CS148 Summer 2025 - Lecture 13

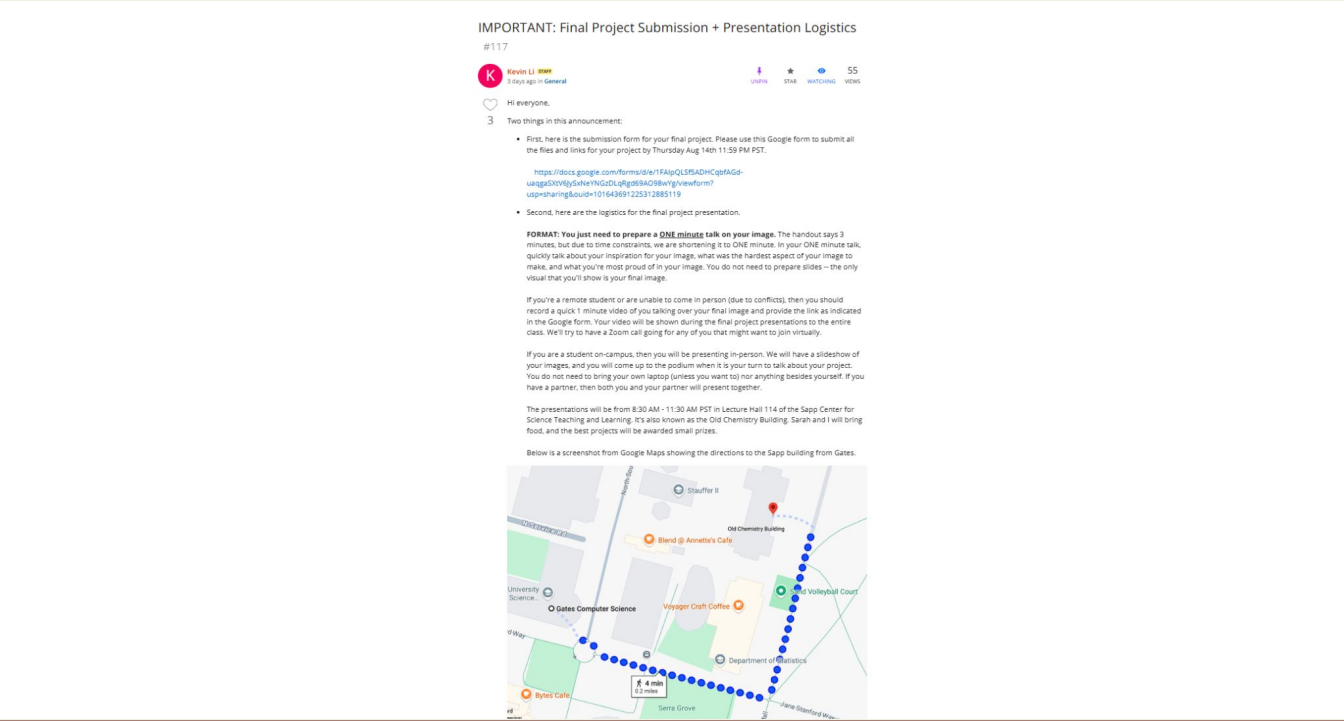
08/05/2025

Advanced Topics

cs148.stanford.edu

Final Project Submission + Presentation

- **https://edstem.org/us/courses/78361/discussion/6847221**



IMPORTANT: Final Project Submission + Presentation Logistics

#117

Kevin Li STAFF
3 days ago in General

UNPIN STAR WATCHING 55 VIEWS

3

Hi everyone,

Two things in this announcement:

- First, here is the submission form for your final project. Please use this Google form to submit all the files and links for your project by Thursday Aug 14th 11:59 PM PST.

  https://docs.google.com/forms/d/e/1FAIpQLSfSADHCqbfAGd-uaqgaSXtV6ly5xNeYNGzDLqRgE69AO9Bw/viewform?usp=sharing&ouid=101643691225312885119

- Second, here are the logistics for the final project presentation.

  FORMAT: You just need to prepare a ONE minute talk on your image. The handout says 3 minutes, but due to time constraints, we are shortening it to ONE minute. In your ONE minute talk, quickly talk about your inspiration for your image, what was the hardest aspect of your image to make, and what you're most proud of in your image. You do not need to prepare slides -- the only visual that you'll show is your final image.

  If you're a remote student or are unable to come in person (due to conflicts), then you should record a quick 1 minute video of you talking over your final image and provide the link as indicated in the Google form. Your video will be shown during the final project presentations to the entire class. We'll try to have a Zoom call going for any of you that might want to join virtually.

  If you are a student on-campus, then you will be presenting in-person. We will have a slideshow of your images, and you will come up to the podium when it is your turn to talk about your project. You do not need to bring your own laptop (unless you want to) nor anything besides yourself. If you have a partner, then both you and your partner will present together.

  The presentations will be from 8:30 AM - 11:30 AM PST in Lecture Hall 114 of the Sapp Center for Science Teaching and Learning. It's also known as the Old Chemistry Building. Sarah and I will bring food, and the best projects will be awarded small prizes.

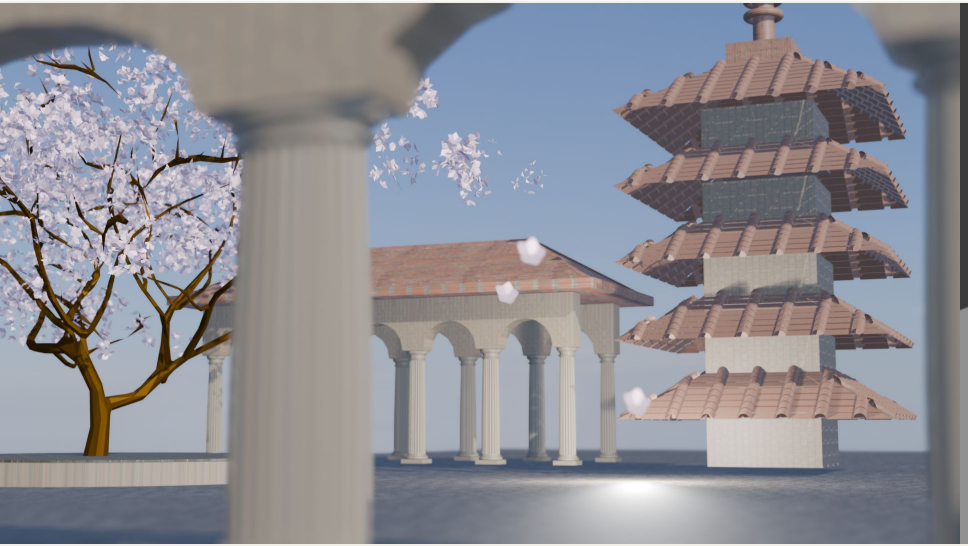
  Below is a screenshot from Google Maps showing the directions to the Sapp building from Gates.

Homework 5 Student Prototypes

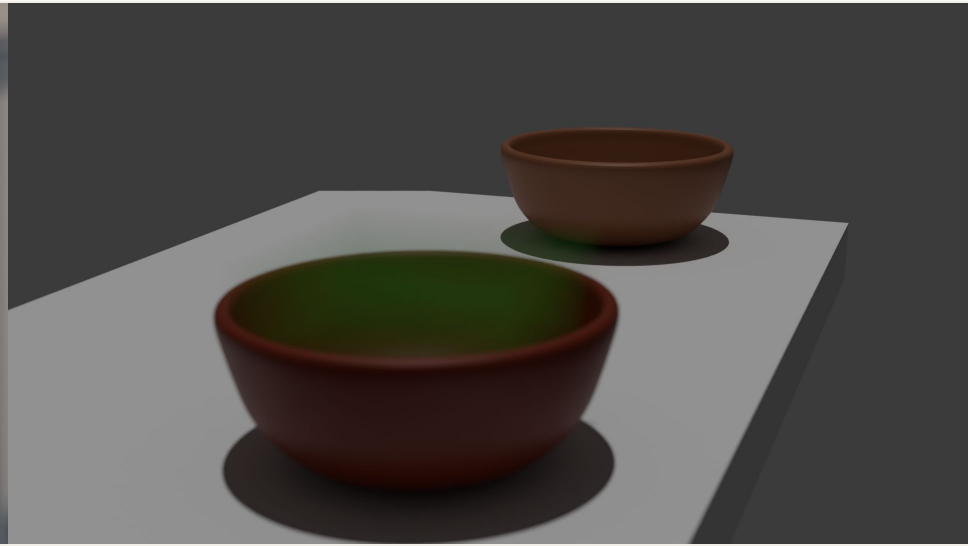
Solveig Jonsdottir

Yimeng Lin

Homework 5 Student Prototypes

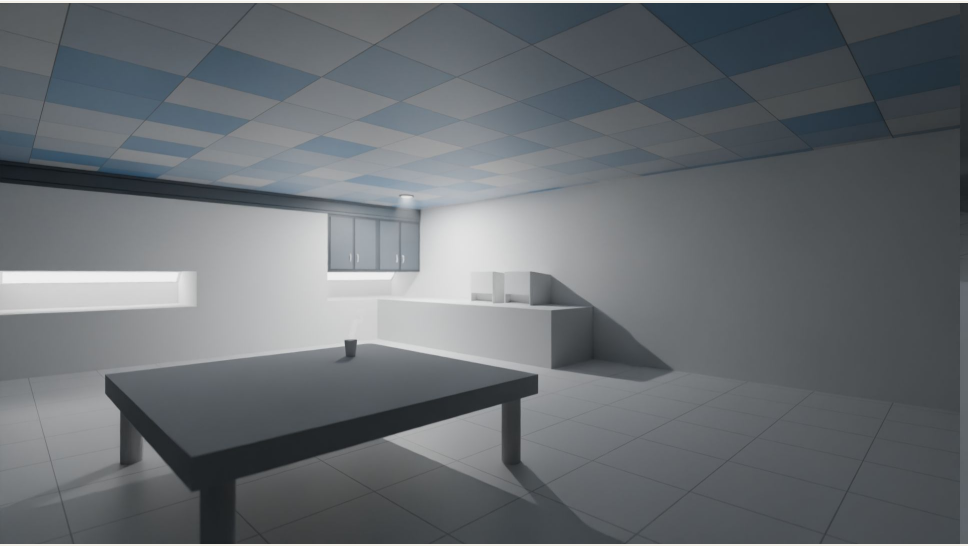


Seigo Hayami

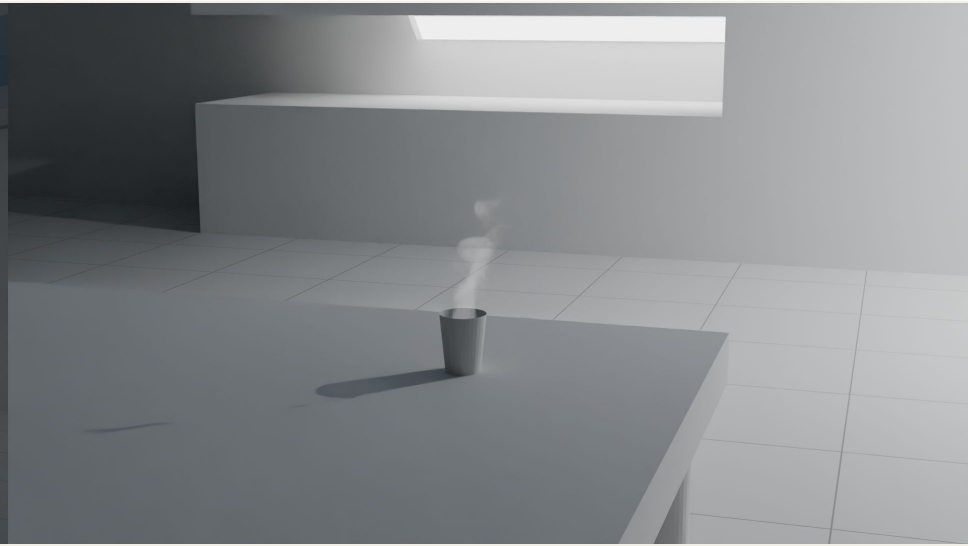


Sureen Heer

Homework 5 Student Prototypes



Sibo Peng



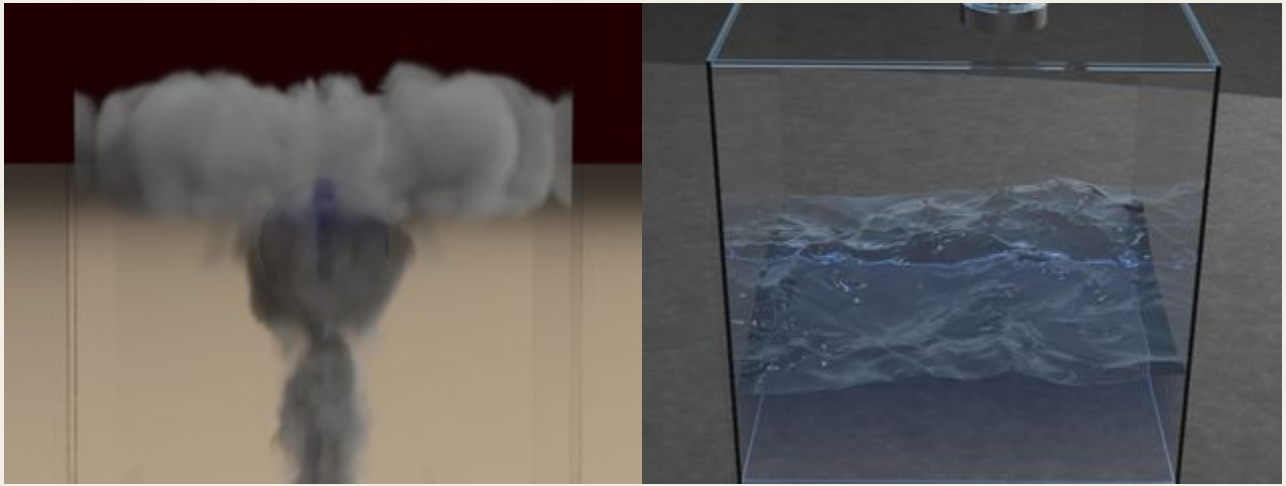
Sibo Peng

Volumetric Rendering

- How to model **participating media** (dust/fog/clouds/smoke,etc)?

Volumetric Rendering

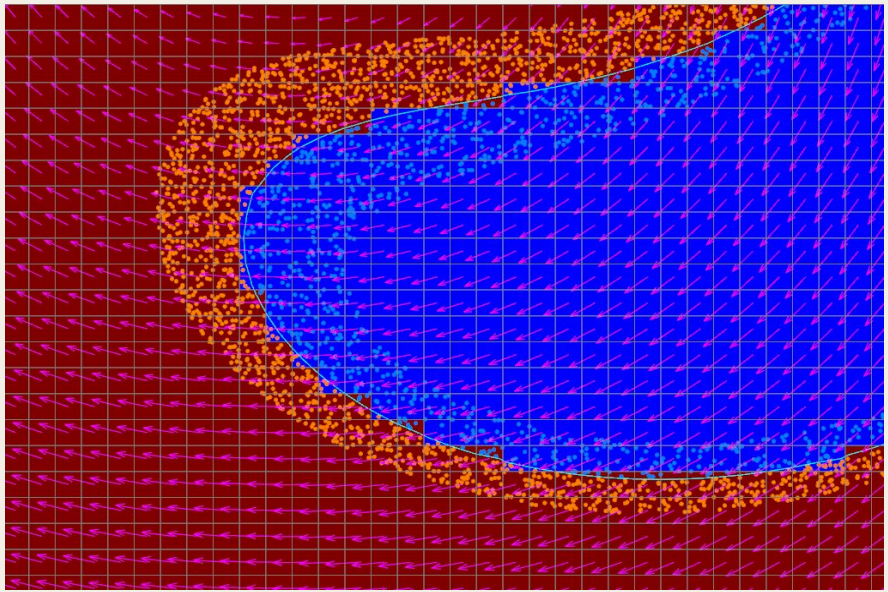
- How to model **participating media** (dust/fog/clouds/smoke,etc)?
- Two part process:
  - **1) Simulate the media** similar to how we similar fluids
  - **2) Ray trace the media** similar to how we handle transmissions



Recall: Fluid Simulation

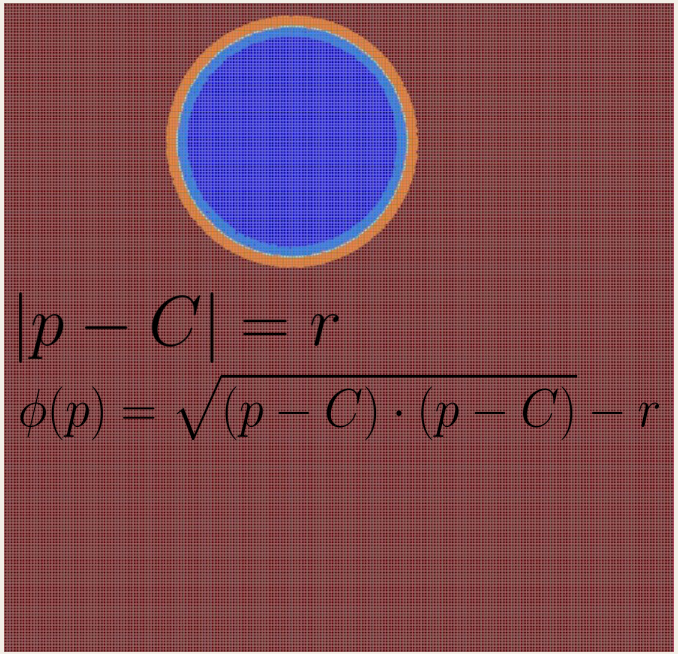
- We track the particles in space using a **fluid domain**
- In 2D we use a rectangular grid, in 3D we use a bounding volume

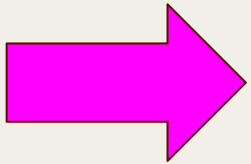
- 2D Example –
- Each **grid cell** stores **velocity vectors** along edges
- Essentially, the domain stores a **velocity field**
- These velocities are used to move the particles in the cell from one state to the next

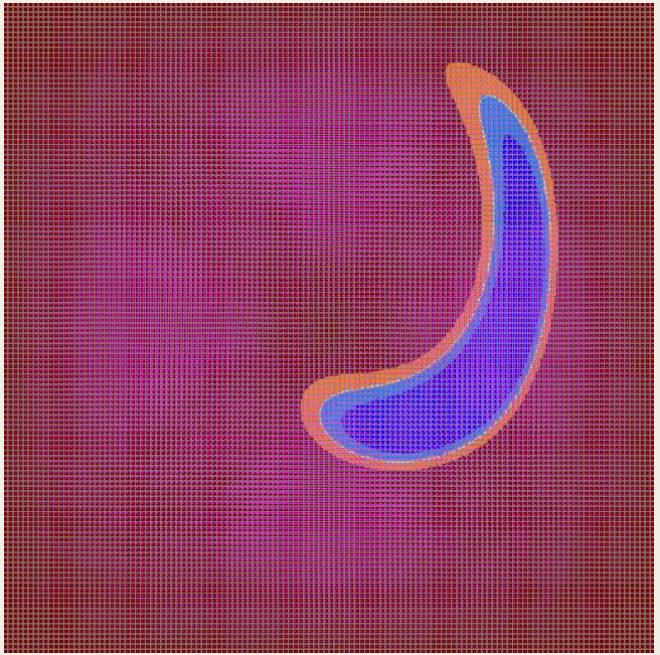


Recall: Fluid Simulation

- Storing the **implicit function** values on each grid cell turns the grid into a **signed distance field** that can move with the **velocity field**



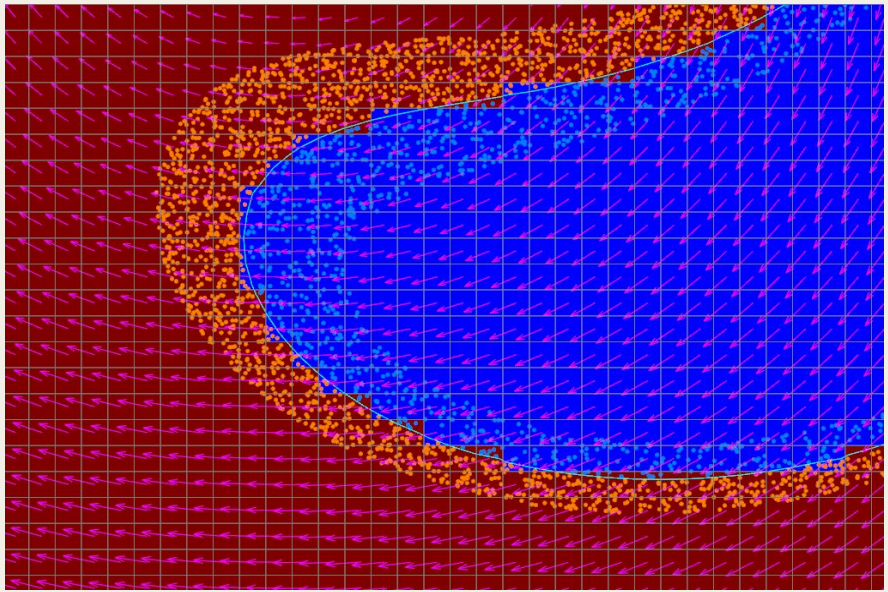




Recall: Fluid Simulation

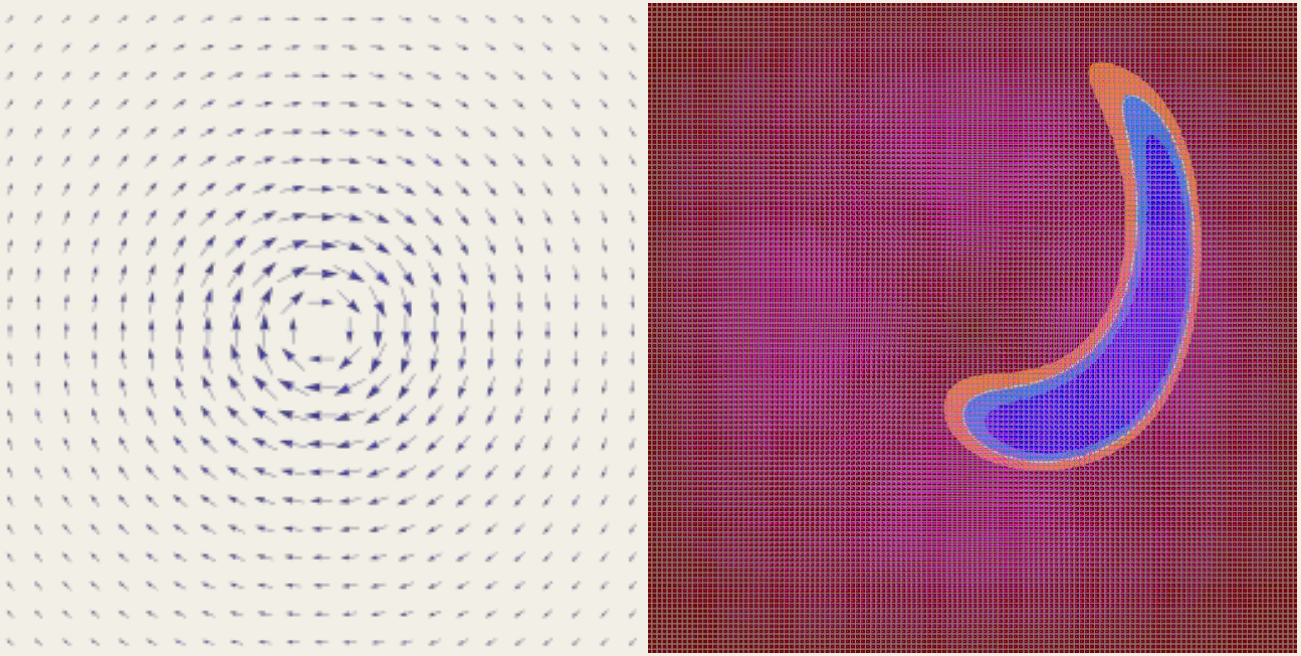
- We track the ~~**particles**~~ **quantities** in space using a fluid domain
- In 2D we use a rectangular grid, in 3D we use a bounding volume

- 2D Example –
- Each grid cell stores velocity vectors along edges
- Essentially, the domain stores a velocity field
- These velocities are used to move the ~~**particles**~~ **quantities** in the cell from state to state



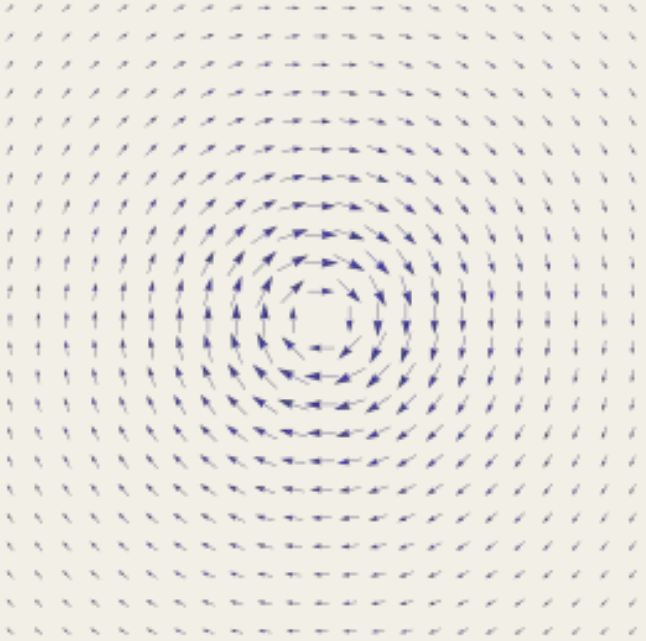
Fields in Calculus and Physics

- Storing quantities on each grid cell gives us a **field**, aka a look-up function that tells us what the quantity is at a point in space



Fields in Calculus and Physics

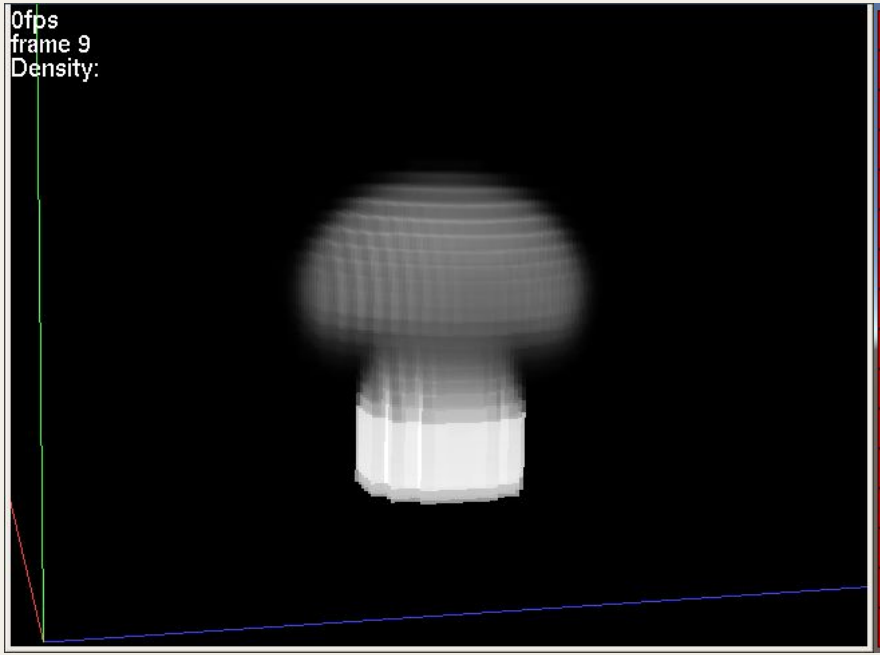
- **Vector fields** store vectors at each grid cell (e.g. velocity fields)
- **Scalar fields** store scalars at each grid cell (e.g. distance fields)



6.6	5.9	5.3	4.7	4.3	3.9	3.6	3.5	3.5	3.6	3.9	4.3	4.7	5.3	5.9	6.6
5.9	5.2	4.5	3.9	3.4	3.0	2.7	2.5	2.5	2.7	3.0	3.4	3.9	4.5	5.2	5.9
5.3	4.5	3.8	3.1	2.5	2.0	1.7	1.5	1.5	1.7	2.0	2.5	3.1	3.8	4.5	5.3
4.7	3.9	3.1	2.4	1.7	1.1	0.7	0.5	0.5	0.7	1.1	1.7	2.4	3.1	3.9	4.7
4.3	3.4	2.5	1.7	0.9	0.3	-0.2	-0.5	-0.5	-0.2	0.3	0.9	1.7	2.5	3.4	4.3
3.9	3.0	2.0	1.1	0.3	-0.5	-1.1	-1.5	-1.5	-1.1	-0.5	0.3	1.1	2.0	3.0	3.9
3.6	2.7	1.7	0.7	-0.2	-1.1	-1.9	-2.4	-2.4	-1.9	-1.1	-0.2	0.7	1.7	2.7	3.6
3.5	2.5	1.5	0.5	-0.5	-1.5	-2.4	-3.3	-3.3	-2.4	-1.5	-0.5	0.5	1.5	2.5	3.5
3.5	2.5	1.5	0.5	-0.5	-1.5	-2.4	-3.3	-3.3	-2.4	-1.5	-0.5	0.5	1.5	2.5	3.5
3.6	2.7	1.7	0.7	-0.2	-1.1	-1.9	-2.4	-2.4	-1.9	-1.1	-0.2	0.7	1.7	2.7	3.6
3.9	3.0	2.0	1.1	0.3	-0.5	-1.1	-1.5	-1.5	-1.1	-0.5	0.3	1.1	2.0	3.0	3.9
4.3	3.4	2.5	1.7	0.9	0.3	-0.2	-0.5	-0.5	-0.2	0.3	0.9	1.7	2.5	3.4	4.3
4.7	3.9	3.1	2.4	1.7	1.1	0.7	0.5	0.5	0.7	1.1	1.7	2.4	3.1	3.9	4.7
5.3	4.5	3.8	3.1	2.5	2.0	1.7	1.5	1.5	1.7	2.0	2.5	3.1	3.8	4.5	5.3
5.9	5.2	4.5	3.9	3.4	3.0	2.7	2.5	2.5	2.7	3.0	3.4	3.9	4.5	5.2	5.9
6.6	5.9	5.3	4.7	4.3	3.9	3.6	3.5	3.5	3.6	3.9	4.3	4.7	5.3	5.9	6.6

Fields for Participating Media

- Participating media like **smoke** and **fog** have **scalar density fields**
- Smoke also involves tracking a **scalar temperature field**



6.6	5.9	5.3	4.7	4.3	3.9	3.6	3.5	3.5	3.6	3.9	4.3	4.7	5.3	5.9	6.6
5.9	5.2	4.5	3.9	3.4	3.0	2.7	2.5	2.5	2.7	3.0	3.4	3.9	4.5	5.2	5.9
5.3	4.5	3.8	3.1	2.5	2.0	1.7	1.5	1.5	1.7	2.0	2.5	3.1	3.8	4.5	5.3
4.7	3.9	3.1	2.4	1.7	1.1	0.7	0.5	0.5	0.7	1.1	1.7	2.4	3.1	3.9	4.7
4.3	3.4	2.5	1.7	0.9	0.3	-0.2	-0.5	-0.5	-0.2	0.3	0.9	1.7	2.5	3.4	4.3
3.9	3.0	2.0	1.1	0.3	-0.5	-1.1	-1.5	-1.5	-1.1	-0.5	0.3	1.1	2.0	3.0	3.9
3.6	2.7	1.7	0.7	-0.2	-1.1	-1.9	-2.4	-2.4	-1.9	-1.1	-0.2	0.7	1.7	2.7	3.6
3.5	2.5	1.5	0.5	-0.5	-1.5	-2.4	-3.3	-3.3	-2.4	-1.5	-0.5	0.5	1.5	2.5	3.5
3.5	2.5	1.5	0.5	-0.5	-1.5	-2.4	-3.3	-3.3	-2.4	-1.5	-0.5	0.5	1.5	2.5	3.5
3.6	2.7	1.7	0.7	-0.2	-1.1	-1.9	-2.4	-2.4	-1.9	-1.1	-0.2	0.7	1.7	2.7	3.6
3.9	3.0	2.0	1.1	0.3	-0.5	-1.1	-1.5	-1.5	-1.1	-0.5	0.3	1.1	2.0	3.0	3.9
4.3	3.4	2.5	1.7	0.9	0.3	-0.2	-0.5	-0.5	-0.2	0.3	0.9	1.7	2.5	3.4	4.3
4.7	3.9	3.1	2.4	1.7	1.1	0.7	0.5	0.5	0.7	1.1	1.7	2.4	3.1	3.9	4.7
5.3	4.5	3.8	3.1	2.5	2.0	1.7	1.5	1.5	1.7	2.0	2.5	3.1	3.8	4.5	5.3
5.9	5.2	4.5	3.9	3.4	3.0	2.7	2.5	2.5	2.7	3.0	3.4	3.9	4.5	5.2	5.9
6.6	5.9	5.3	4.7	4.3	3.9	3.6	3.5	3.5	3.6	3.9	4.3	4.7	5.3	5.9	6.6

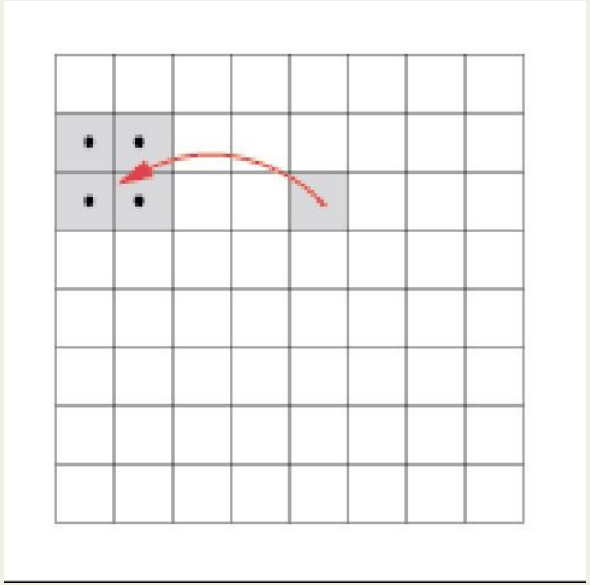
Advecting Fields with the Velocity Field

- The **evolution of a field** over time **due to a velocity field** u can be thought of as moving a particle in a cell due to the velocity

- Recall the equation of motion for a point given its initial position and velocity, and a time step size:

$$q(t + \Delta t) = q(t) + (\Delta t)v(t)$$

$$q_{k+1} = q_k + (\Delta t)v_k$$



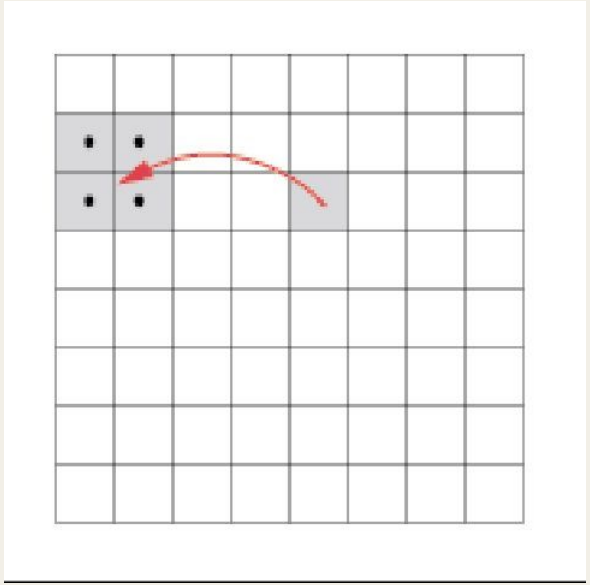
Advecting Fields with the Velocity Field

- The **evolution of a field** over time **due to a velocity field** u can be thought of as moving a particle in a cell due to the velocity

- Naive Approach: **Treat the grid cell as if it were a point** via its center, and **compute where it moves** to based on the velocity vector that was stored at the cell:

$$q_{k+1} = q_k + (\Delta t) v_k$$

- After computing where the grid cell moves to, **store its quantity in the new location**



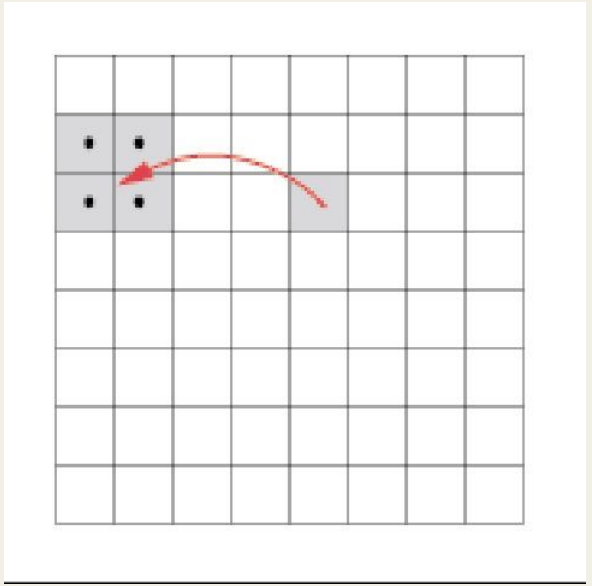
Advecting Fields with the Velocity Field

- The **evolution of a field** over time **due to a velocity field** u can be thought of as moving a particle in a cell due to the velocity

- Naive Approach:

$$q_{k+1} = q_k + (\Delta t) v_k$$

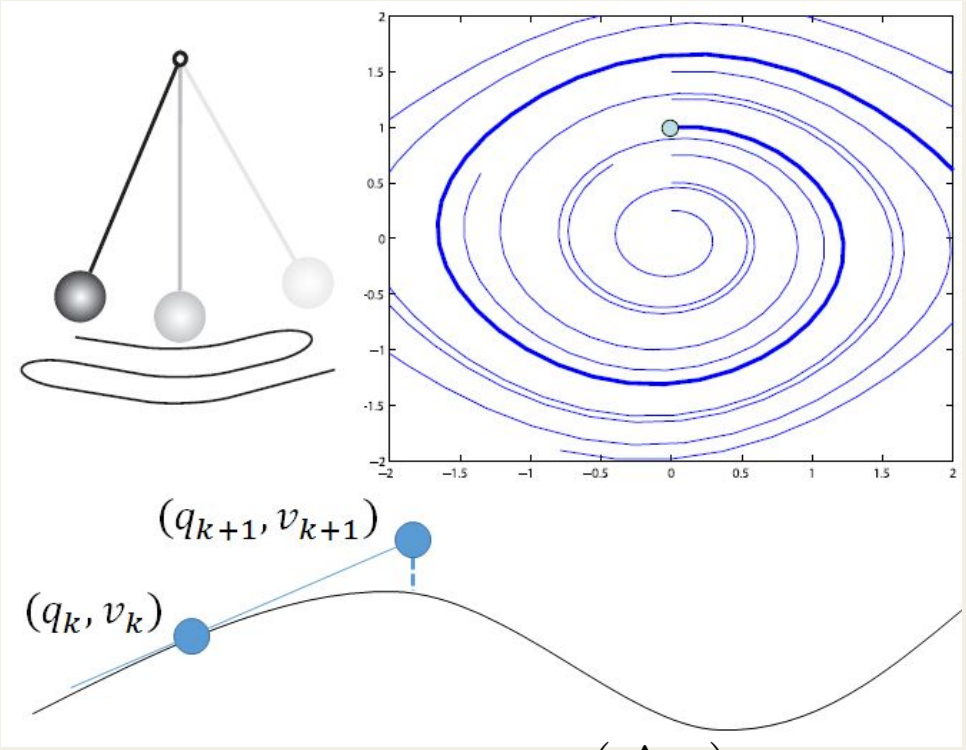
- Problems:
  - Uses **Explicit Euler**, which recall can be unstable (“blow up”) for big time steps
  - What if the new location of the cell isn’t exactly a grid cell (e.g. see diagram)?



Recall: Explicit Euler

- Simple Pendulum: We may increase the amplitude of the pendulum every step!
- Why? Our equations are approximations (1st order Taylor's series)

- Need to take small time steps (and can do analysis to see how small)



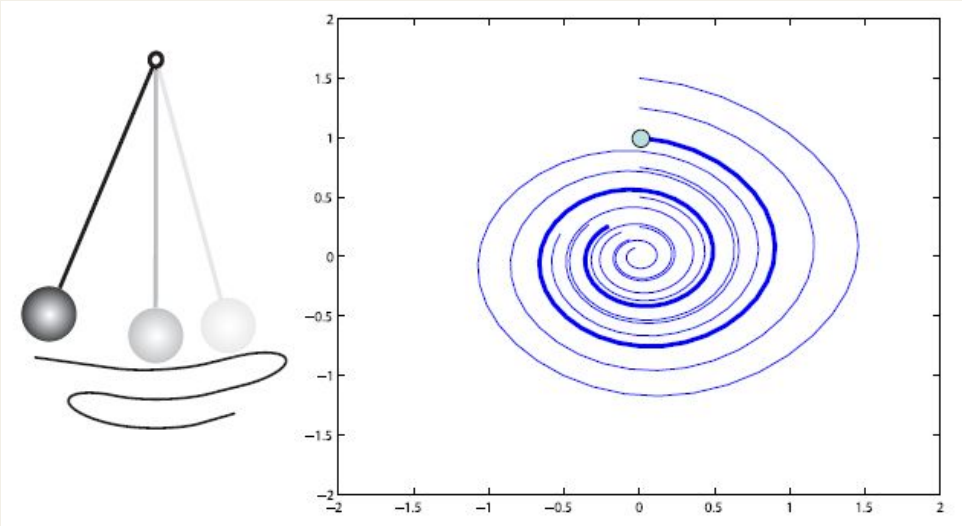
$$q_{k+1} = q_k + (\Delta t) v_k$$

Alternatively: Implicit Euler

- Consider **Implicit Euler** (aka Backward Euler):

$$q_{k+1} = q_k + (\Delta t) v_{k+1}$$

- Can do analysis to see this is stable for ALL time steps!

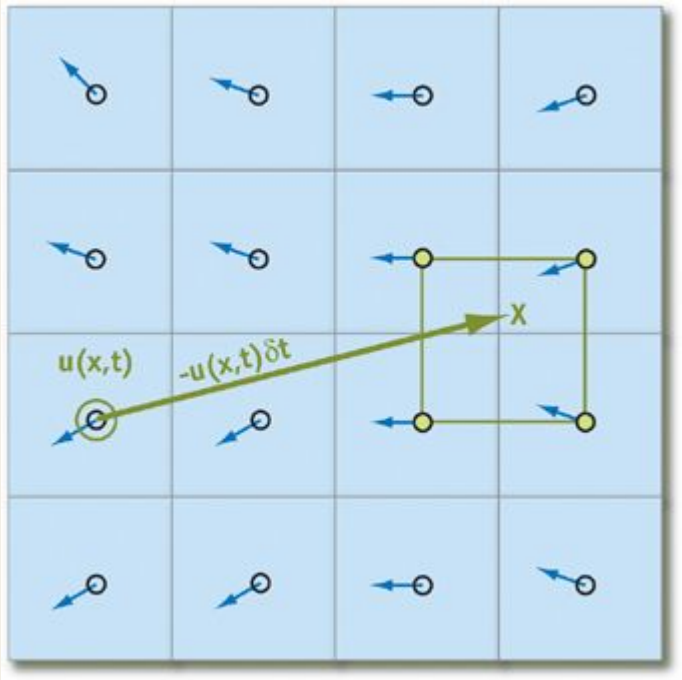


- **What if we use Implicit Euler for our advection process?**

Implicit Euler for Advection

- Use the velocity stored at the grid cell to **trace it "back in time"** to its last position (hence "Backward" Euler)

- Implicit Euler Advection Process:
  - Trace grid cell center back in time to get position X
  - Interpolate surrounding grid cells around X to get quantity at X
  - Store computed quantity at the grid cell for its new value



Advecting Velocity with Velocity

- We can advect any field (distance, density, temperature, etc fields) with the velocity field
- Why not also advect the velocity field with itself?

- Actually done in the **Navier-Stokes equations for incompressible flow** when updating the state of a fluid with velocity field u:

$$\frac{\partial u}{\partial t} = -(u \cdot \nabla)u - \frac{1}{\rho}\nabla p + \nu \nabla^2 u + F$$

$$\nabla \cdot u = 0$$

The "Advection Operator"

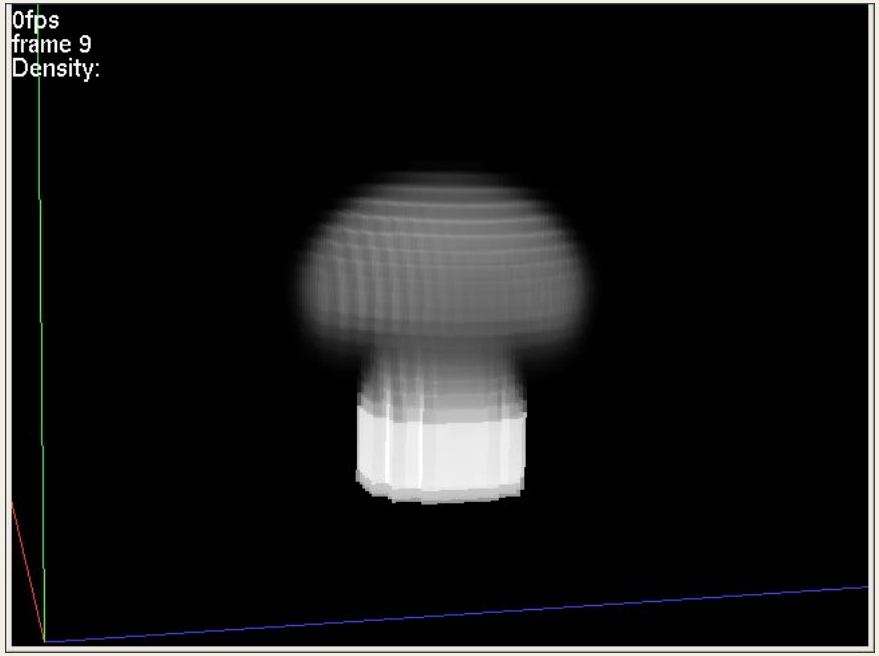
Can read more on Navier-Stokes from this Nvidia paper

Questions?

Ray Tracing Density Fields

- Participating media like **smoke** and **fog** have **scalar density fields**
- Step 1: Simulate the media and update its (density) fields

- Step 2: Take into account the density field as we shoot rays through the domain



Absorption

- Step 2: Take into account the density field as we shoot rays through the domain
- As light travels through participating media, it can be **absorbed** and converted into non-visible energy, e.g. heat

- Computer science parallel: as light moves a distance dx along a ray, a fraction (absorption coefficient σ_a) of its radiance is lost/absorbed:

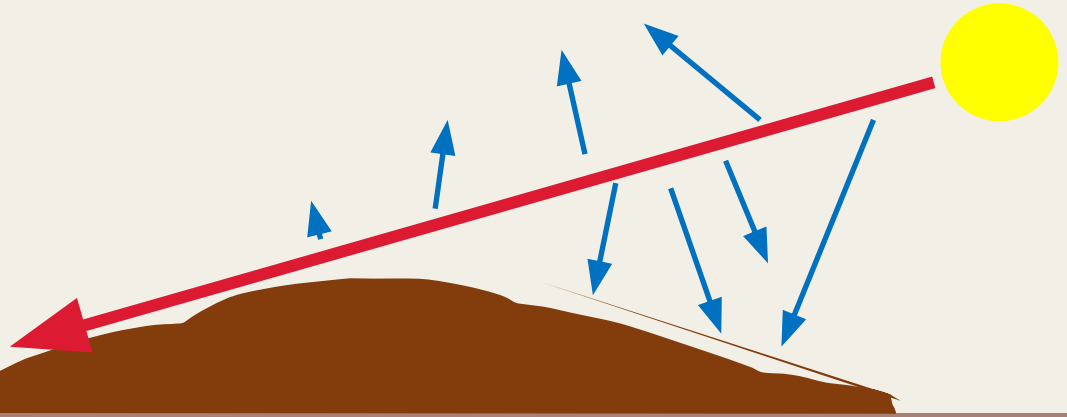
$$dL(x,\omega) = -\sigma_a(x)L(x,\omega)dx$$

Based on Density Field

Out-Scattering

- As light travels through participating media, it can be **out-scattered** to different direction(s)
- Computer science parallel: as light moves a distance dx along a ray, a fraction (scattering coefficient σ_S) of its radiance is lost/scattered

$$dL(x,\omega) = -\sigma_s(x)L(x,\omega)dx$$



Attenuation

- Combine the absorption and scattering coefficients to get the **attenuation coefficient**:

$$c(x) = \sigma_a(x) + \sigma_s(x)$$

$$dL(x, \omega) = -c(x)L(x, \omega)dx$$

Attenuation

- Combine the absorption and scattering coefficients to get the **attenuation coefficient**:

$$c(x) = \sigma_a(x) + \sigma_s(x)$$

$$dL(x, \omega) = -c(x)L(x, \omega)dx$$

- Integrate both sides… we get **Beer's Law**!

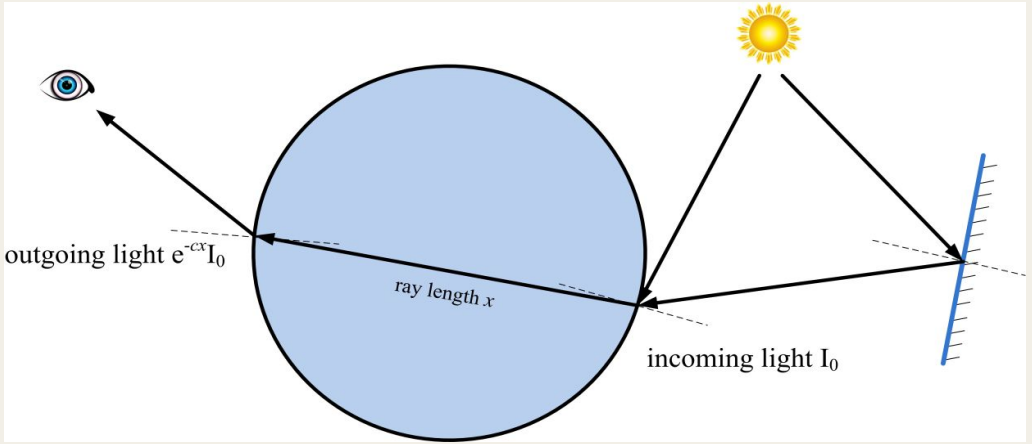
$$L_{atten} = Le^{-cx}$$

Beer's Law

- Recall **Beer's Law** tells us how much light is lost as we go through a medium, e.g. going from air through a transparent material

$$L_{atten} = Le^{-cx}$$

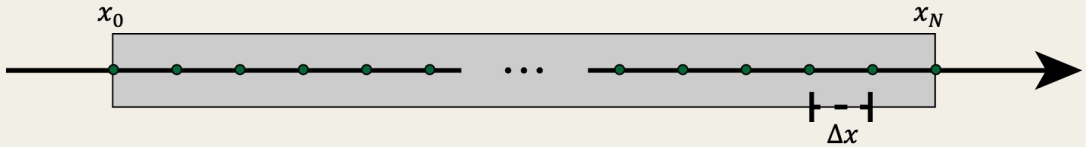
- Light enters a medium along a ray and travels through a distance x
- The **falloff** (fraction of light lost) is e^(-cx)



Heterogeneous Beer's Law

- For homogenous media, the attenuation coefficients are constant throughout (e.g. the glass from HW3)
- For participating media (smoke, fog, etc), our density field has changing quantities throughout our grid domain
- Must use **heterogeneous Beer's Law** to compute falloff for light along the ray at every tiny step as we ray trace:

$$Le^{-c(\frac{x_0+x_1}{2})\Delta x}e^{-c(\frac{x_1+x_2}{2})\Delta x}\ldots e^{-c(\frac{x_{N-1}+x_N}{2})\Delta x}$$



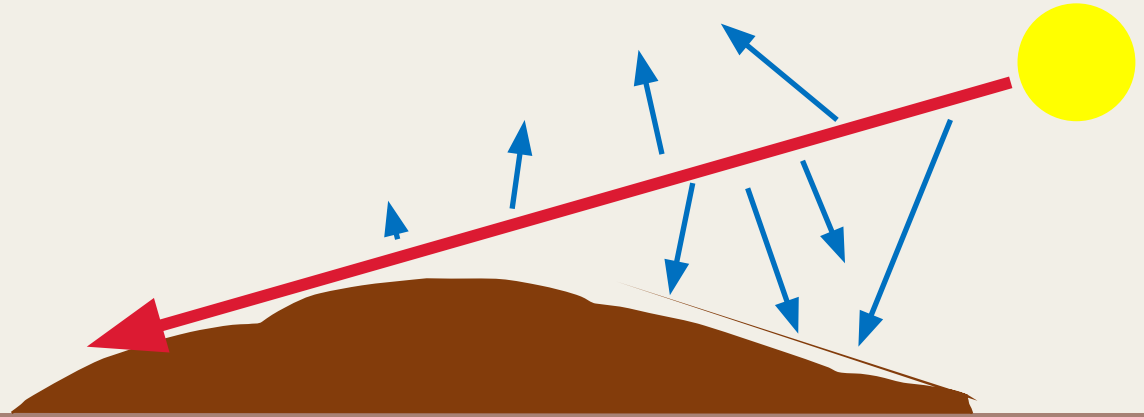
Attenuation of Shadow and Camera Rays

- Attenuation causes participating media like smoke to cast a shadow
- Light gets lost as it goes through the smoke density field, becoming darker

- Attenuation also obscures objects in smoke due to lost light
- But what causes the smoke shape?

In-Scattering

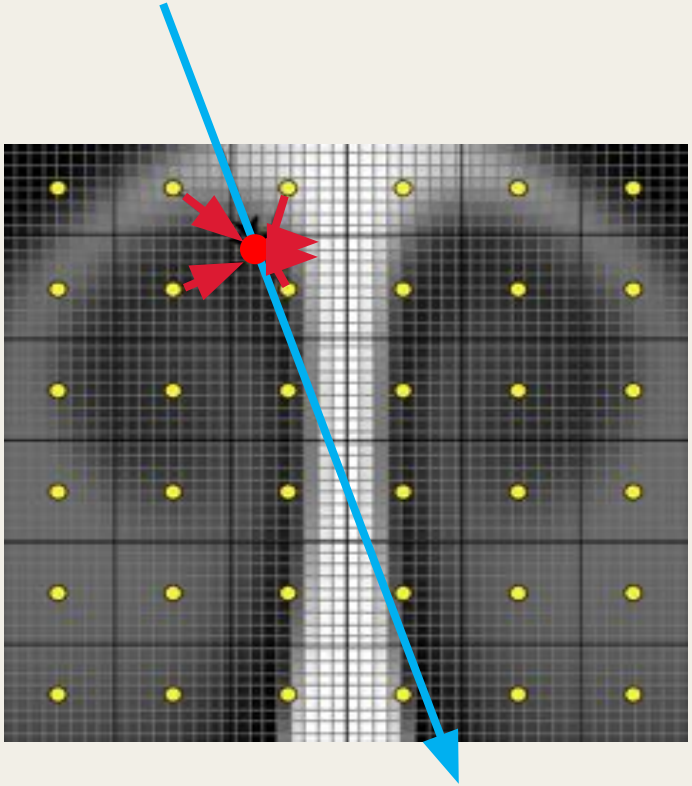
- As light travels through participating media, it can be **out-scattered** to different direction(s)
- The out-scattered light can be **in-scattered** to our eyes/camera
- Example: the blue components of sunlight out-scattered by the atmosphere become in-scattered into our eye to make the sky blue



Volumetric Light Map

- Model in-scattering when ray tracing with a **volumetric light map**
- **Precompute** one **for each light**

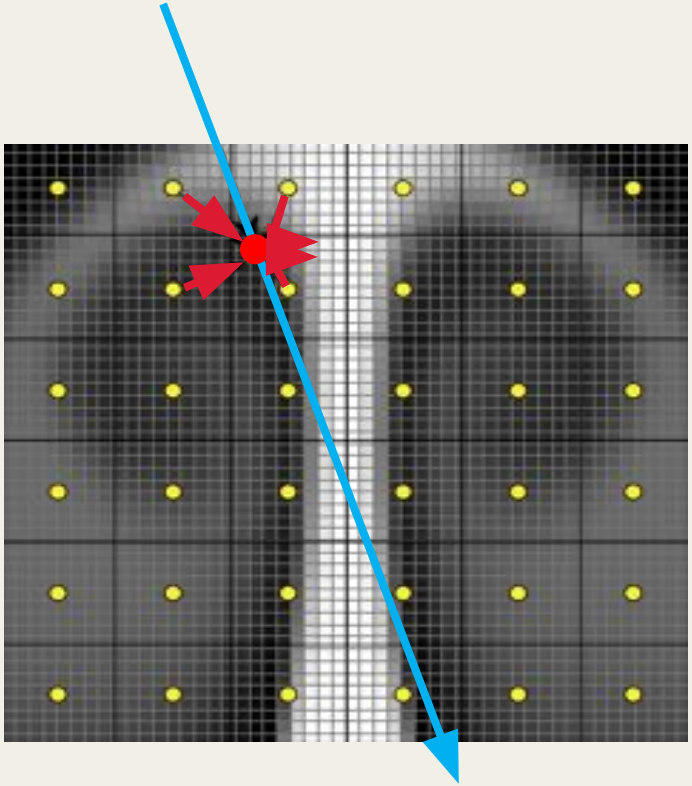
- For each volumetric light map:
  - At each grid cell of our domain, cast a ray to the light
  - Compute how much light gets to that grid cell through the density field
  - Store that radiance at the cell



Volumetric Light Map

- As we ray trace, interpolate radiance values in nearby grid cells to see how much light is available for in-scattering

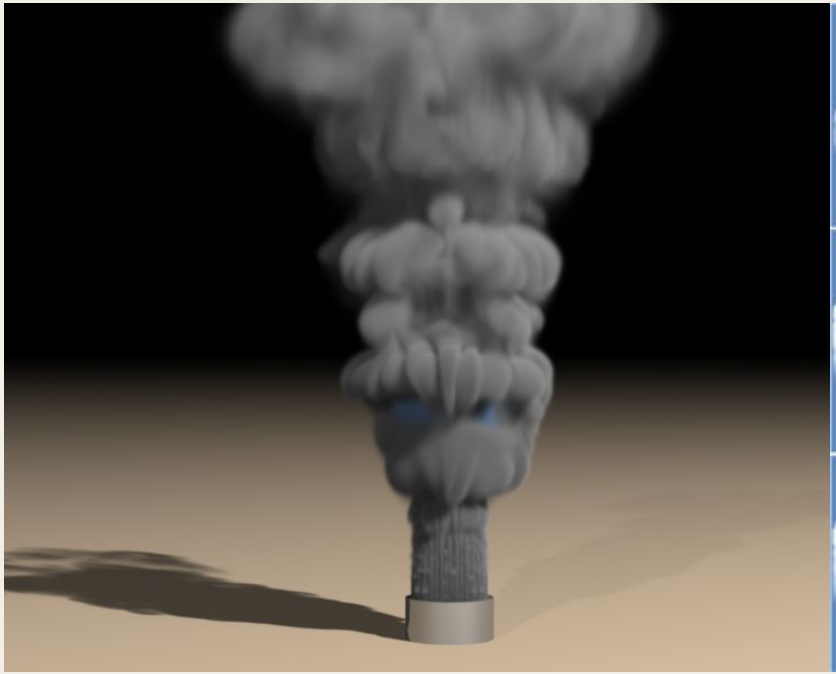
- Use a **phase function** that depends on the direction of our ray and the light to compute the probability of adding the interpolated light as in-scattered light

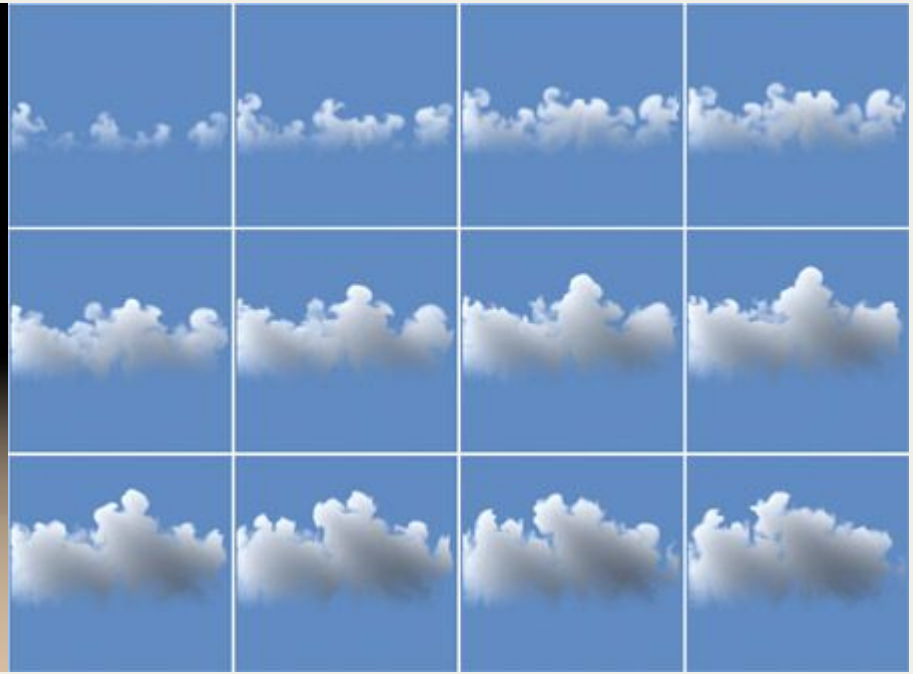


Phase Functions

- As we ray trace, interpolate radiance values in nearby grid cells to see how much light is available for in-scattering
- Use a **phase function** that depends on the direction of our ray and the light to compute the probability of adding the interpolated light as in-scattered light
- Example phase functions of angle between our ray and the light:
  - Rayleigh (for atmosphere): $$p(\theta) = \frac{3}{8}(1 + \cos^2\theta)$$
  - Henyey-Greenstein (more general): $$p(\theta) = \frac{\frac{1}{4\pi}(1 - g^2)}{(1 + g^2 - 2g\cos\theta)^{1.5}}$$

Volumetric Rendering





Questions?