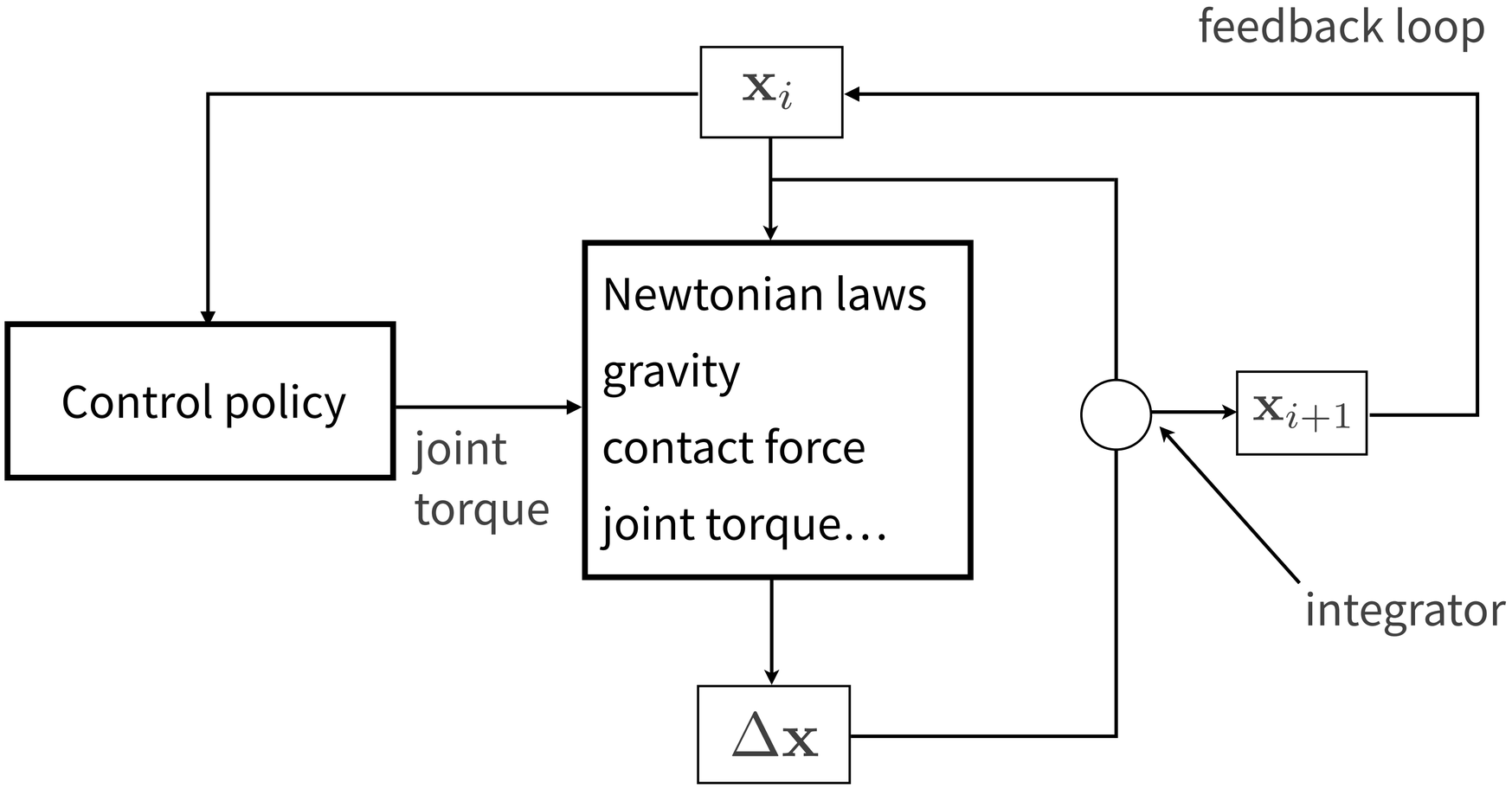


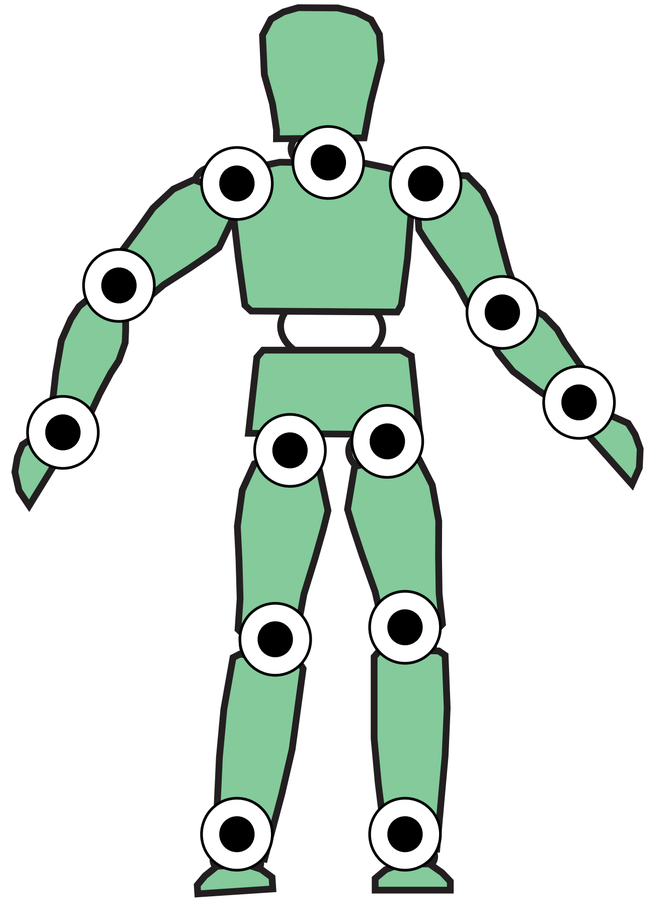
Physics simulation



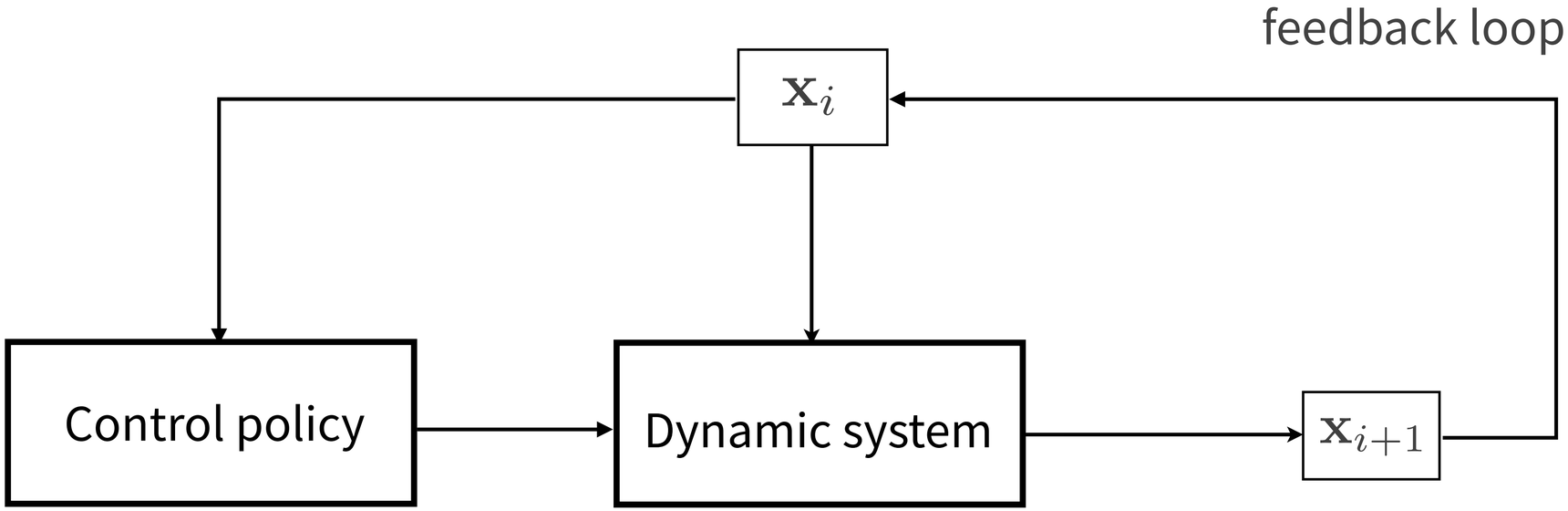


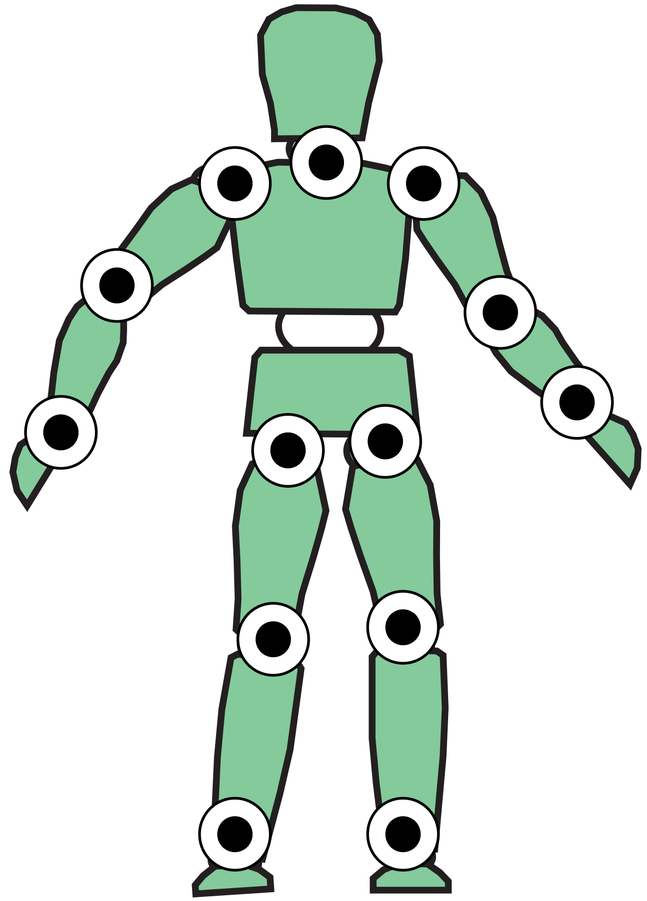
Control a dynamic system



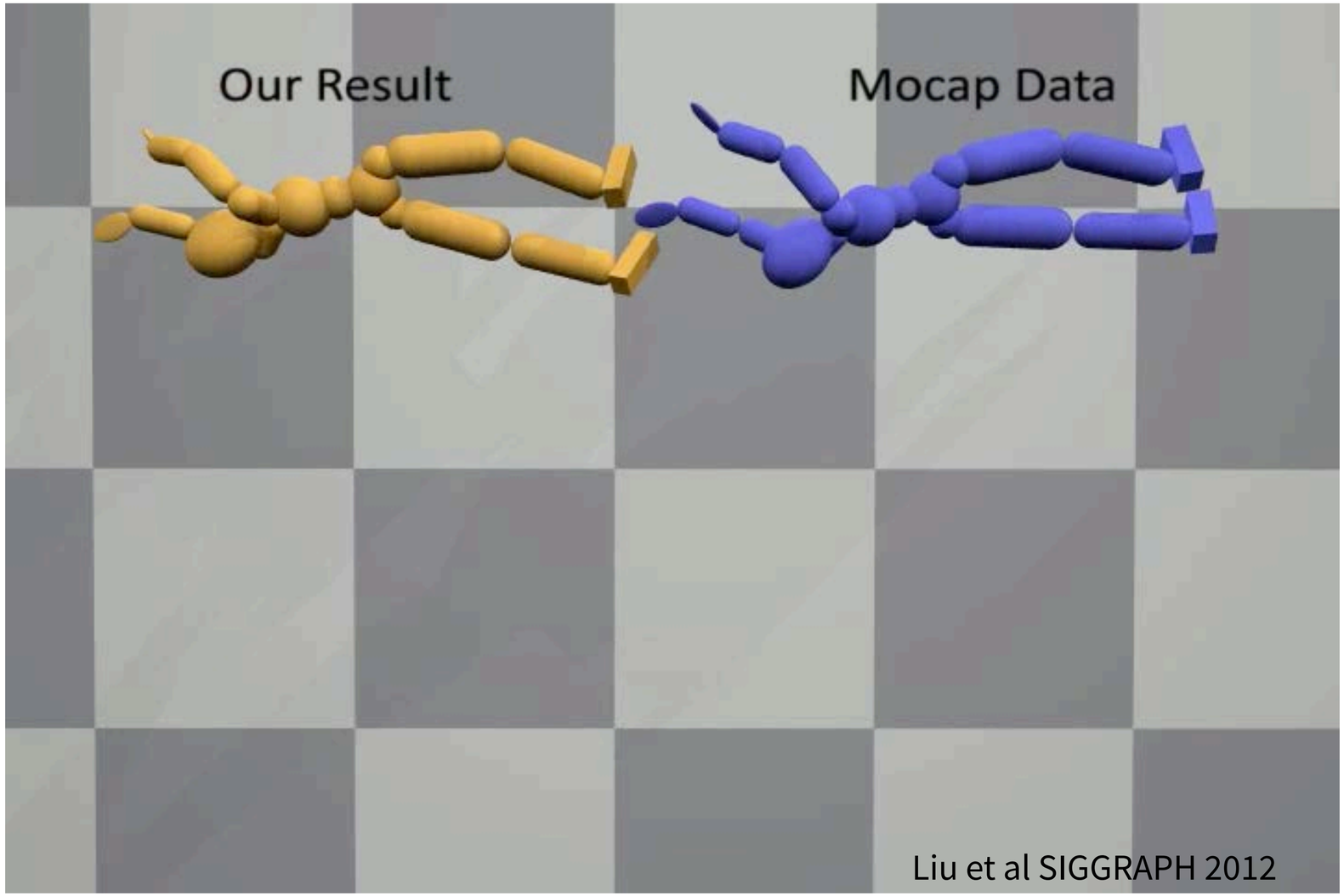


Control a dynamic system





Tracking human motion



Trajectory optimization

$$\min_{\boldsymbol{\tau},\boldsymbol{q}} \sum \boldsymbol{\tau}^2 + C_{task}(\boldsymbol{q})$$

subject to

$$\ddot{\boldsymbol{q}} = f_{skelDyn}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\tau})$$

$$\boldsymbol{\tau}_{low} \leq \boldsymbol{\tau} \leq \boldsymbol{\tau}_{high}$$

$$\boldsymbol{q}_{low} \leq \boldsymbol{q} \leq \boldsymbol{\tau}_{high}$$

Trajectory optimization

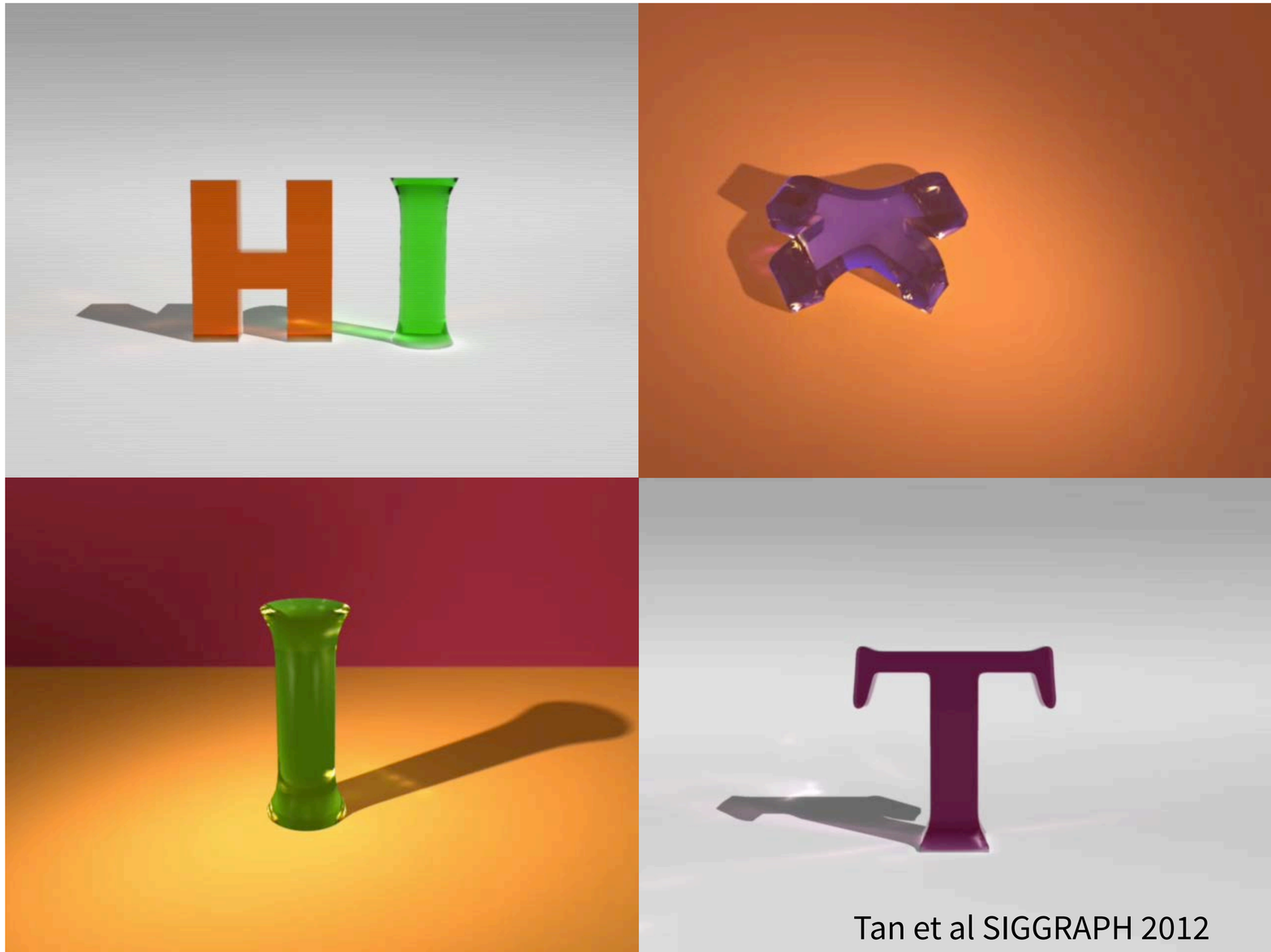
$$\min_{\boldsymbol{\tau},\boldsymbol{q}} \sum \boldsymbol{\tau}^2 + C_{task}(\boldsymbol{q})$$

subject to

$$\ddot{\boldsymbol{q}} = f_{skelDyn}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\tau})$$

$$\boldsymbol{\tau}_{low} \leq \boldsymbol{\tau} \leq \boldsymbol{\tau}_{high}$$

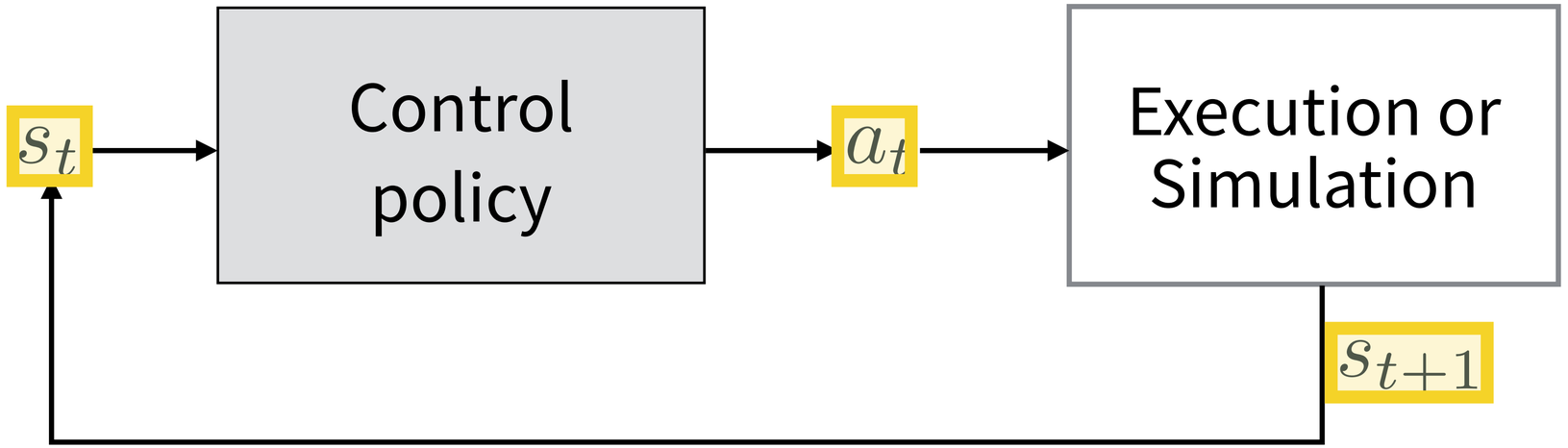
$$\boldsymbol{q}_{low} \leq \boldsymbol{q} \leq \boldsymbol{\tau}_{high}$$



Tan et al SIGGRAPH 2012

Reinforcement learning

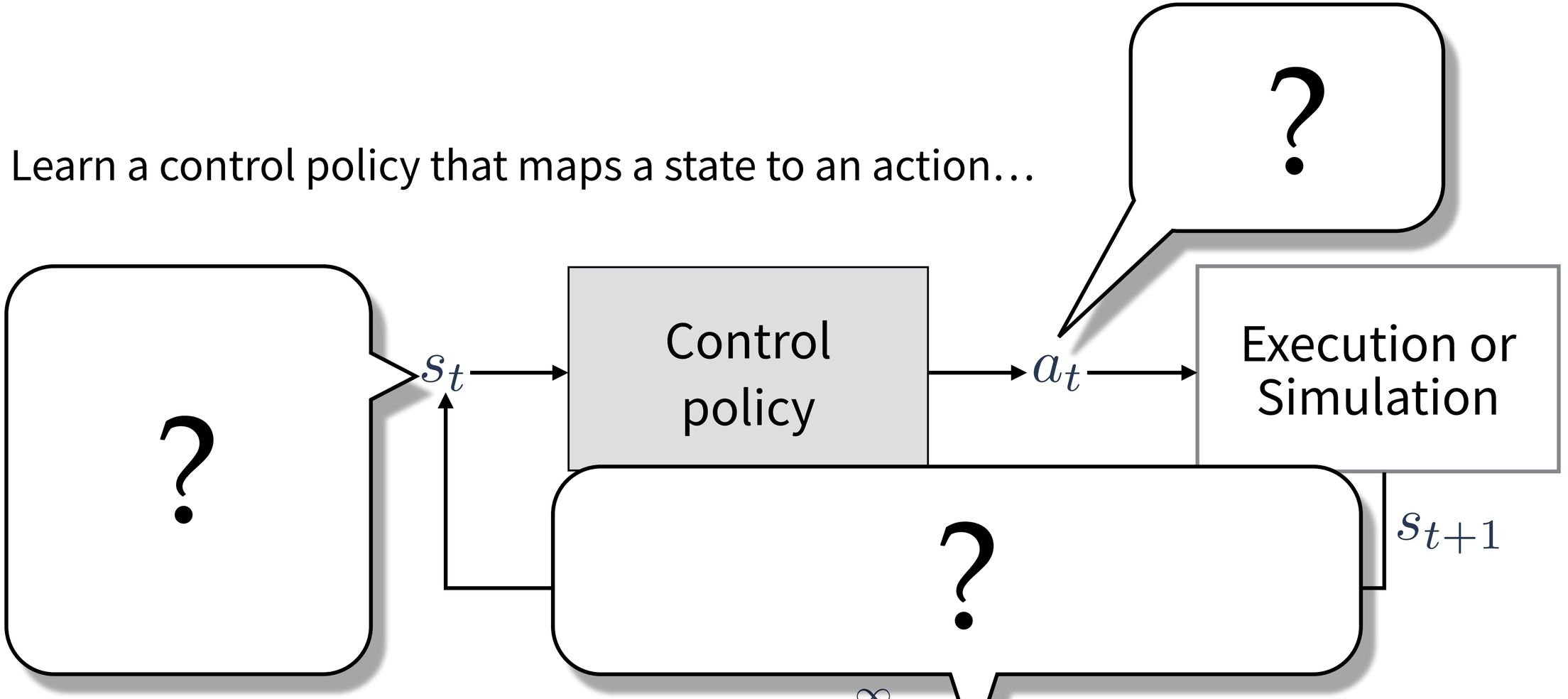
Learn a control policy that maps a state to an action…



…such that $\mathcal{J}(\tau) = \mathbb{E}_{\tau \sim p_\tau}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$ is optimized.

Reinforcement learning

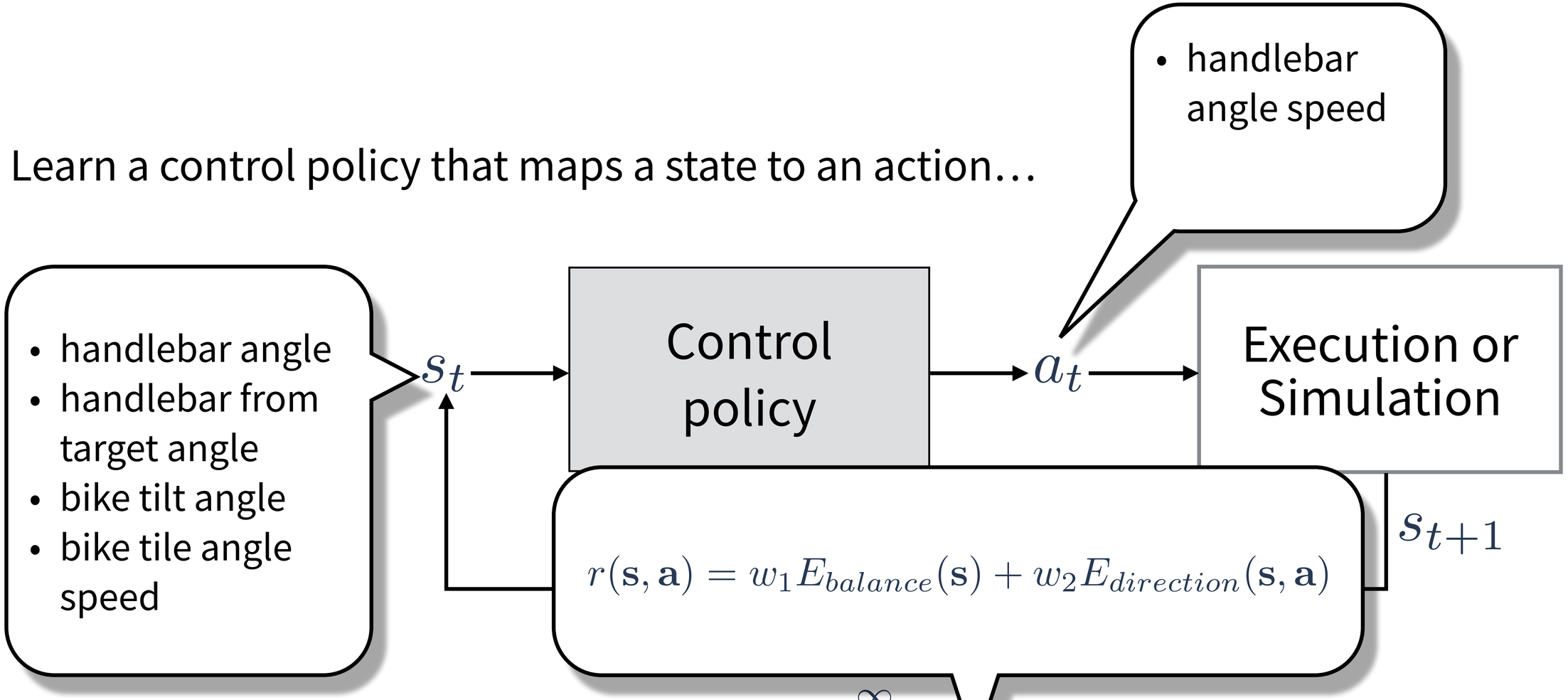
Learn a control policy that maps a state to an action…



…such that $\mathcal{J}(\tau) = \mathbb{E}_{\tau \sim p_\tau}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$ is optimized.

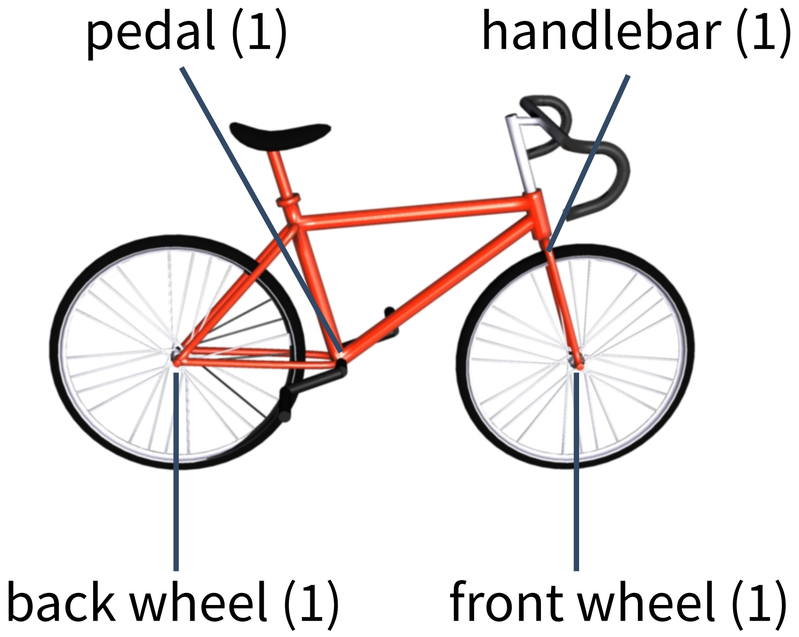
Reinforcement learning

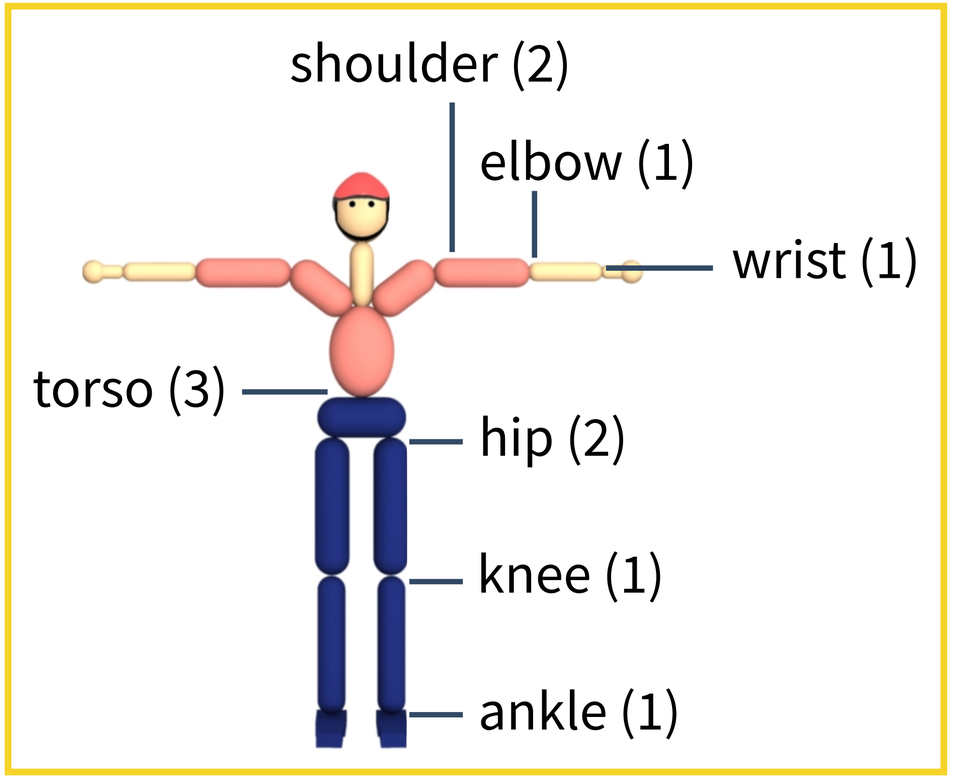
Learn a control policy that maps a state to an action…



…such that $\mathcal{J}(\tau) = \mathbb{E}_{\tau \sim p_\tau}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$ is optimized.

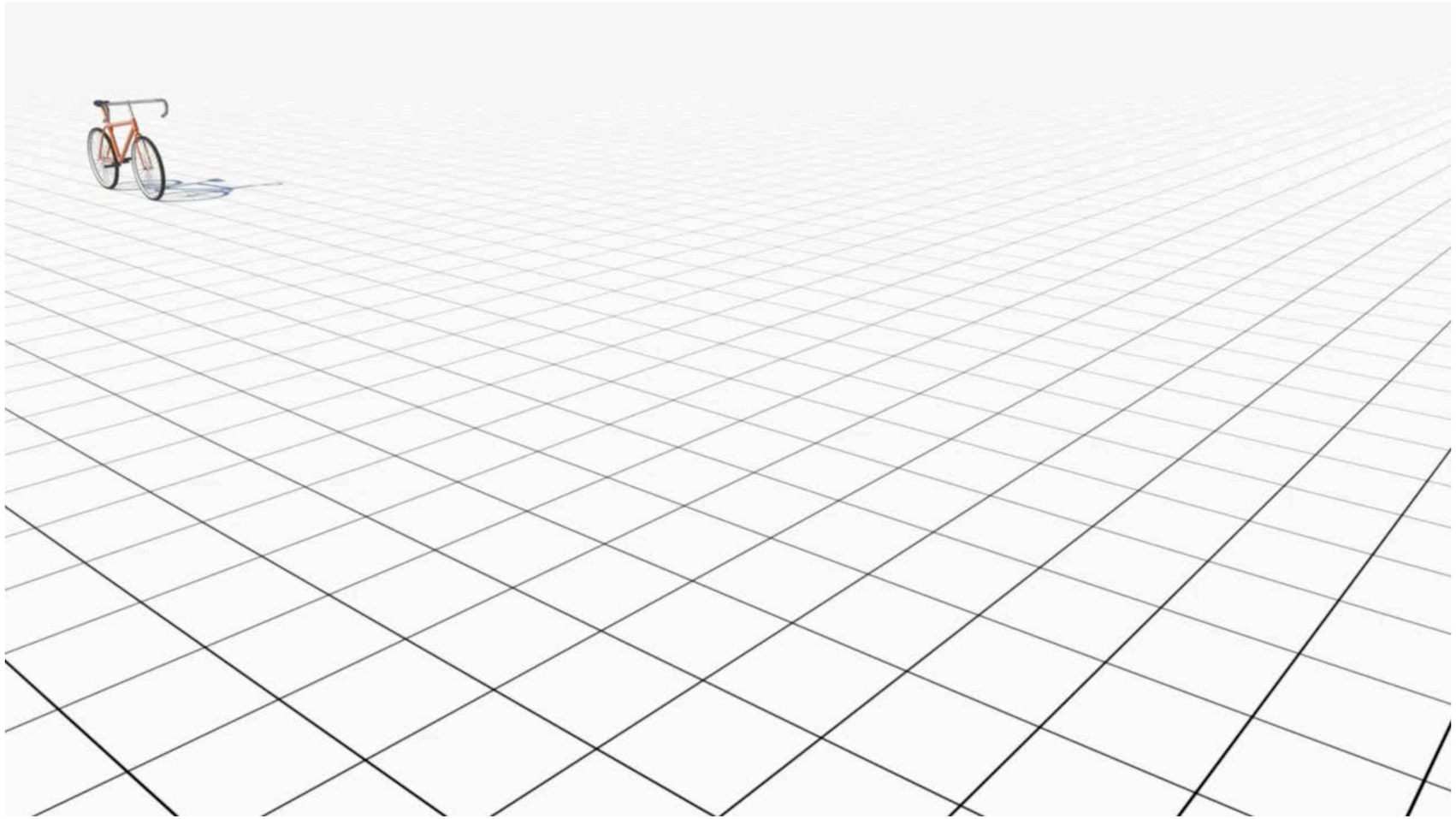
Reinforcement learning

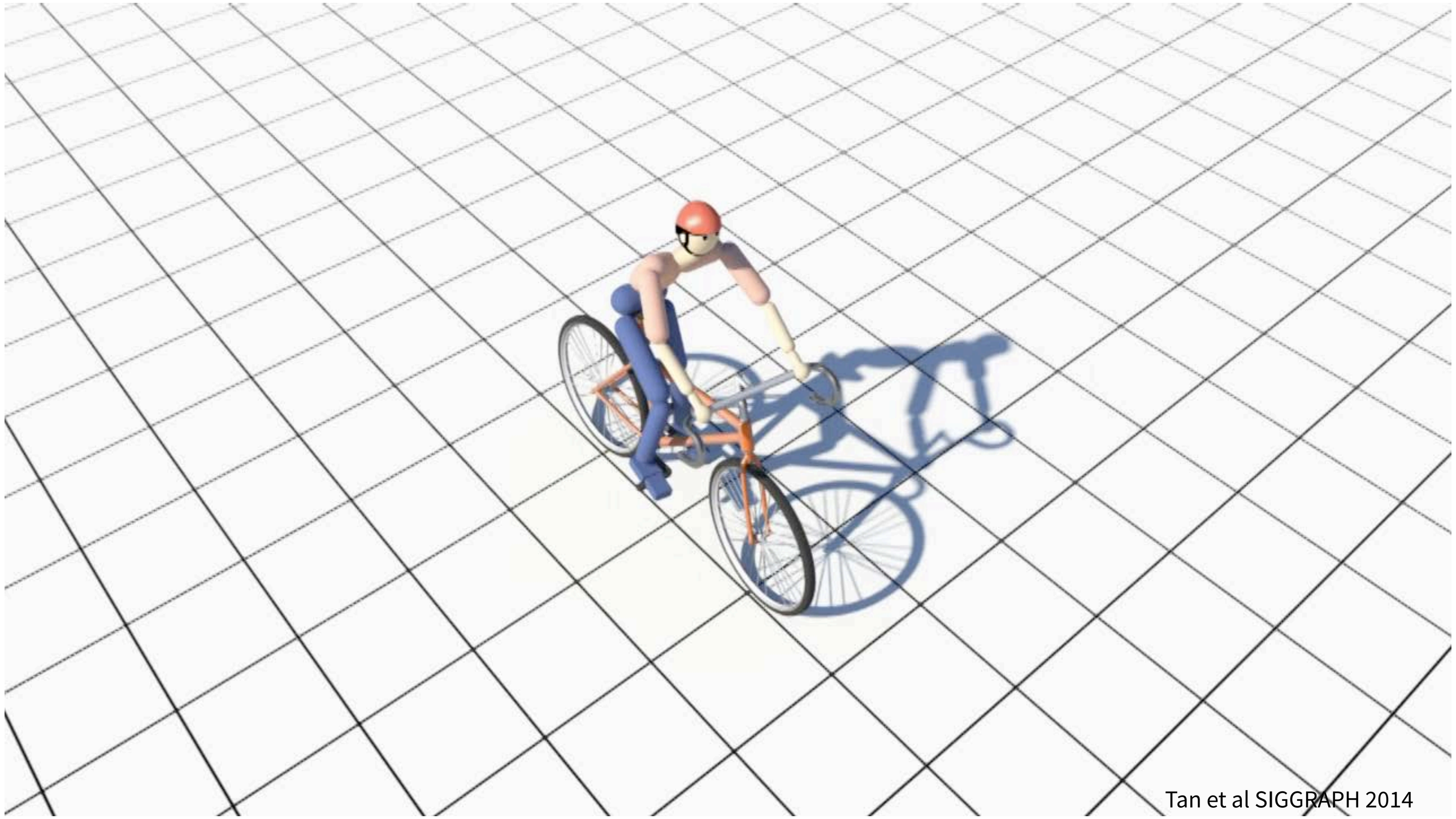


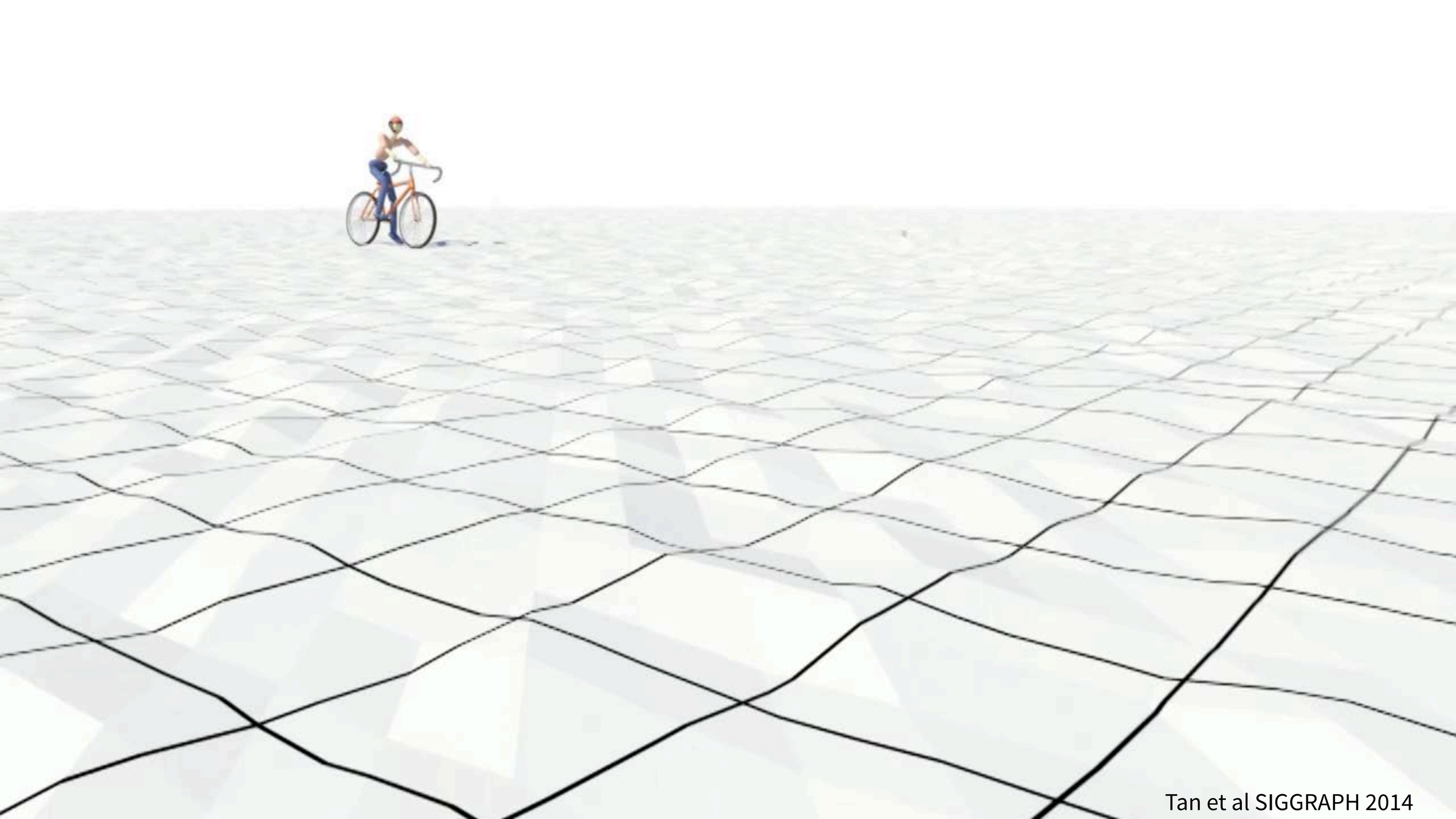


actuated degrees of freedom

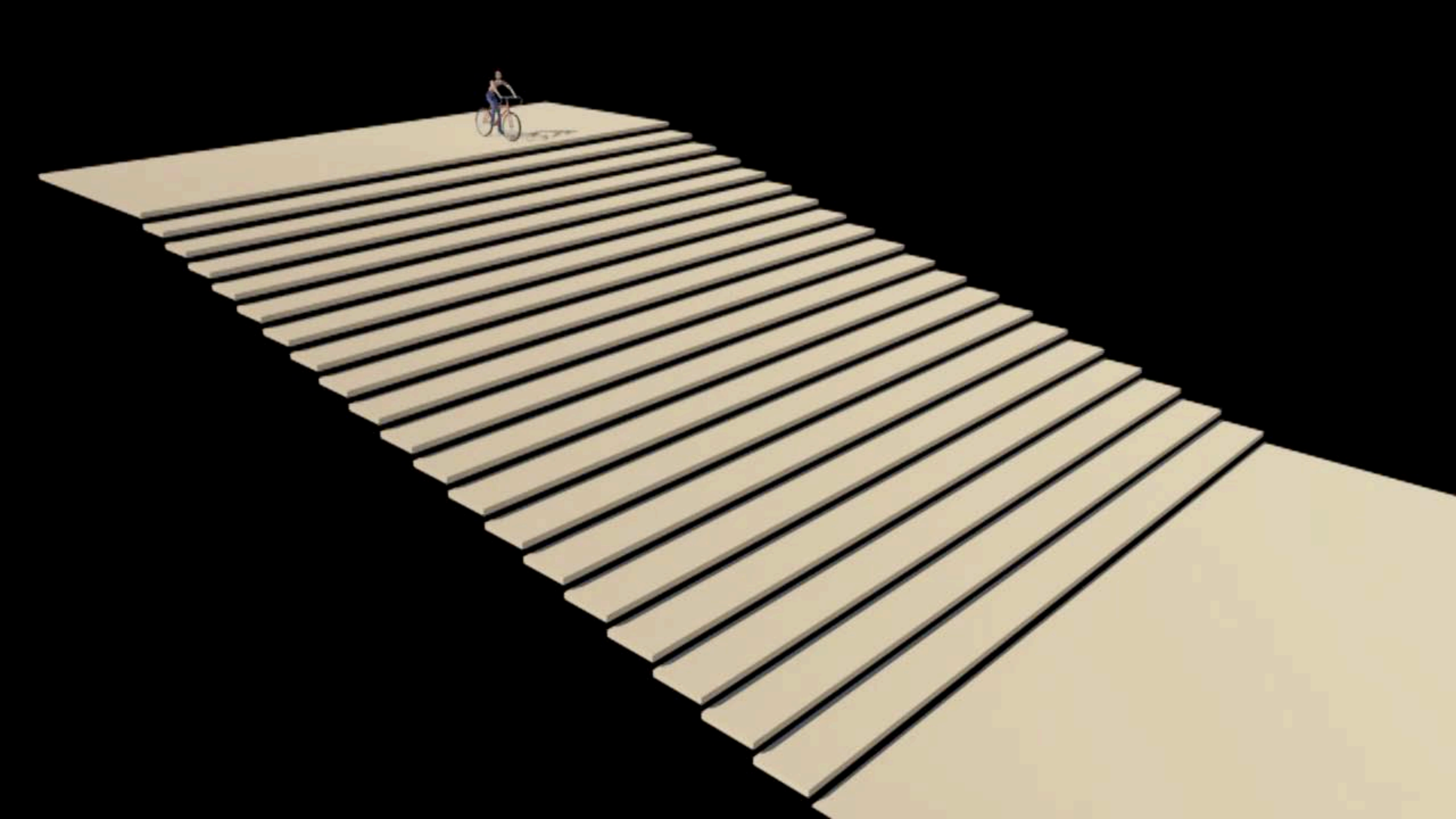
Uncontrolled dynamic system

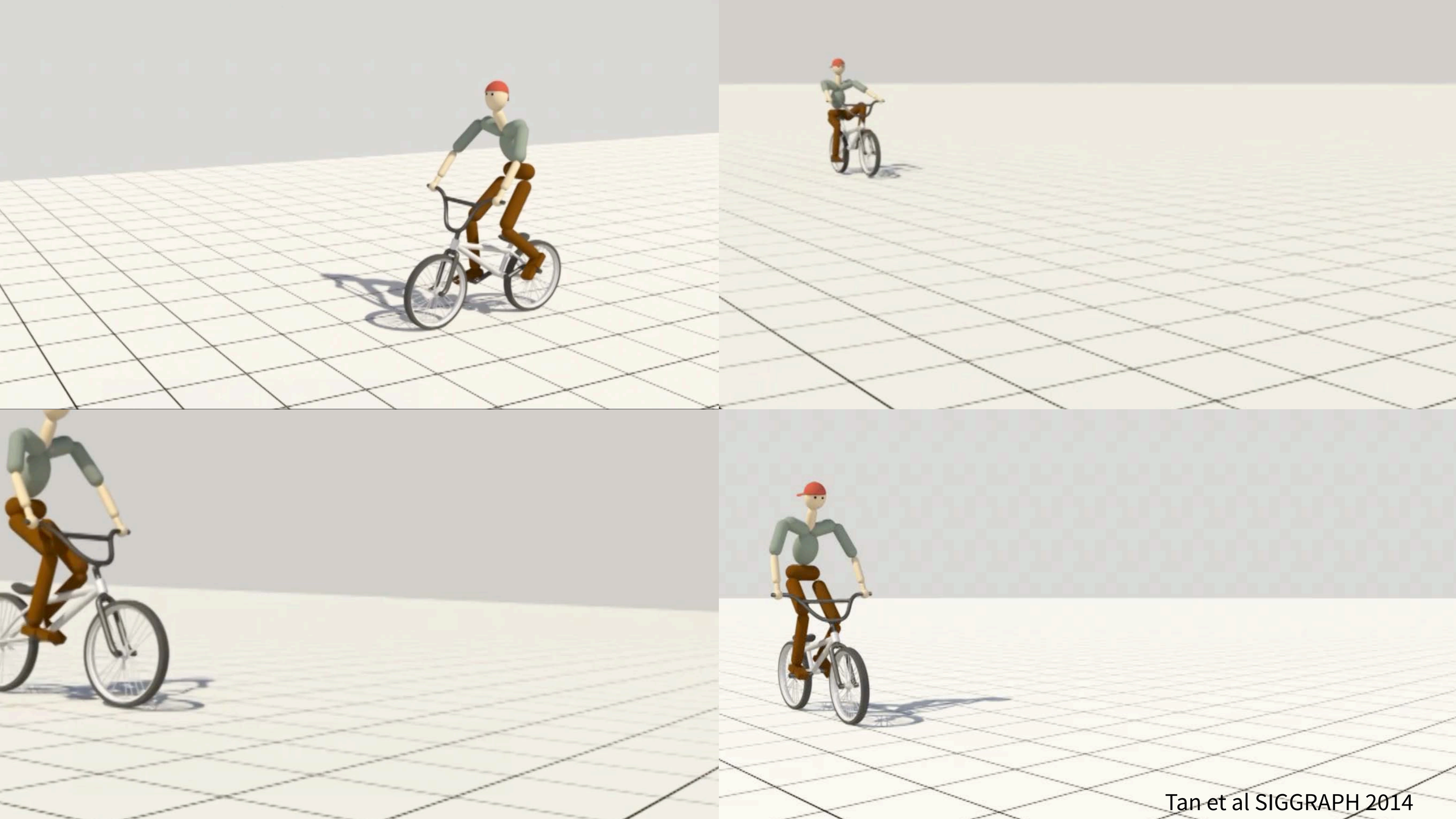



Tan et al SIGGRAPH 2014

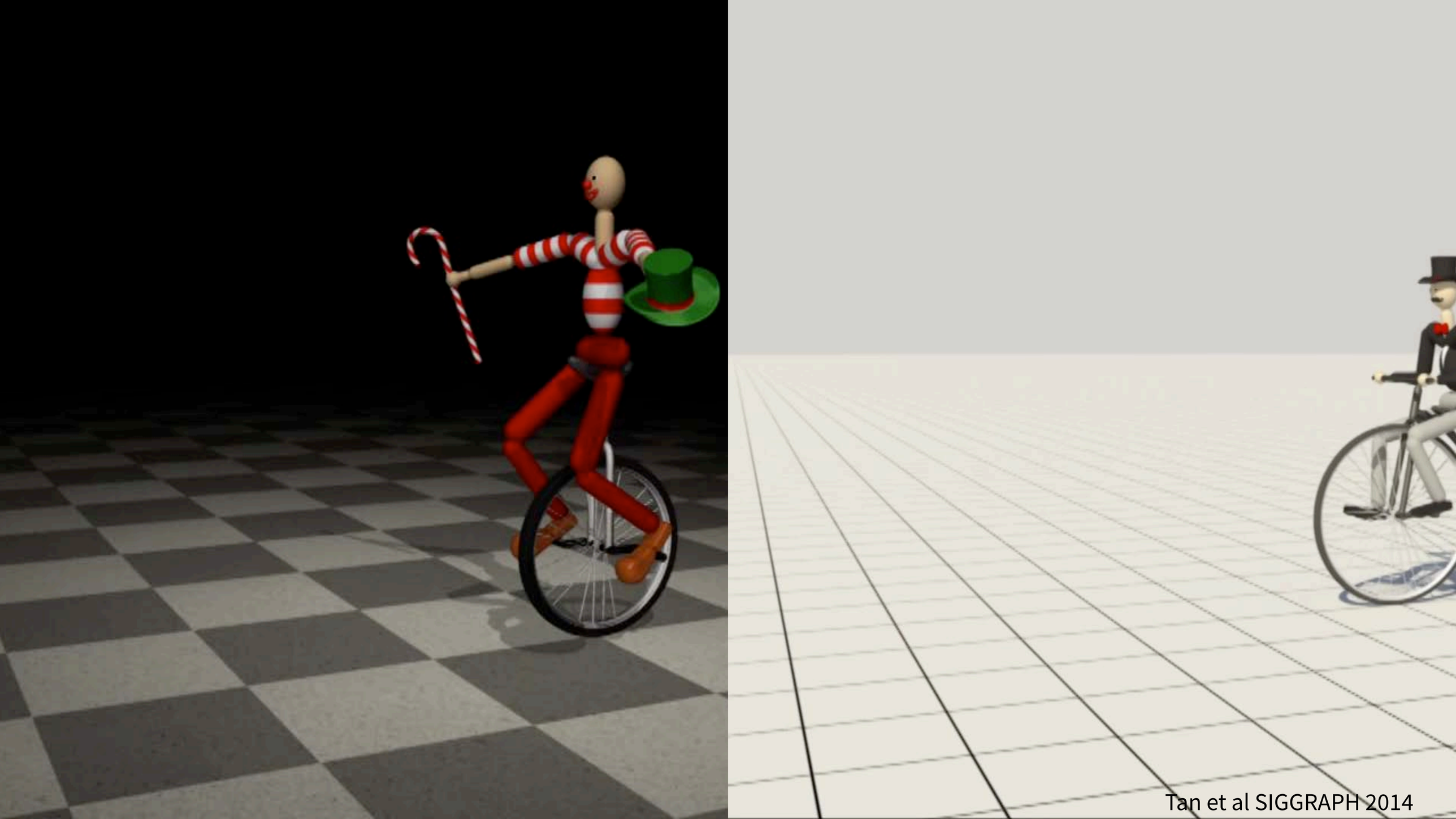


Tan et al SIGGRAPH 2014



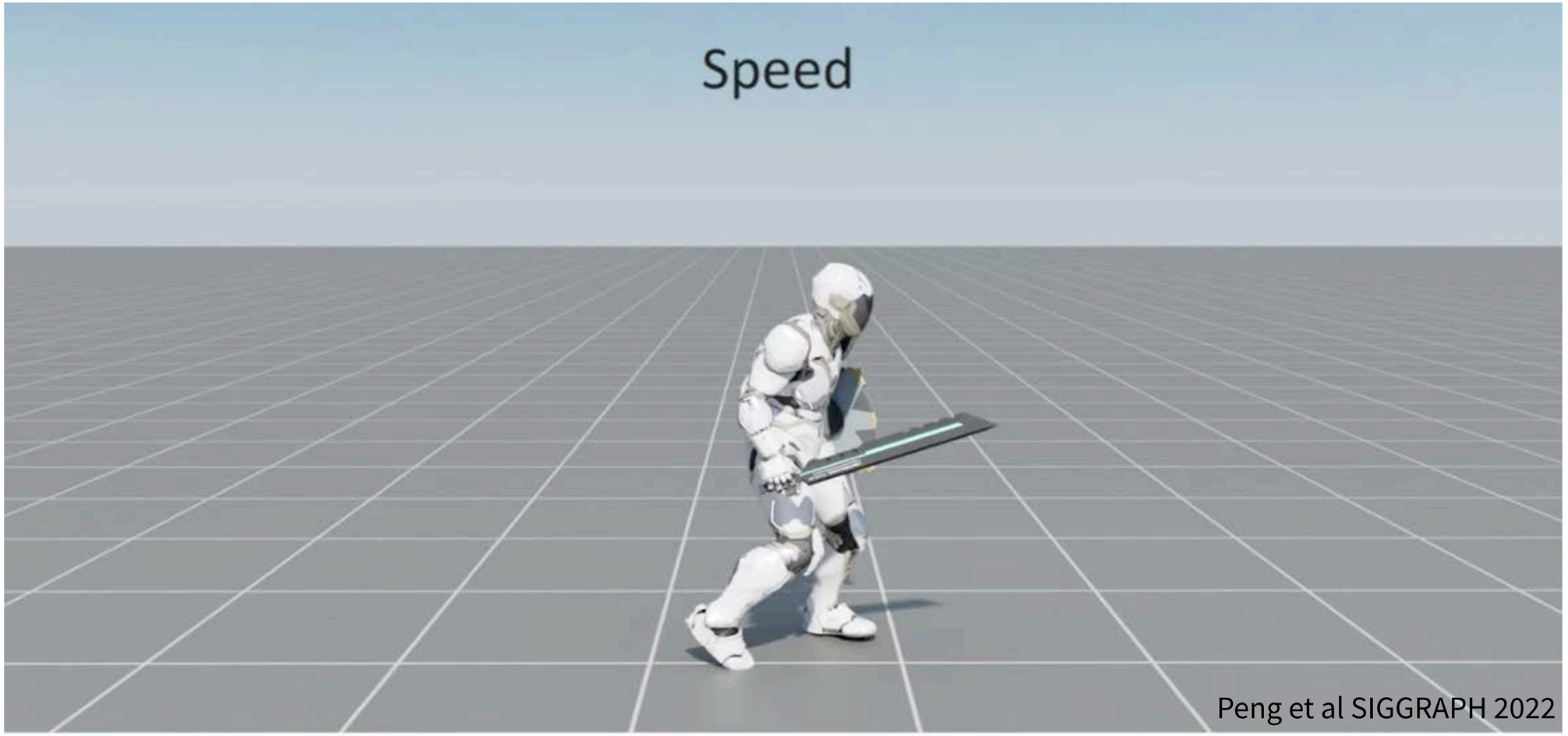


Tan et al SIGGRAPH 2014

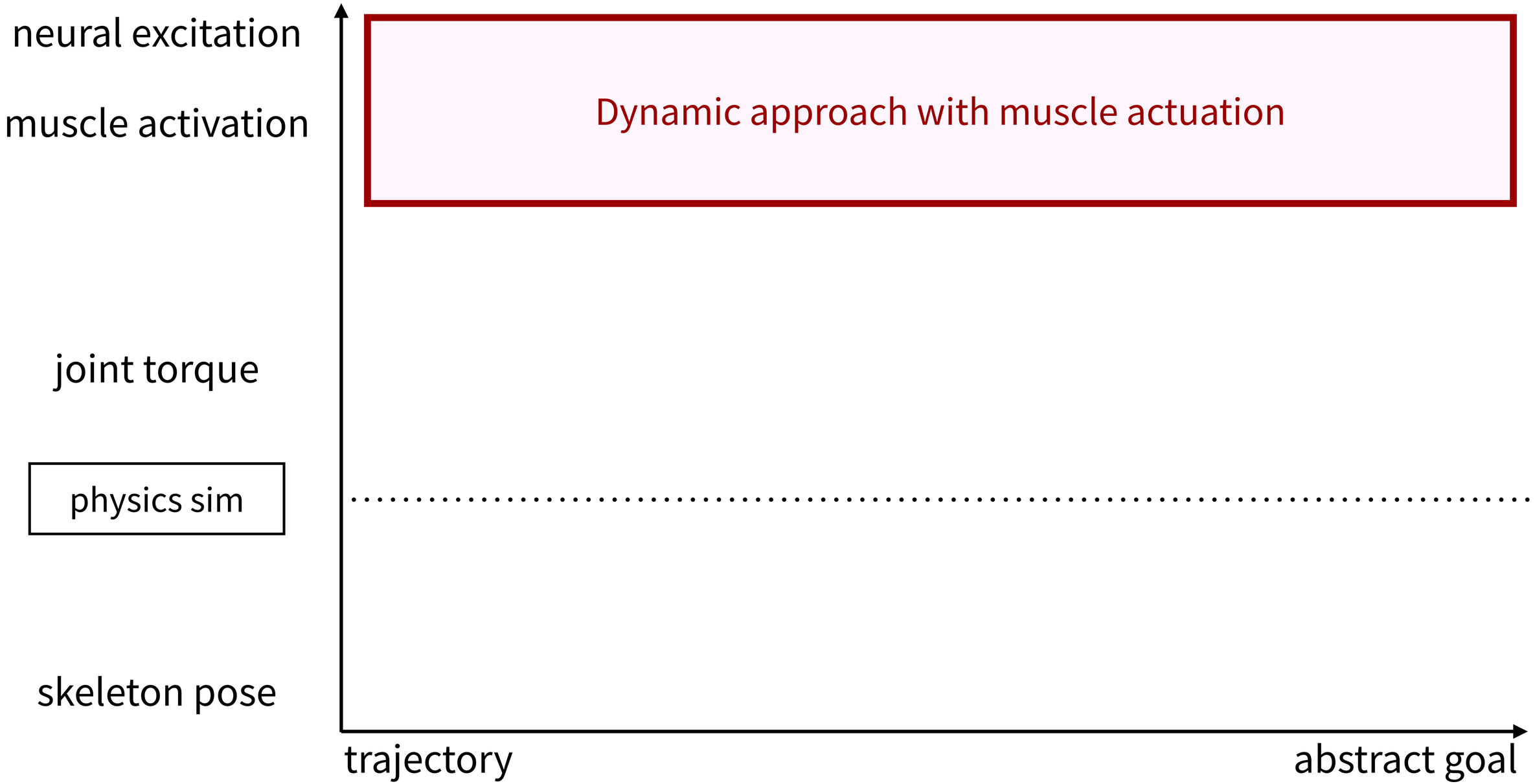


Tan et al SIGGRAPH 2014

Deep reinforcement learning



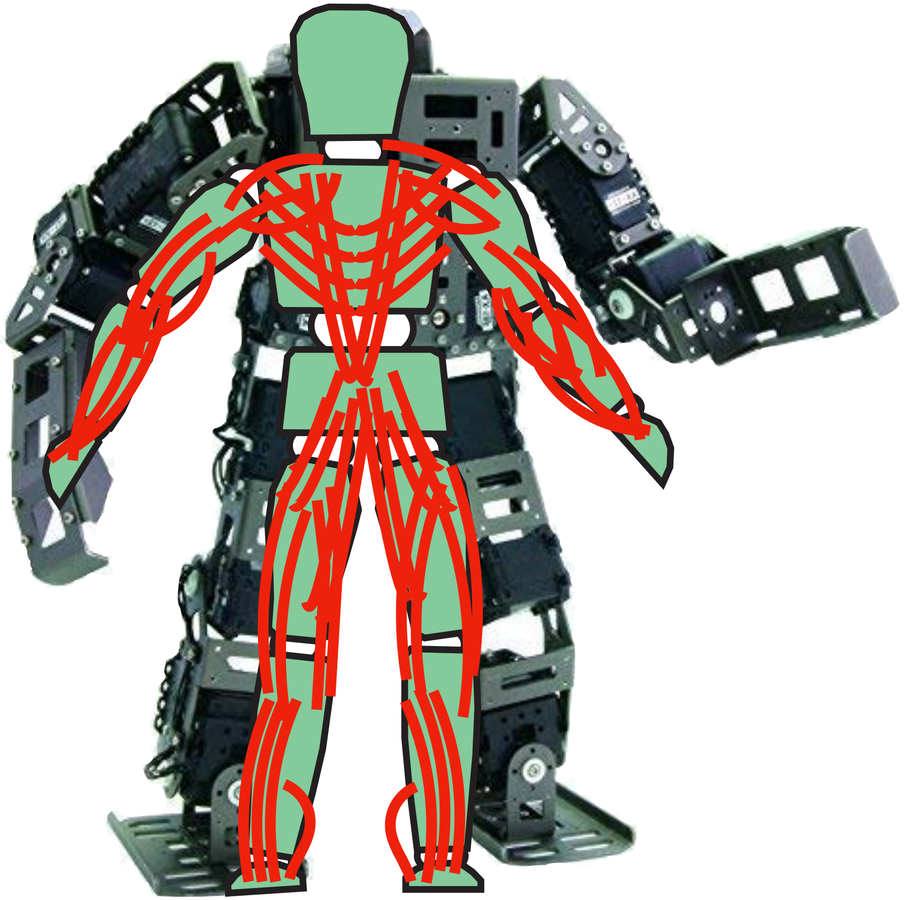
First, we have a target speed task,

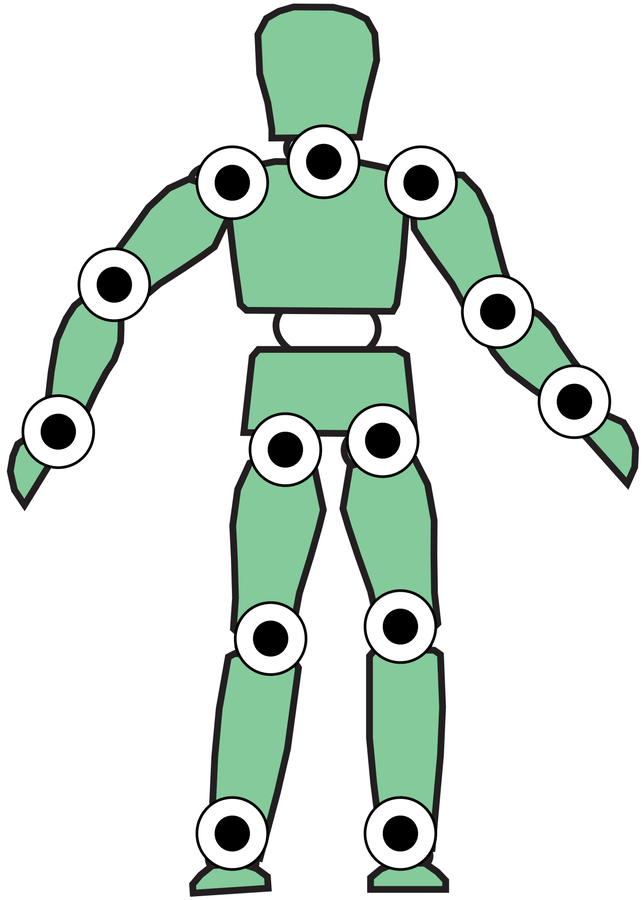

neural excitation
muscle activation
Dynamic approach with muscle actuation
joint torque
physics sim
skeleton pose
trajectory
abstract goal

Week 10: Final Week

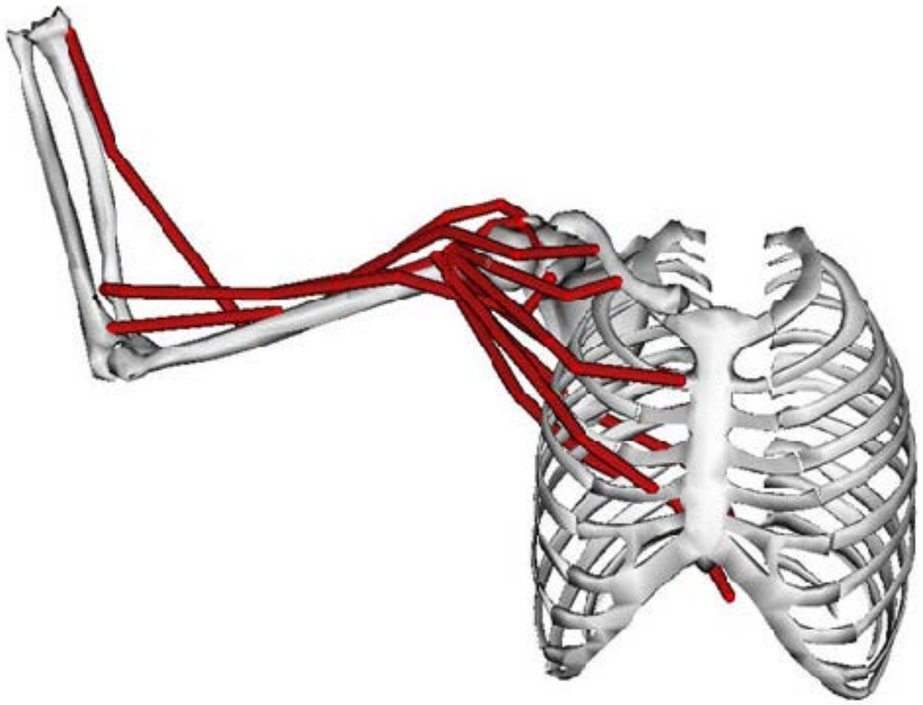
- Musculoskeletal animation
- Future directions in visual computing
- Guest lecture from the graphics industry (TBD)

Muscle activation



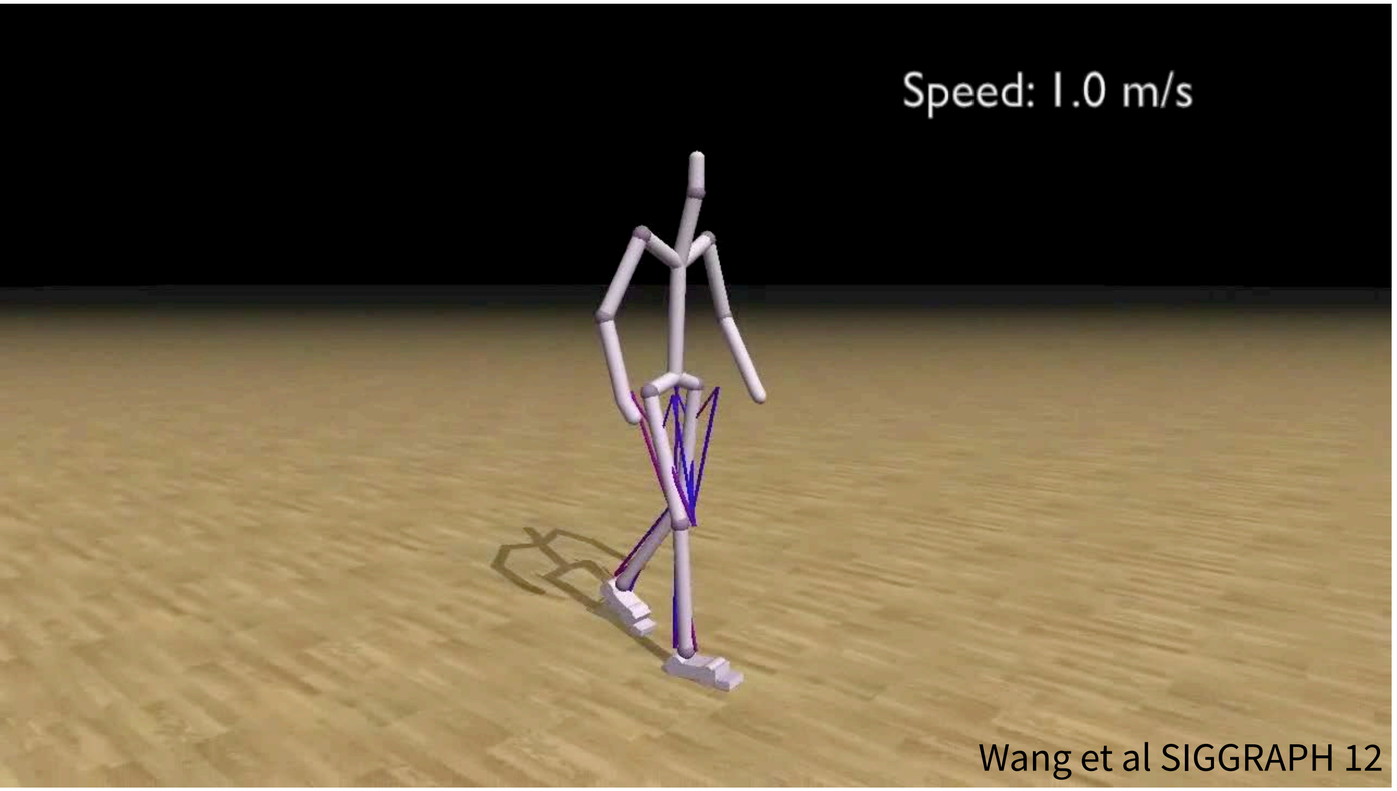


Muscle activation

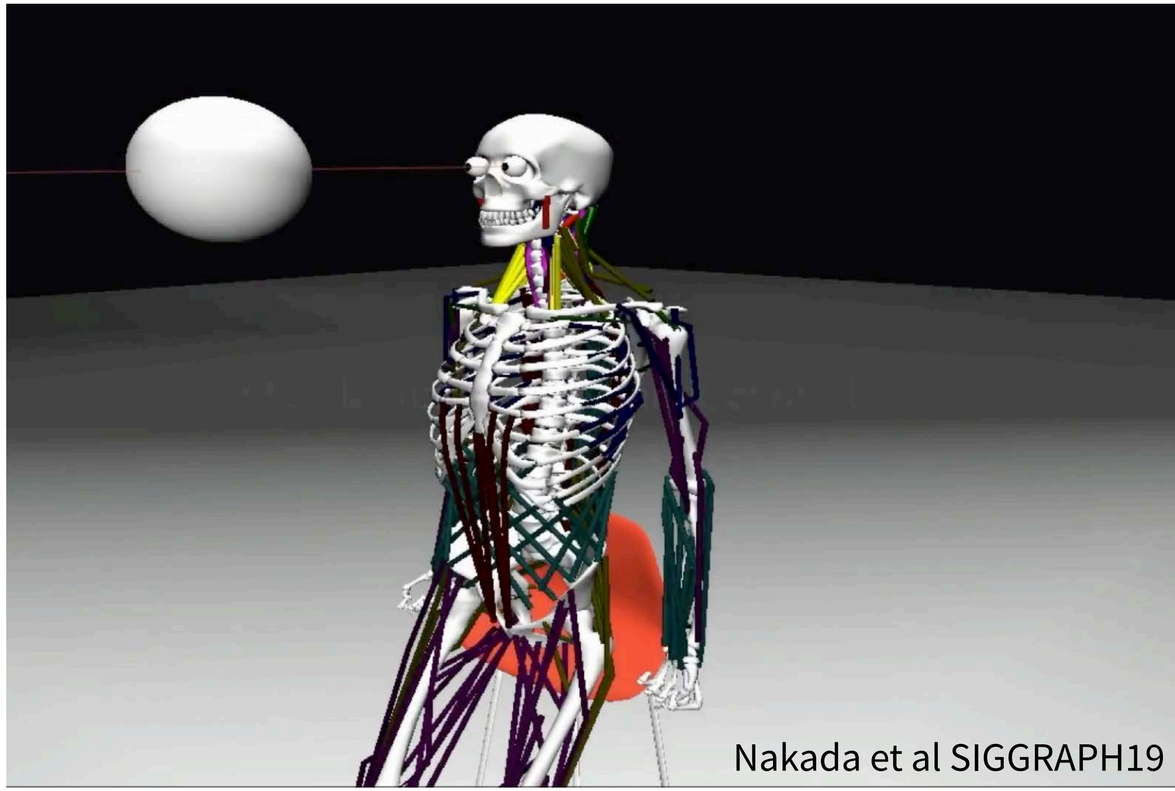


- Model and simulate dynamics of muscles and tendons, in addition to skeleton dynamics
- As muscles contract, they pull the attached bones to generate torques around joints
- Parameters for muscles and tendons can be obtained from biomechanics experiments
- Muscle activation becomes the control variables instead of joint torques

Control muscle activation



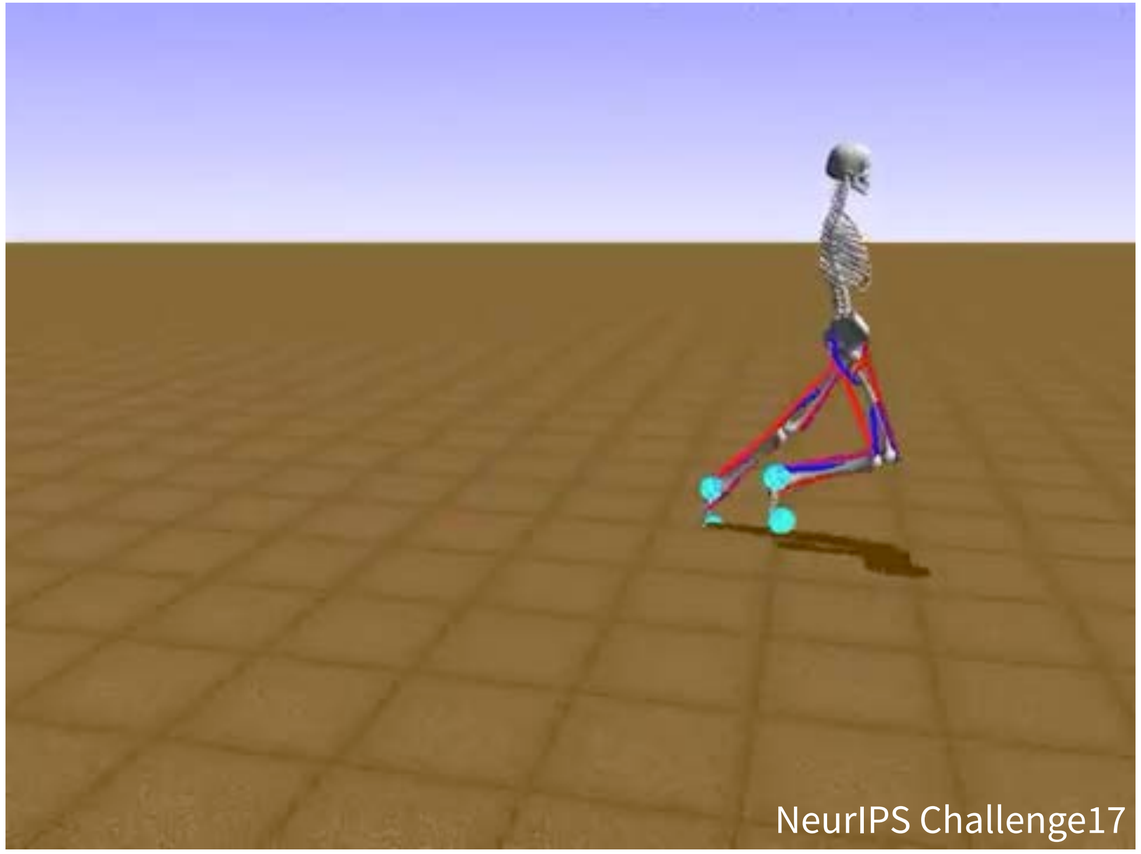
Wang et al SIGGRAPH 12



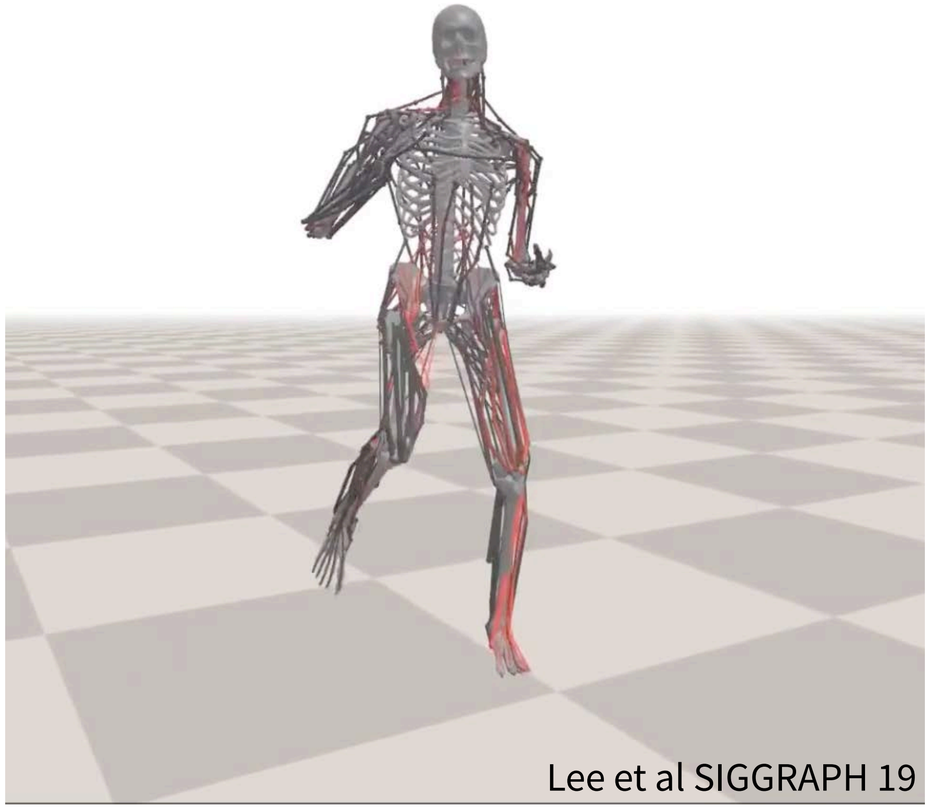
Nakada et al SIGGRAPH19

Learning muscle activation

Learning **without** reference motion



Learning **with** reference motion



Week 10: Final Week

- Musculoskeletal animation
- Future directions in visual computing
- Guest lecture from the graphics industry (TBD)

Visual Computing Track

Undergraduate Major in Computer Science
Stanford Univeristy

Visual Computing Track

Starting Fall 2022, the CS department will introduce a new Track, “Visual Computing”, which is a specialization at the intersection of AI/ML, Graphics, and Systems. “Visual Computing Track” takes a broader view than the former “Graphics Track” by giving students the increased flexibility to decide what areas of visual computing they want to focus on.

Creating, manipulating, and interpreting visual information (images, videos, 3D geometry, and simulated virtual worlds) is pervasive in modern computing. Today, we see “visual computing” play a huge role in applications ranging from entertainment (games, CGI for TV and film), digital imaging (smartphone cameras, AR/VR capture, scientific imaging), robotics (driver assistance, autonomous vehicles, health-care robotics), engineering, and commerce. The visual computing track aims to give students the background needed to master the fundamental techniques of visual computing and to advance the development of new visual computing techniques and applications.

You can find all the details about the Visual Computing Track here, including the course requirements, possible curriculum paths, changes to the former Graphics Track and more.

If you have any questions, please email Kayvon, Doug, or Karen.

Visual Computing Track

Principles of the Visual Computing track's design: (compared to the old graphics track)

- Take a broader view of visual computing than the former "Graphics" track by giving students the increased flexibility to decide what areas of visual computing they want to focus on, regardless of whether that focus is AI/ML heavy, "traditional graphics" heavy, or systems heavy.
- Make the track attractive to a broad range of students:
  - Reduce friction by providing students the ability to "start" the track within any quarter.
  - Allow track credit for AI/ML-centric methods courses, since those techniques play a big part in modern computer graphics. (We believe some AI track students would be better served as visual computing track students since they are already focused on applying AI/ML to visual computing applications.)
  - Design an elective system that gives students the ability to get track credit for courses that are highly relevant to modern visual computing, and that they are likely to take anyway.

Visual Computing Track

Track Requirements

TL;DR

In addition to the standard CS core, students must take seven courses from the offerings listed in Tier 1, Tier 2, or Tier 3 of the track. Of these seven:

- At least two courses must be from Tier 1 (Visual Computing Core)
- At least five courses must be from Tier 1 or 2 (Visual Computing Core or Depth)
- One of the courses used to satisfy the Tier 3 (Visual Computing Elective) requirement can be any course from the general CS elective pool.

Visual Computing Track

Tier 1: Fundamentals Courses

Students are required to take at least two Tier 1 courses (in any order).

- CS248A - Computer Graphics: Rendering, Geometry, and Image Manipulation (Winter)
- CS248B - Computer Graphics: Animation, Simulation, Optimization (Fall)
- CS231N - Deep Learning for Computer Vision (Spring)

Faculty rationale for Tier 1's design:

- We want students to have technical foundations in at least one "modeling the visual world" course (248-level), but the track is not opinionated about whether that should be focused on rendering/image processing or animation/simulation/optimization.
- We wanted to present a schedule where one of the fundamentals courses is offered each quarter. Therefore, students can "start" the Tier 1 part of the track with a fundamentals course at any time.
- We anticipate that two courses listed under Tier 3 electives (CS148 and CS131) will serve as important "feeder" courses to the visual computing track. Students will be able to get track credit for those courses, but not Tier 1 credit. Please see the comments in the Tier 3 section.

Visual Computing Track

Tier 2: Depth Courses

Tier 2 courses give students the opportunity to conduct detailed study of visual computing topics. Tier 2 courses provide technical depth that builds upon the fundamentals established in Tier 1.

Students must take at least two Tier 2 courses. Three Tier 2 courses are required if a student only takes two courses from Tier 1.

- CS205L - Continuous Mathematical Methods with an Emphasis on Machine Learning (Winter)
- CS223A - Introduction to Robotics (Winter)
- CS231A - Computer Vision, From 3D Reconstruction to Recognition (Winter)
- CS233 - Geometric and Topological Data Analysis (Spring)
- CS348B - Computer Graphics: Image Synthesis Techniques (Spring)
- CS348C - Computer Graphics: Animation and Simulation (Winter)
- CS348E - Character Animation: Modeling, Simulation, and Control of Human Motion (Spring)
- CS348K - Visual Computing Systems (Spring)
- CS348I - Computer Graphics in the Era of AI (Fall)
- CS348N - Neural Models for 3D Geometry (Winter)
- CS448I - Computational Imaging (also EE 367) (Winter)
- EE267 - Virtual Reality (Spring)

Visual Computing Track

Tier 3: Track Electives

Students may count up to two Tier 3 courses to the seven course requirement of the track. Students may also count up to one 1 general CS course as a Tier 3 course toward this requirement.

Tier 3 electives aim to give students the opportunity to count a wide variety of courses toward the track. These might be courses that are “funnel courses” that introduce fundamentals, but don’t have the technical or implementation rigor to receive Tier 1 classification, courses that are not specific to visual computing but teach relevant techniques, or advanced special topics in visual computing.

Introductory courses that serve to bring students into the track:

- CS148 - Introduction to Computer Graphics and Imaging
- CS131 - Computer Vision: Foundations and Applications

Courses teaching fundamental techniques of high relevance to advanced visual computing methods and applications:

- CS149 - Parallel Computing
- CS221 - Artificial Intelligence: Principles and Techniques
- CS229 - Machine Learning
- CS230 - Deep Learning
- CS236 - Deep Generative Models
- CS236G - Generative Adversarial Networks (GANs)
- EE261 - The Fourier Transform and its Applications

Advanced special topics:

- CS331B - Interactive Simulation for Robot Learning (Spring)
- CS448B - Data Visualization
- CS448M - Making Making Machines for Makers
- CS448Z - Physically Based Animation and Sound

Visual Computing Track

Example Student Paths

Here is the list of Tier 1 and Tier 2 courses sorted and **colored** by the quarter they are offered.

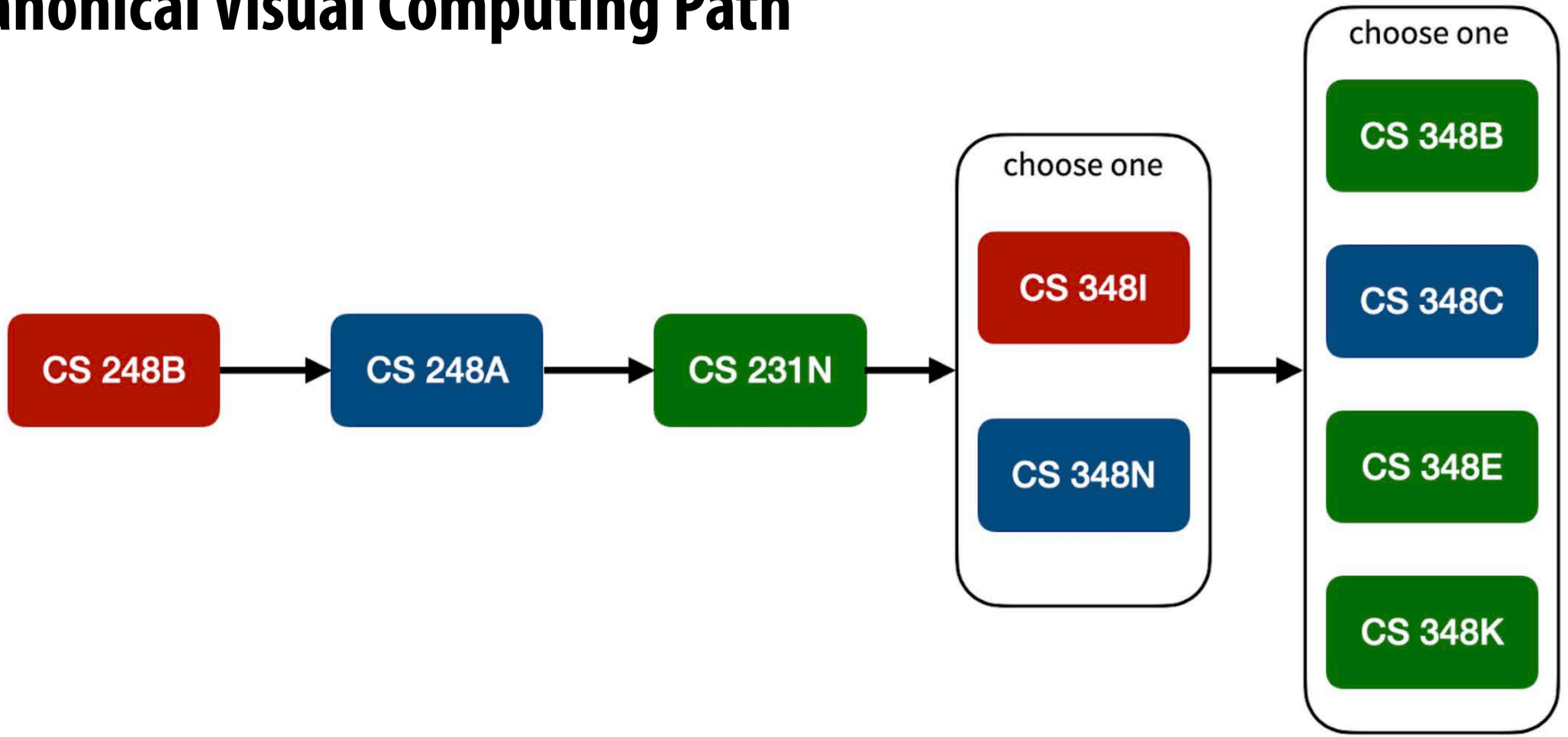
Autumn	Winter	Spring
• CS 248B: Computer Graphics: Animation, Simulation, Optimization • CS 348I: Computer Graphics in the Era of AI	• CS 248A: Computer Graphics: Rendering, Geometry, Image Manipulation • CS 205L: Continuous Mathematical Methods • CS 223A: Introduction to Robotics • CS 231A: Computer Vision, from 3D Reconstruction to Recognition • CS 348C: Animation and Simulation • CS 348N: Neural Models for 3D geometry • CS 448I: Computational Imaging (EE367)	• CS 231N: Deep Learning for Computer Vision • CS233: Geometric and Topological Data Analysis • EE 267: Virtual Reality • CS 348B: Image Synthesis Techniques • CS 348E: Character Animation • CS 348K: Visual Computing Systems

Students can create different paths to satisfy Tier 1 and Tier 2 requirements based on their own interests. Below we show some examples of curriculum paths:

Visual Computing Track

NEW!

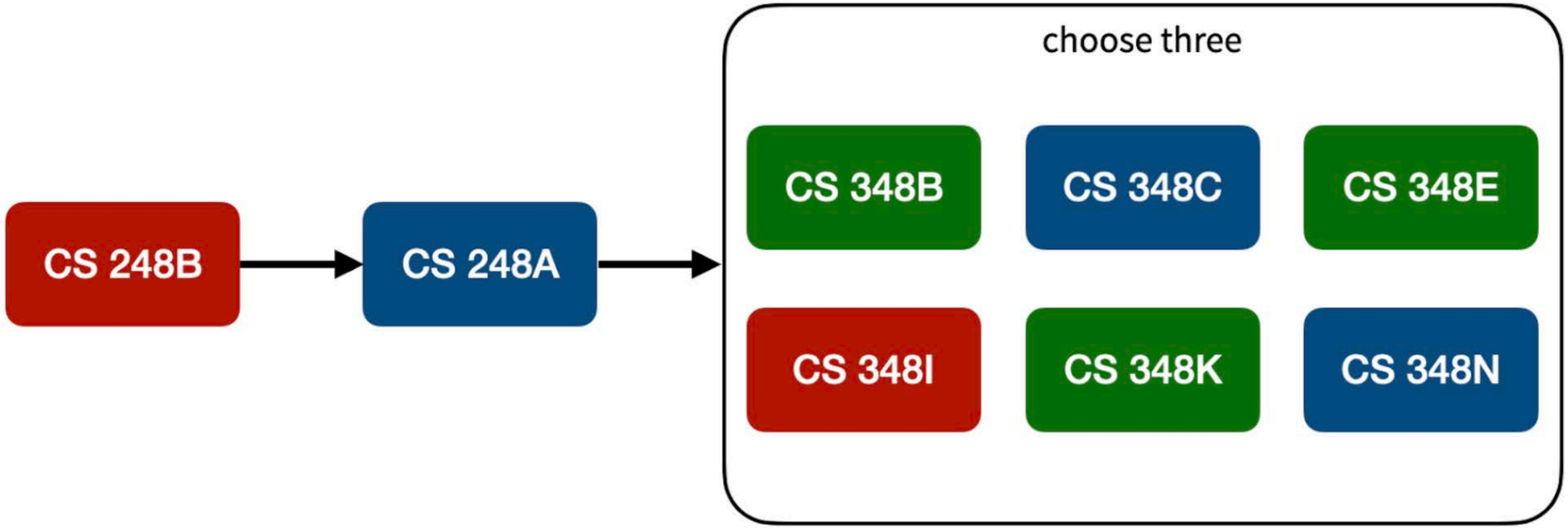
Canonical Visual Computing Path



Visual Computing Track

NEW!

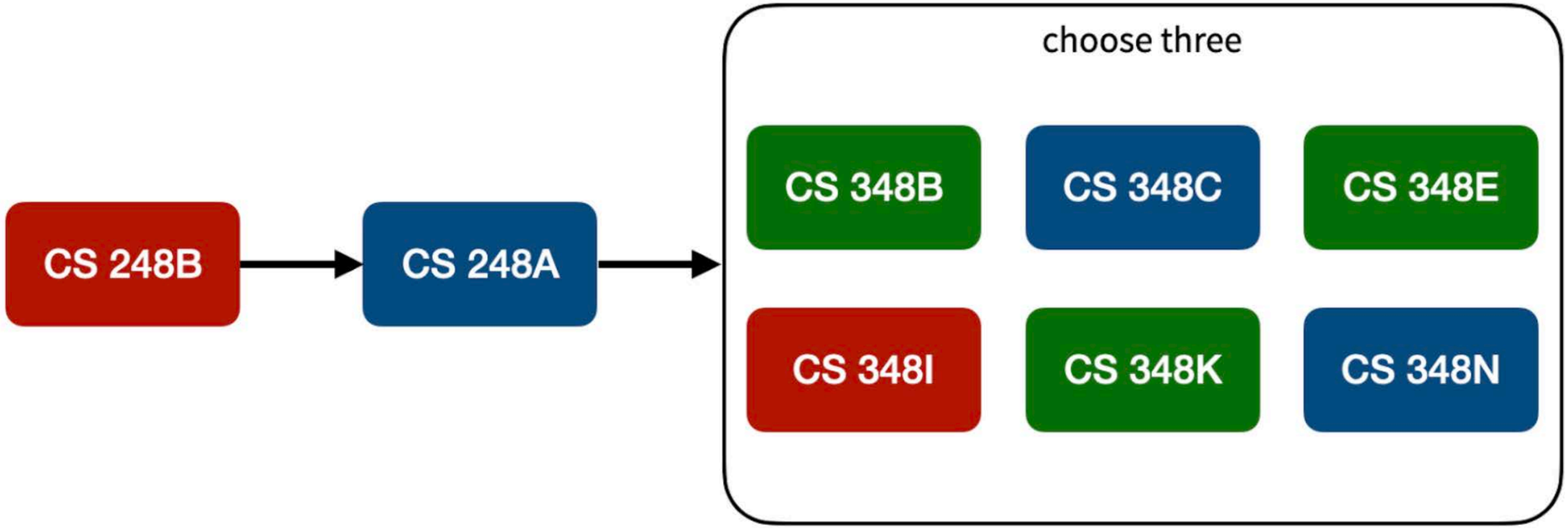
Traditional Computer Graphics Path



Visual Computing Track

NEW!

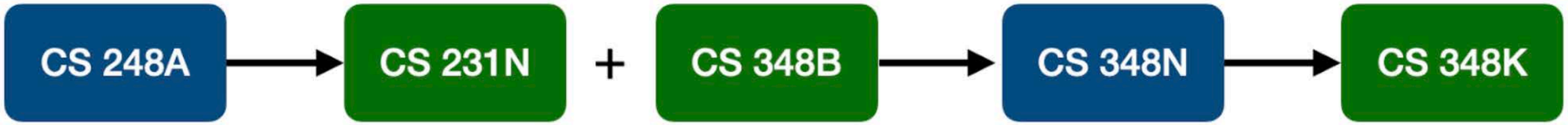
Traditional Computer Graphics Path



Visual Computing Track

Rendering/Geometry Specialization Paths

or



Visual Computing Track

Animation/Robotics Specialization Path