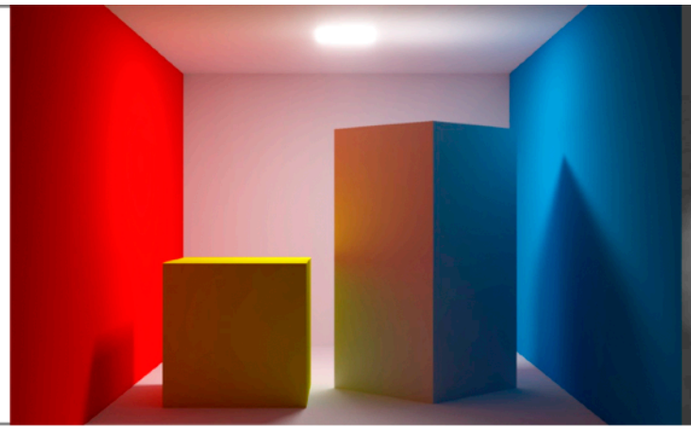
Introduction to Computer Graphics

GAMES101, Lingqi Yan, UC Santa Barbara

Lecture 22:
Animation (cont.)






Announcements

- Final project: extended by 1 week
- Homework 7: 175 submissions in total
  - Resubmission is welcomed
  - May decrease its weight in your final score
- Homework 8 will be released soon today
- Course certification with my signature
  - Will be sent out in electronic version after final project and resubmissions

  - Sign up for “Certification Request” (like a homework)
- Last lecture of this course :(

Today

Single particle simulation

- Explicit Euler method
- Instability and improvements

Rigid body simulation

Fluid simulation

Advertisements!

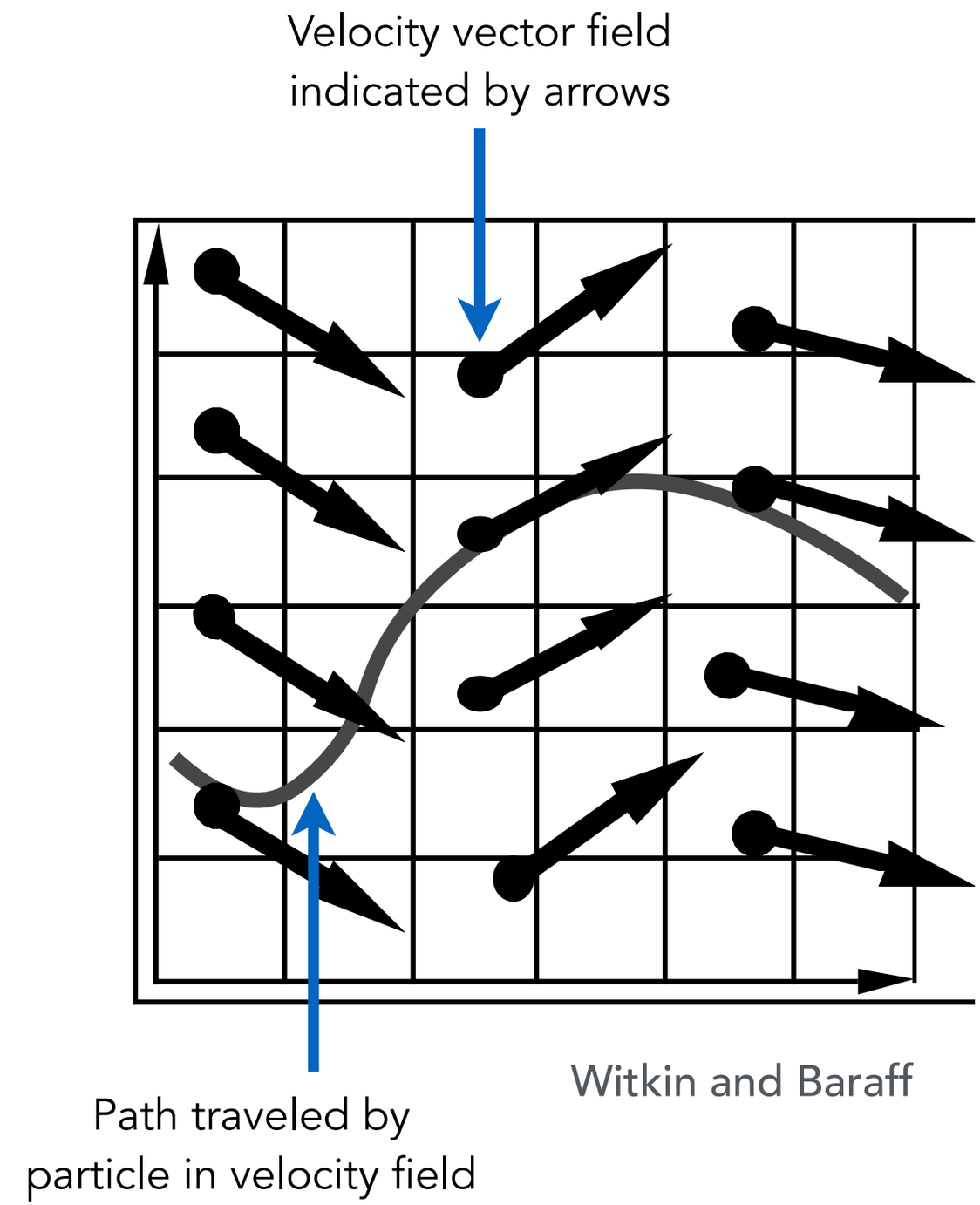
Single Particle Simulation

First study motion of a single particle

- Later, generalize to a multitude of particles

To start, assume motion of particle determined by a velocity vector field that is a function of position and time:

$$v(x,t)$$



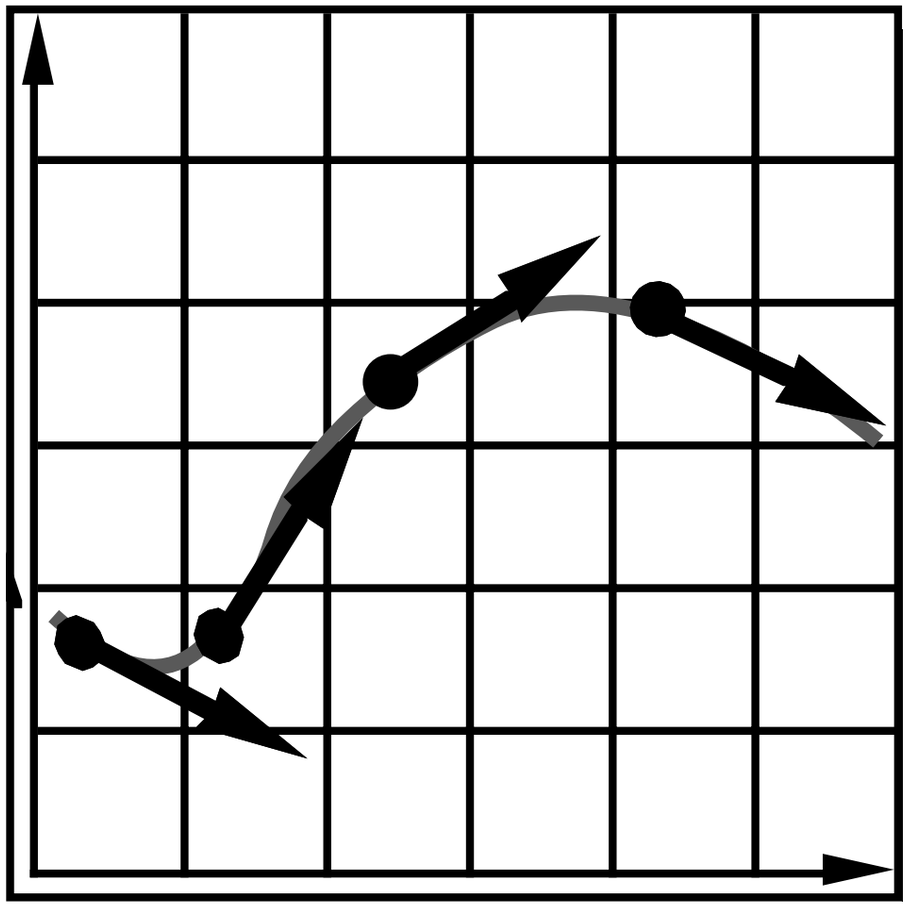
Ordinary Differential Equation (ODE)

Computing position of particle over time requires solving a first-order ordinary differential equation:

$$\frac{dx}{dt} = \dot{x} = v(x,t)$$

"First-order" refers to the first derivative being taken.

"Ordinary" means no "partial" derivatives, i.e. x is just a function of t

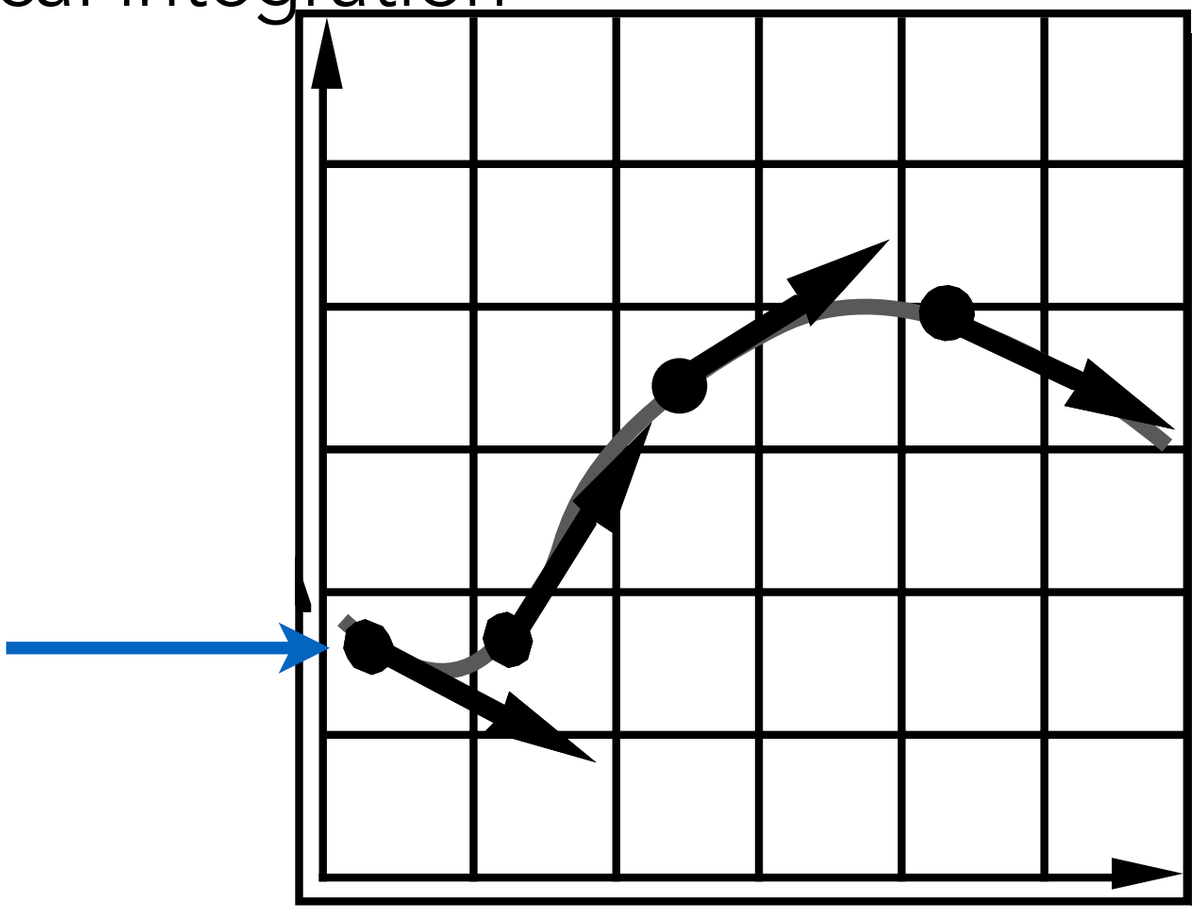


Witkin and Baraff

Solving for Particle Position

We can solve the ODE, subject to a given initial particle position x_0, by using forward numerical integration

Starting position x_0



Witkin and Baraff

Euler's Method

Euler's Method (a.k.a. Forward Euler, Explicit Euler)

- Simple iterative method
- Commonly used
- Very inaccurate
- Most often goes unstable

$$\boldsymbol{x}^{t+\Delta t} = \boldsymbol{x}^{t} + \Delta t\, \dot{\boldsymbol{x}}^{t}$$

$$\dot{\boldsymbol{x}}^{t+\Delta t} = \dot{\boldsymbol{x}}^{t} + \Delta t\, \ddot{\boldsymbol{x}}^{t}$$

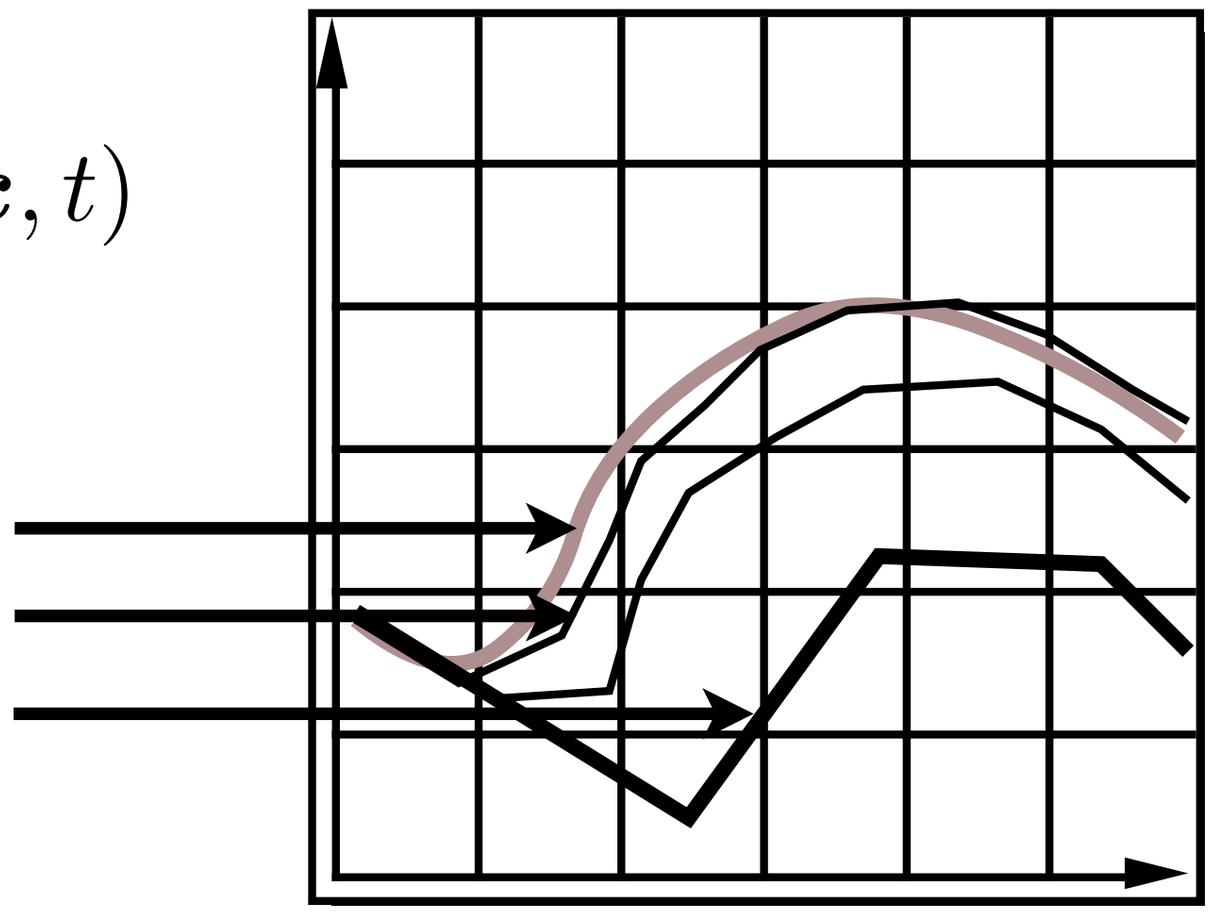
Euler's Method - Errors

With numerical integration, errors accumulate

Euler integration is particularly bad

Example:

$$\boldsymbol{x}^{t+\Delta t} = \boldsymbol{x}^{t} + \Delta t\, \boldsymbol{v}(\boldsymbol{x}, t)$$

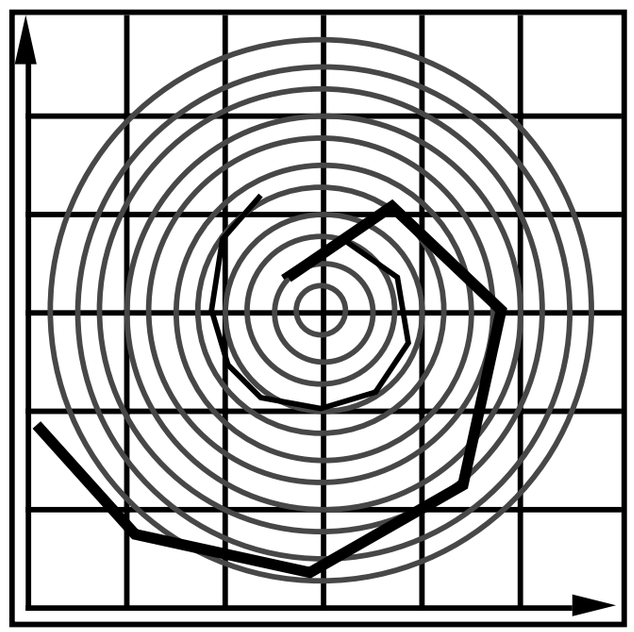


Witkin and Baraff

Instability of the Euler Method

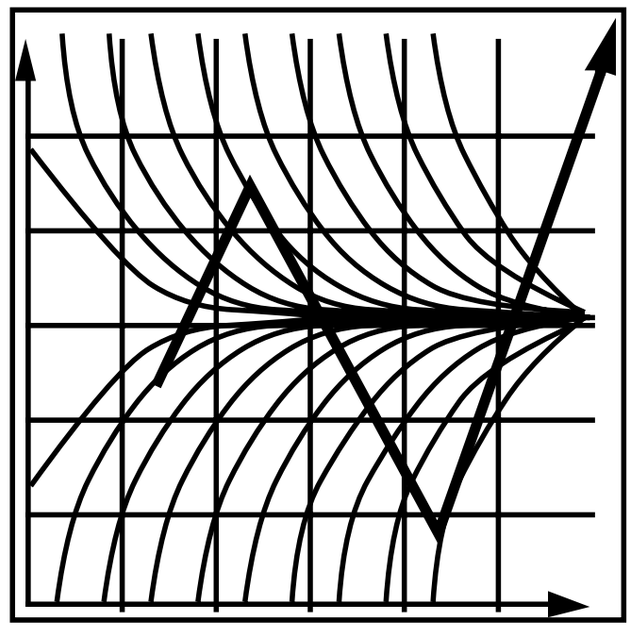
The Euler method (explicit / forward)

$$\boldsymbol{x}^{t+\Delta t} = \boldsymbol{x}^{t} + \Delta t\, \boldsymbol{v}(\boldsymbol{x}, t)$$



Two key problems:

- Inaccuracies increase as time step Δt increases
- Instability is a common, serious problem that can cause simulation to diverge



Witkin and Baraff

Errors and Instability

Solving by numerical integration with finite differences leads to two problems:

Errors

- Errors at each time step accumulate.
  Accuracy decreases as simulation proceeds
- Accuracy may not be critical in graphics applications

Instability

- Errors can compound, causing the simulation to **diverge** even when the underlying system does not
- Lack of stability is a fundamental problem in simulation, and cannot be ignored

Instability: A Disastrous Example

[PLAYERUNKNOWN'S BATTLEGROUNDS, https://www.youtube.com/watch?v=Bz8n6GBAsys]

Combating Instability

Some Methods to Combat Instability

Midpoint method / Modified Euler

- Average velocities at start and endpoint

Adaptive step size

- Compare one step and two half-steps, recursively, until error is acceptable

Implicit methods

- Use the velocity at the next time step (hard)

Position-based / Verlet integration

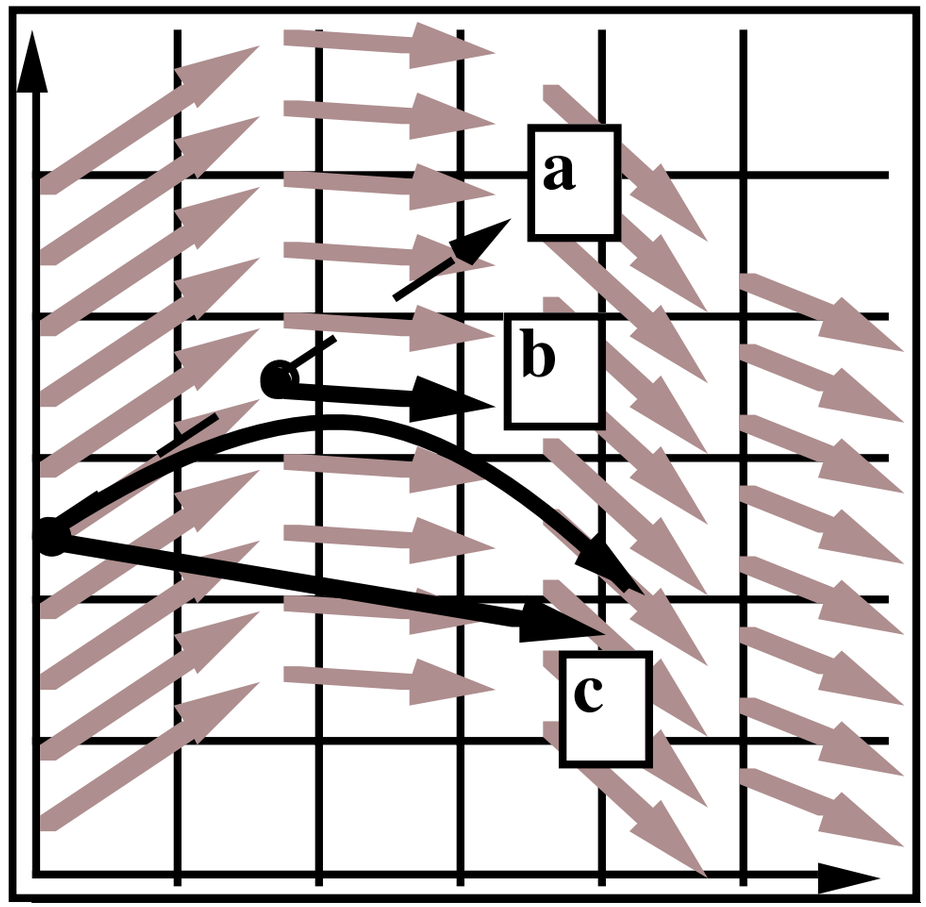
- Constrain positions and velocities of particles after time step

Midpoint Method

Midpoint method

- Compute Euler step (a)
- Compute derivative at midpoint of Euler step (b)
- Update position using midpoint derivative (c)

$$x_{\text{mid}} = x(t) + \Delta t/2 \cdot v(x(t), t)$$
$$x(t + \Delta t) = x(t) + \Delta t \cdot v(x_{\text{mid}}, t)$$



Witkin and Baraff

Modified Euler

Modified Euler

- Average velocity at start and end of step
- Better results

$$\boldsymbol{x}^{t+\Delta t} = \boldsymbol{x}^{t} + \frac{\Delta t}{2}\left(\dot{\boldsymbol{x}}^{t} + \dot{\boldsymbol{x}}^{t+\Delta t}\right)$$

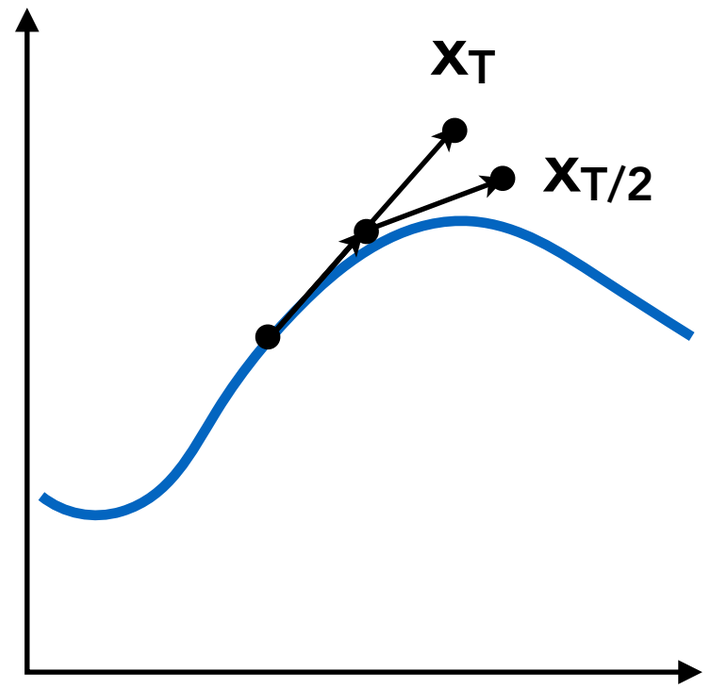
$$\dot{\boldsymbol{x}}^{t+\Delta t} = \dot{\boldsymbol{x}}^{t} + \Delta t\ \ddot{\boldsymbol{x}}^{t}$$

$$\boldsymbol{x}^{t+\Delta t} = \boldsymbol{x}^{t} + \Delta t\ \dot{\boldsymbol{x}}^{t} + \frac{(\Delta t)^2}{2}\ \ddot{\boldsymbol{x}}^{t}$$

Adaptive Step Size

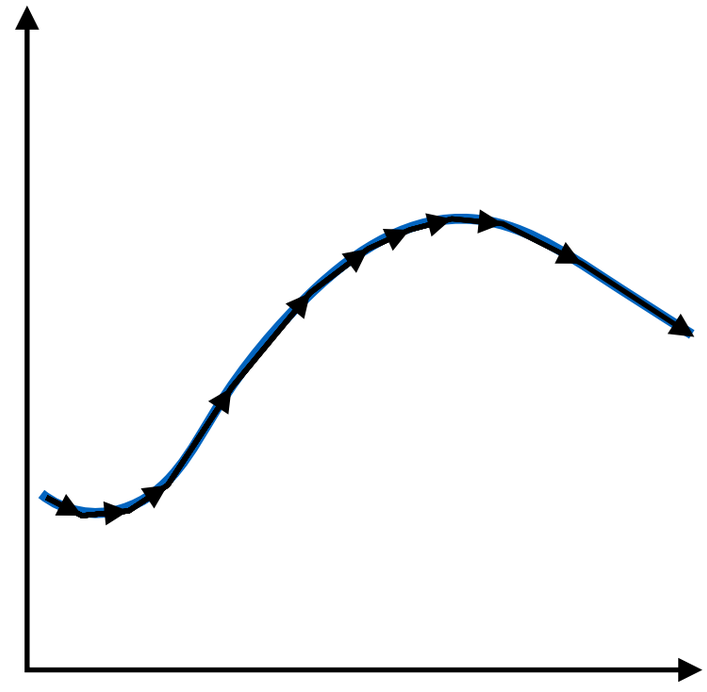
Adaptive step size

- Technique for choosing step size based on error estimate
- Very practical technique
- But may need very small steps!



Repeat until error is below threshold:

- Compute x_T an Euler step, size T
- Compute $x_{T/2}$ two Euler steps, size T/2
- Compute error $\| x_T - x_{T/2} \|$
- If (error > threshold) reduce step size and try again



Slide credit: Funkhouser

Implicit Euler Method

Implicit methods

- Informally called backward methods
- Use derivatives in the future, for the current step

$$\boldsymbol{x}^{t+\Delta t} = \boldsymbol{x}^{t} + \Delta t\, \dot{\boldsymbol{x}}^{t+\Delta t}$$

$$\dot{\boldsymbol{x}}^{t+\Delta t} = \dot{\boldsymbol{x}}^{t} + \Delta t\, \ddot{\boldsymbol{x}}^{t+\Delta t}$$

Implicit Euler Method

Implicit methods

- Informally called backward methods
- Use derivatives in the future, for the current step

$$\boldsymbol{x}^{t+\Delta t} = \boldsymbol{x}^{t} + \Delta t\, \dot{\boldsymbol{x}}^{t+\Delta t}$$

$$\dot{\boldsymbol{x}}^{t+\Delta t} = \dot{\boldsymbol{x}}^{t} + \Delta t\, \ddot{\boldsymbol{x}}^{t+\Delta t}$$

- Solve nonlinear problem for $\boldsymbol{x}^{t+\Delta t}$ and $\dot{\boldsymbol{x}}^{t+\Delta t}$
- Use root-finding algorithm, e.g. Newton's method
- Offers much better stability

Implicit Euler Method

How to determine / quantize “stability”?

- We use the local truncation error (every step) / total accumulated error (overall)
- Absolute values do not matter, but the orders w.r.t. step
- Implicit Euler has order 1, which means that
  - Local truncation error: $O(h^2)$ and
  - Global truncation error: $O(h)$ (h is the step, i.e. Δt)
- Understanding of $O(h)$
  - If we halve h, we can expect the error to halve as well

Runge-Kutta Families

A family of advanced methods for solving ODEs

- Especially good at dealing with non-linearity
- It's order-four version is the most widely used, a.k.a. RK4

Initial condition:

$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0.$$

RK4 solution:

$$y_{n+1} = y_n + \frac{1}{6}h\left(k_1 + 2k_2 + 2k_3 + k_4\right),$$
$$t_{n+1} = t_n + h$$

where

$$k_1 = f(t_n, y_n),$$
$$k_2 = f\left(t_n + \frac{h}{2}, y_n + h\frac{k_1}{2}\right),$$
$$k_3 = f\left(t_n + \frac{h}{2}, y_n + h\frac{k_2}{2}\right),$$
$$k_4 = f\left(t_n + h, y_n + hk_3\right).$$

Position-Based / Verlet Integration

Idea:

- After modified Euler forward-step, constrain positions of particles to prevent divergent, unstable behavior
- Use constrained positions to calculate velocity
- Both of these ideas will dissipate energy, stabilize

Pros / cons

- Fast and simple
- Not physically based, dissipates energy (error)

Details in Assignment 8

Rigid Body Simulation

Simple case

- Similar to simulating a particle
- Just consider a bit more properties

$$\frac{d}{dt}\begin{pmatrix} \mathrm{X} \\ \theta \\ \dot{\mathrm{X}} \\ \omega \end{pmatrix} = \begin{pmatrix} \dot{\mathrm{X}} \\ \omega \\ \mathrm{F}/M \\ \Gamma/I \end{pmatrix}$$

X : positions

θ : rotation angle

ω : angular velocity

F : forces

Γ : torque

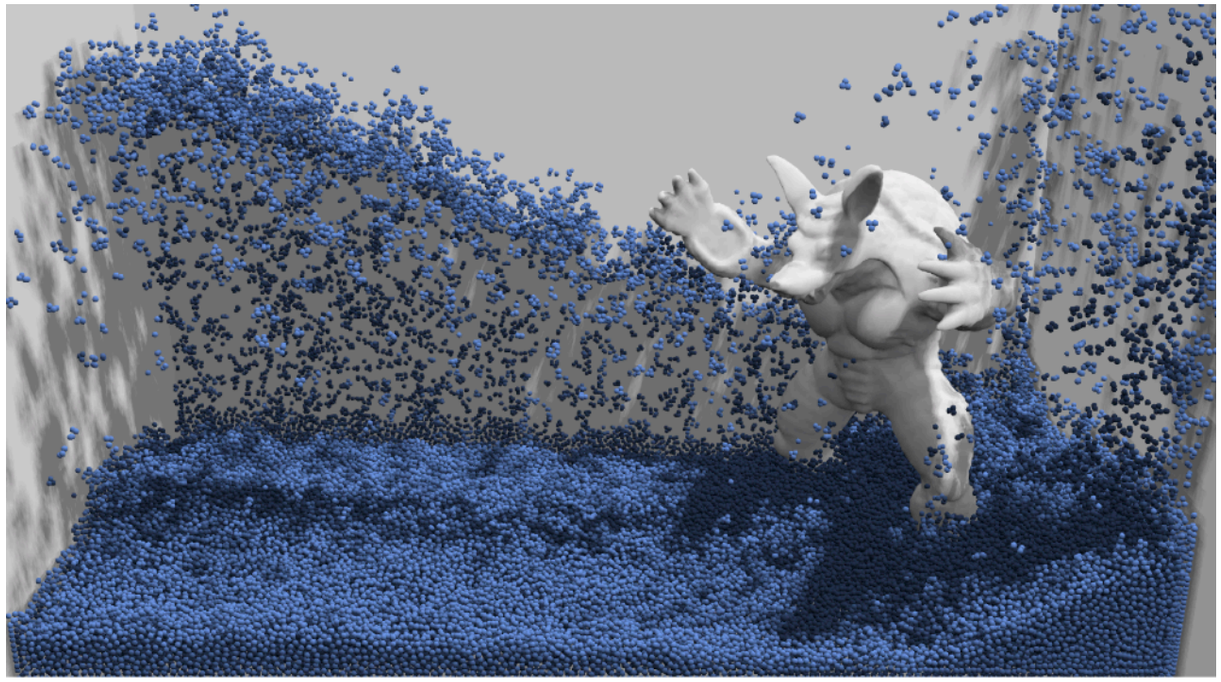
I : momentum of inertia

Fluid Simulation

A Simple Position-Based Method

Key idea

- Assuming water is composed of small rigid-body spheres
- Assuming the water cannot be compressed (i.e. const. density)
- So, as long as the density changes somewhere,
  it should be “corrected” via changing the positions of particles
- You need to know the gradient of the density anywhere w.r.t. each particle’s position
- Update? Just gradient descent!

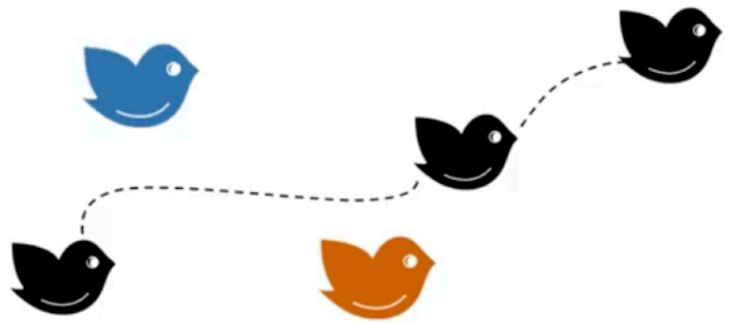


[Macklin et al.]

Eulerian vs. Lagrangian

Two different views to simulating large collections of matters

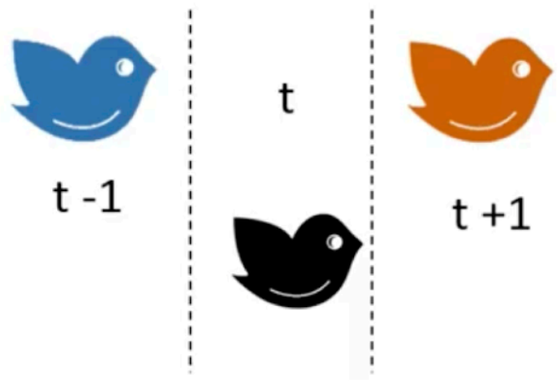
("质点法") LAGRANGIAN APPROACH



Photographer follows the same bird through out its course.

("网格法") EULERIAN APPROACH

The photographer is stationary and can take picture of all the birds passing through one frame only (say at time 't').

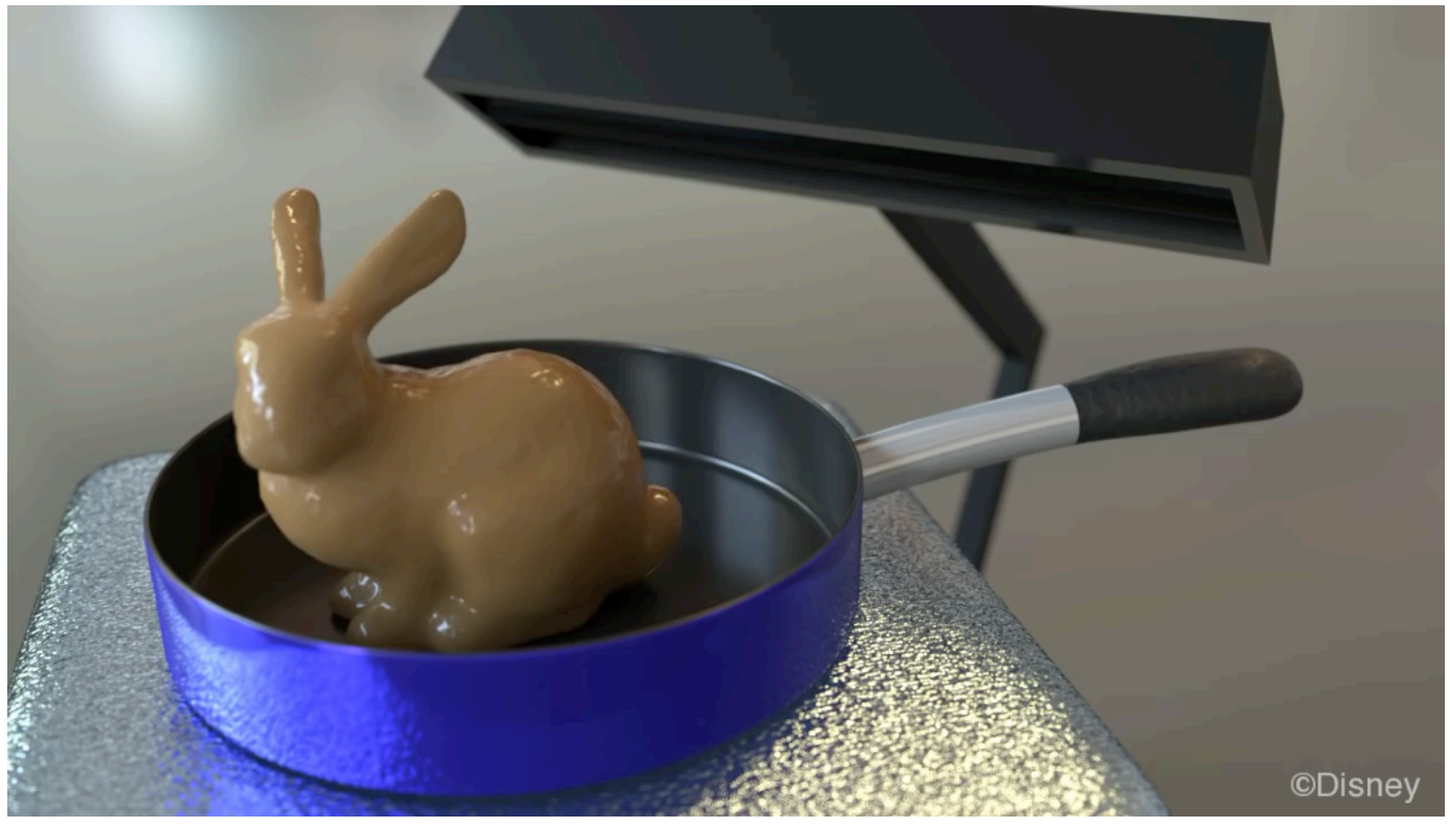


https://www.youtube.com/watch?v=iDIzLkic1pY

Material Point Method (MPM)

Hybrid, combining Eulerian and Lagrangian views

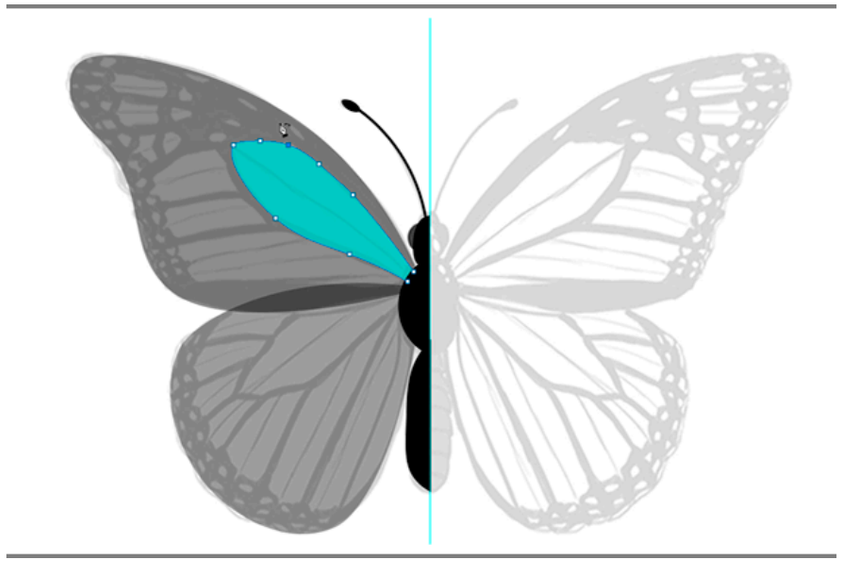
- Lagrangian: consider particles carrying material properties
- Eulerian: use a grid to do numerical integration
- Interaction: particles transfer properties to the grid, grid performs update, then interpo-late back to particles



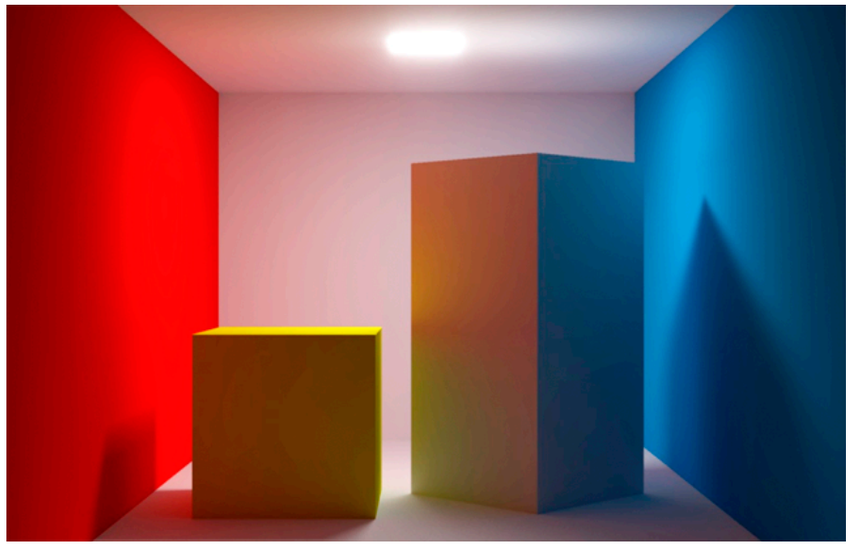
[Stomakhin et al. 2014]

Congratulations!

Rasterization



Geometry



Light Transport

Animation / simulation

Advertisements

GAMES：图形学在线交流平台

GAMES: Graphics And Mixed Environment Symposium

- 2017年6月创建，主页：**http://games-cn.org**
- 宗旨：**学术**界与**产业**界交流与合作的平台
- 线上活动：
  - 每**周四**晚8:00-9:30《在线报告》
  - 2020年2月开始推出《在线课程》
  - 所有活动的视频及PPT都可在线学习和下载

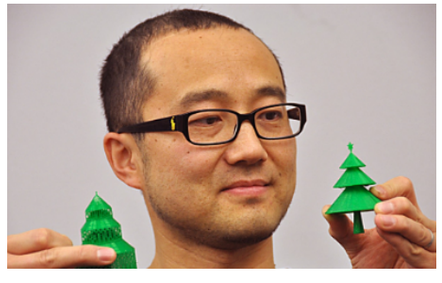
- **【加入GAMES微信群方法】**
  - 步骤1：在微信中加 “**gamesrobot3**” 为好友
  - 步骤2：回复 “**GAMES**”即可获取入群邀请
  - 【前7群基本已满（每群500人），现在加第8群】

学术会议：Chinagraph 2020
（第十三届中国计算机图形学大会）

- 主页：https://chinagraph2020.xmu.edu.cn/
- 重要日期
  - 投稿日期：2020年6月21日 （录用论文将推荐到期刊发表）
  - 会议日期：2020年10月23-25日，厦门，厦门大学

特邀报告

Henry Fuchs
UNC at Chapel Hill



Richard Hao Zhang
Simon Fraser University

会前课程

闫令琪	胡渊鸣	姜仲石	Teseo Schneider
加州大学圣芭芭拉分校	麻省理工学院	纽约大学科朗数学研究所	纽约大学科朗数学研究所

学术会议：ChinaVR 2020
（第二十届中国虚拟现实大会）

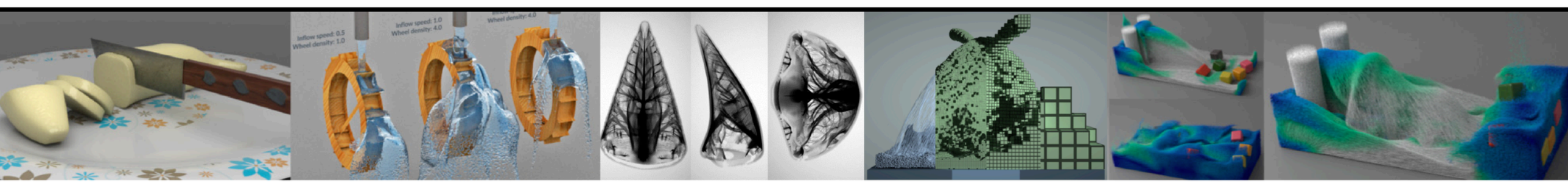
- 主页：https://www.chinavr.info
- 重要日期
  - 投稿日期：2020年5月30日 （录用论文将推荐到期刊发表）
  - 会议日期：2020年9月18-20日，长春，吉林动画学院
- 会前课程
  - 课程1：轻量级移动WebVR示范应用开发
    - 组织者：贾金原（同济大学、吉林动画学院）
  - 课程2：SLAM for AR
    - 组织者：章国锋（浙江大学）
  - 课程3：虚拟现实中的真实行走漫游
    - 组织者：傅孝明（中国科学技术大学）
  - 课程4：高品质移动游戏开发实战
    - 组织者：张鑫（侑虎科技）
- 其他：主题论坛、创新与挑战赛、企业展览

GAMES 201
Advanced Physics Engines 2020: A Hands-on Tutorial

高级物理引擎实战2020 (基于太极编程语言)

Yuanming Hu 胡渊鸣 MIT CSAIL
麻省理工学院 计算机科学与人工智能实验室

Taichi Programming Language



高级物理引擎实战2020

- 课程目标：自己动手打造影视级物理引擎
- 适合人群：0-99岁的计算机图形学爱好者
- 预备知识：高等数学、Python或任何一门程序设计语言
- 课程安排：每周一北京时间晚上20:30-21:30　共10节课
- 课程内容：Taichi语言基础 刚体 液体 烟雾 弹塑性体 PIC/FLIP法 Krylov-子空间求解器 预条件 无矩阵法 多重网格 弱形式与有限元 隐式积分器 辛积分器 拓扑优化 带符号距离场 自由表面追踪 物质点法 大规模物理效果渲染 现代处理器微架构 内存层级 并行编程 GPU编程 稀疏数据结构 可微编程...

2020年六一儿童节开课 不见不散!

Any Other Rendering Courses?

Real-Time High Quality Rendering (in seminar style)

- Soft Shadows and Environment Lighting
- Precomputed Radiance Transfer
- Image-Based Rendering
- Non-Photorealistic Rendering
- Interactive Global Illumination
- Real-time Ray Tracing & DLSS, etc.

[PRT by Wang et al.]

[DLSS 2.0 by NVIDIA]

[Mulan by Disney]

[VXGI by NVIDIA]

[Siren by Unreal Engine]

[RTRT by NVIDIA]

Any Other Rendering Courses?

Advanced Image Synthesis

- Part 1: Advanced Light Transport
- Part 2: Advanced Appearance Modeling
- Part 3: Emerging Technology for Rendering
- Foundations for rendering research!

My Rendering Equation

Interested in Rendering Research?

- It is not because of teaching that I was recognized by the community…, but for research!
- In **rendering** research, I hold the human record of 7 **first-authored** SIGGRAPH[1] papers during Ph.D.

  "Lingqi Yan, first of his name, the unrejected, author of seven papers, breaker of the record, and the chicken eater." -- *Born to be Legendary*, by Lifan Wu, UCSD

- Join my group and become legendary!

Credit: Naruto

Credit: Hearthstone

[1] ACM Transaction on Graphics (ToG)

Special Thanks to All of You!

(And thank Prof. Ren Ng for many of the slides!)