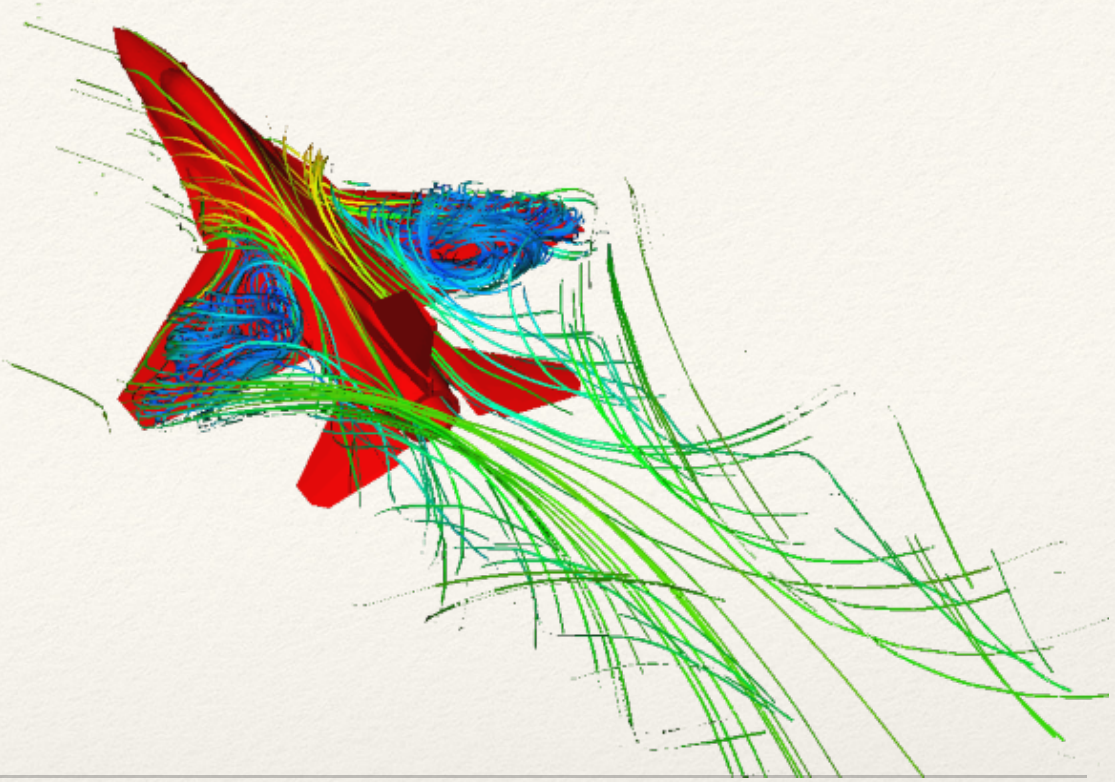


Dr. Sarah Abraham
University of Texas at Austin
Computer Science Department

Physical Simulation

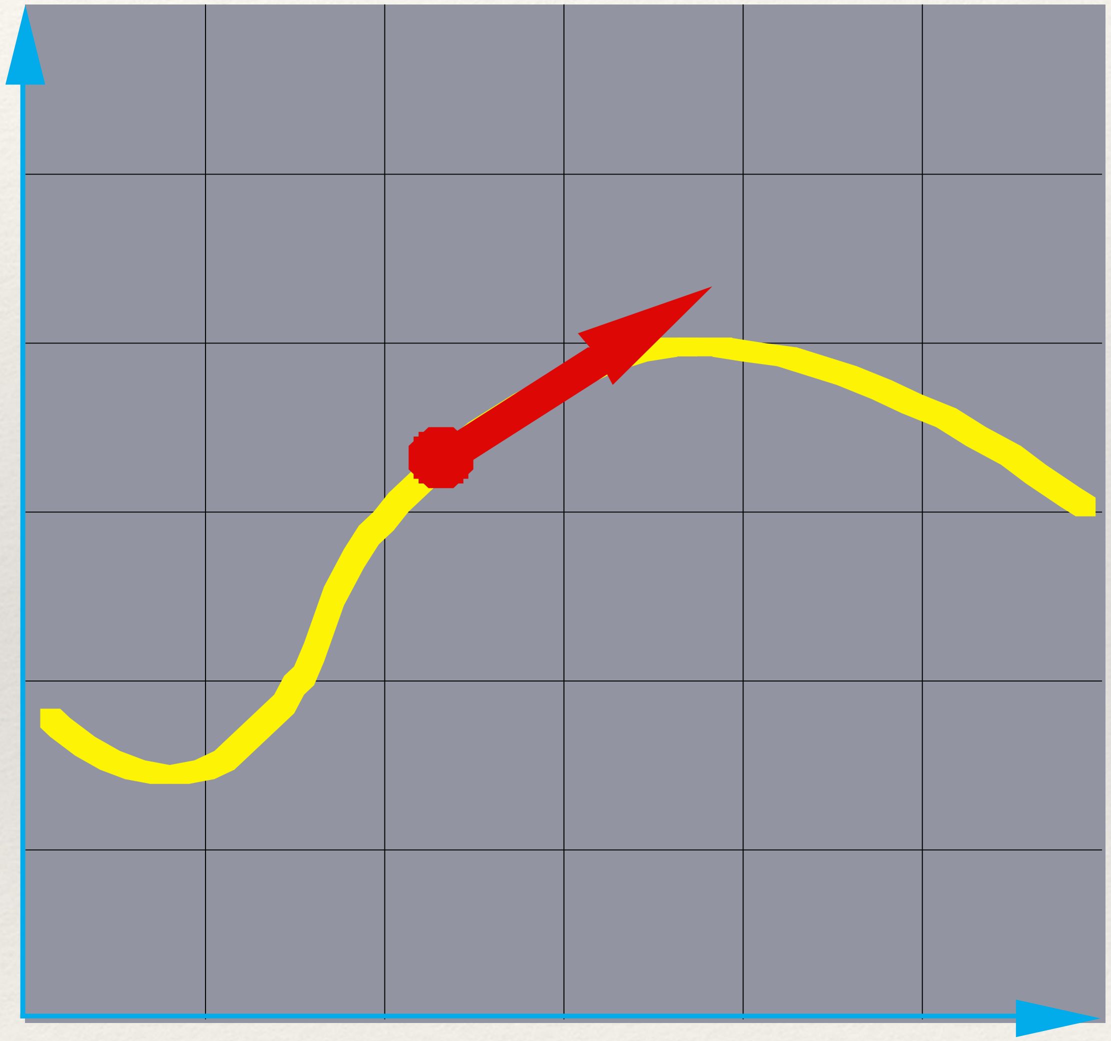
Elements of Graphics
CS324e

Newton's Equations of Motion

- Equations that describe motion over time
- Provide model for relating forces to object trajectory
  - $F = ma$
- Integrating over time captures a system's physical behaviors
- How are we discretizing these equations for computer simulation?

Particles Along a Trajectory

- Particle has a position and a velocity
- Calculate position over time by starting at a point and considering velocity for that given time interval



Euler's Method

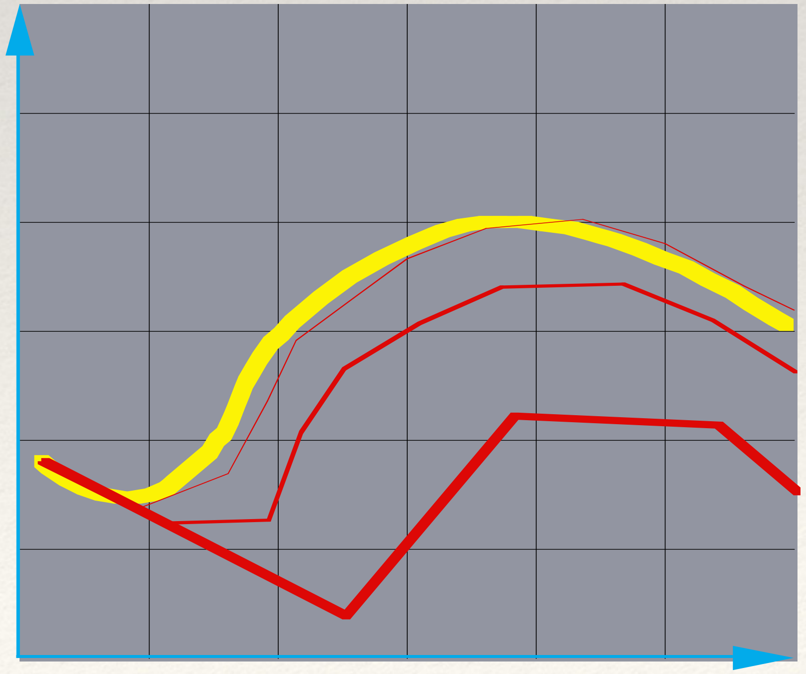
- Take linear time steps (Δt) along the flow:

$$\vec{\mathbf{x}}(t+\Delta t) = \vec{\mathbf{x}}(t) + \Delta t \cdot \dot{\vec{\mathbf{x}}}(t) = \vec{\mathbf{x}}(t) + \Delta t \cdot g(\vec{\mathbf{x}},t)$$

- Write as a time iteration:

$$\vec{\mathbf{x}}^{\,i+1} = \vec{x}^{\,i} + \Delta t \cdot \vec{\mathbf{v}}^{\,i}$$

- Visualized across time steps:



Accounting for Mass

- Particle has mass m
- Particle is in a force field $\mathbf{f}$
- Newton's Second Law:

$$\vec{\mathbf{f}} = m\vec{\mathbf{a}} = m\ddot{\mathbf{x}}$$

Particle With Mass Example

```
//Class fields

float x, y;

float vx, vy;

float ax, ay;

float m;
```

```
//Class method to apply forces

applyForces(float fx, float fy)
{
   ax = fx/m;
   ay = fy/m;
   vx += ax;
   vy += ay;
   x += vx;
   y += vy;
}
```

Problems

- Inaccurate over larger time steps
- Creates numeric instabilities as error accumulates
- Better, more stable methods exist, so explicit Eulerian is rarely used
- But it should be okay for our purposes in this class!

Verlet Integration

- A better solver that doesn't require much additional calculations
- Consider our forward Euler equations:

  $v_{t+1} = v_t + a\Delta t$

  $p_{t+1} = p_t + v_{t+1}\Delta t$

- Verlet looks like this:

  $p_{t+1} = p_t + (p_t - p_{t-1}) + a\Delta t^2$

  $p_{t-1} = p_t$

Spring Forces

- Spring force is based on:
  - Spring stiffness (k)
  - Amount of stretch from resting position (X)
- Hooke's Law: $f = -kX$

Spring Example

```
float y;
float vy;
float m = 1.0;
float ry = 250;
float ks = 0.1;

void setup() {
  size(500, 500);
}
```

```
void draw() {
  background(210);

  float f = -(ks * (y - ry));
  float a = f/m;
  vy = vy + a;
  y += vy;

  rect(200, y, 100, 20);
}
```

Instapoll Question: Springs

- What does **ry** represent in the line of code: float f = -(ks * (y - ry)); ?
  - Spring stiffness
  - Amount of stretch
  - Spring damping
  - Spring resting position

Spring Damping

- If force due to a spring is:

$$F = -k_s X$$

- Spring force with damping is:

$$F = -k_s X - k_d v$$

Dampening Force

```
float y;
float vy;
float m = 1.0;
float ry = 250;
float ks = 0.1;
float kd = 0.1;

void setup() {
  size(500, 500);
}
```

```
void draw() {
  background(210);

  float f = -((ks * (y - ry)) + kd*vy);
  float a = f/m;
  vy = vy + a;
  y += vy;

  rect(200, y, 100, 20);
}
```

Further Extensions

- Fixed-length springs (springs that have a resting distance between the end positions) resemble physical-world springs
- Multi-part system of springs resemble ropes and cords etc

Uses of Springs

- A sequence of particles can simulate:
  - Hair
  - Rope
  - Grass
- A network of particles can simulate:
  - Cloth

Hands-on: Using Masses and Springs

- Today's activities:
    1. Implement the basic mass example using PVectors
    2. Implement the basic spring example using PVectors, so the spring can move in arbitrary directions
    3. Bonus: Create a sequence of multiple particles connected by springs, where each particle's position is based on the previous particle's position. Include mouse controls, so the sequence can be moved around the screen