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

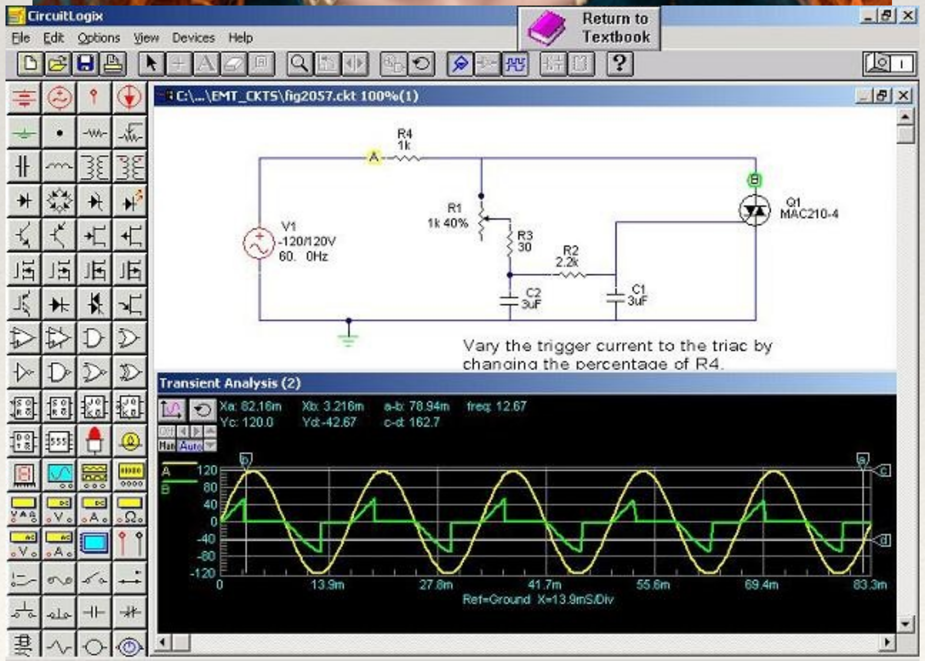
# Simulation and Particle Systems

Elements of Graphics
CS324e

# What is Simulation?

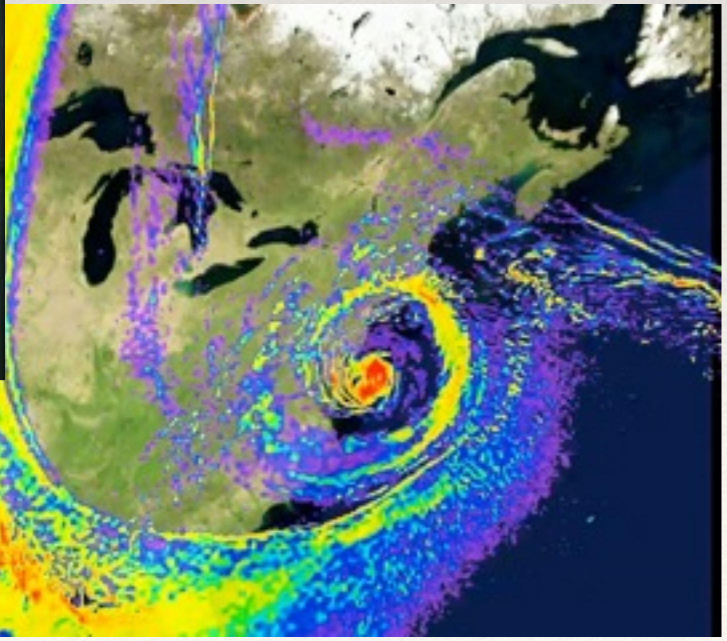
- The capture of behaviors based on rules over time
- Physical simulation
  - Models natural phenomena
  - Physics, chemistry, astronomy, climatology etc
- Operational simulation
  - Models processes and human interactions
  - Economics, manufacturing, engineering etc

# Uses for Simulation



Vary the trigger current to the triac by changing the percentage of R4.

```
Xa: 82.16m   Xb: 3.216m   a-b: 78.94m   freq: 12.67
Yc: 120.0    Yd:-42.67    c-d: 162.7
```



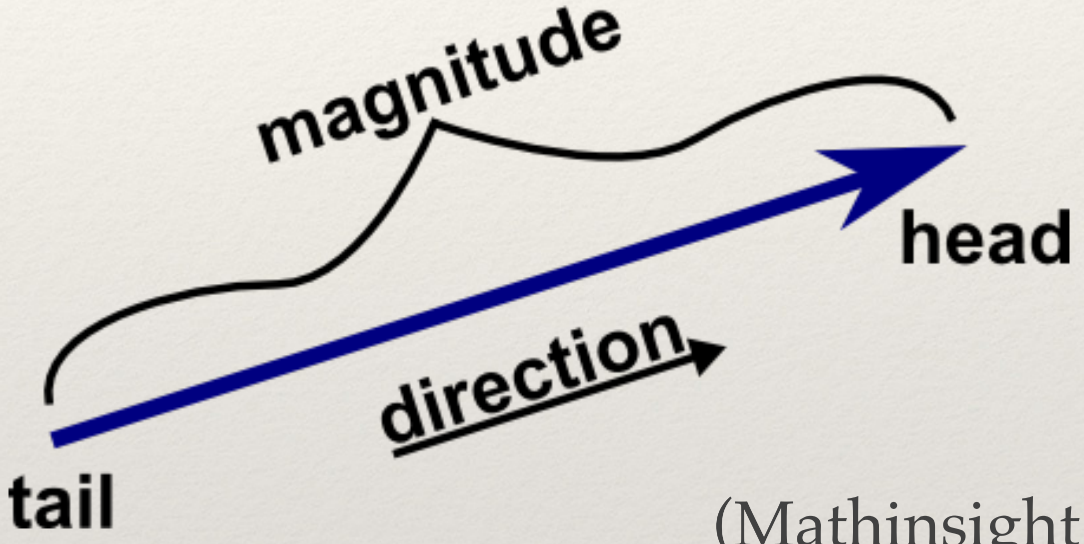
And many, many more…

# Physical Simulation

- Classical mechanics is the study of forces on bodies
- Well understood physical laws
- Velocity is the rate of change in position (m/s)
- Acceleration is the rate of change in velocity ($m/s^2$)

# Vectors

- Velocity and acceleration can both be represented as vectors
- Vectors have:
  - Direction
  - Magnitude



(Mathinsight)

- Positive and negative values in the x and y axes determine the direction
- Magnitude determines the rate of change

# Velocity and Acceleration

- We can set rules/relationships between position, velocity and acceleration to model physical behavior
- For now, we'll work with an intuitive approximation of Newton's laws of motion:
    - Velocity increments position
    - Acceleration increments velocity

# Velocity and Acceleration Example

```
float y = 0.0;
float r = 30.0;
float vel = 0.0;
float accel = 0.03;

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

```
void draw() {
 ellipse(250, y, r, r);
  vel += accel;
  y += vel;
  if (y > height) {
    y = 0.0;
  }
}
```

# Observations

- Changes in position are initially small but increase throughout the simulation
- The object's speed will increase indefinitely (there is no limit on either acceleration or velocity)
- There is no concept of object mass so we are not conserving momentum (F != ma in our example)
- We need more rules to create more interesting behavior

# Question

- What are additional forces we can apply to our object?

# Other Forces

- Gravity is a downward force
- Friction is an opposing force (vector points in the opposite direction of the velocity)
- Spring compression is the force needed to compress a spring (Hooke's Law)
- Coefficient of restitution describes the loss of energy upon collision

# Adding Restitution

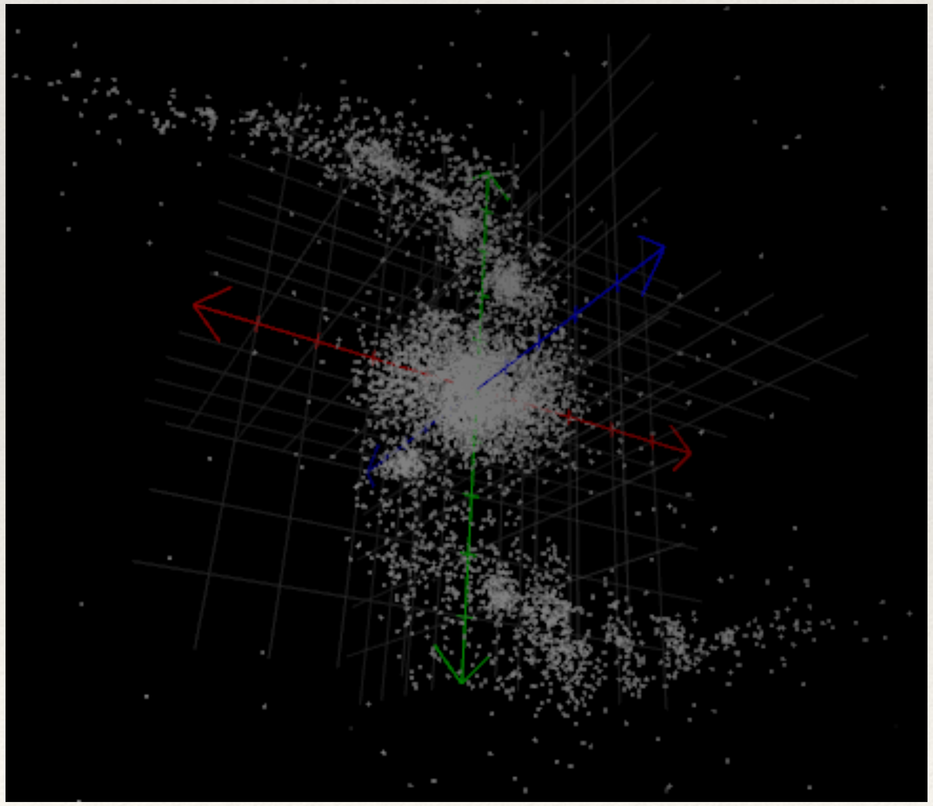
```
float y = 50.0;
float r = 15.0;
float vel = 0.0;
float accel = 0.03;
float friction = 0.995;

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

```
void draw() {
  background(210);
  ellipse(250, y, r, r);
  vel += accel;
  vel *= friction;
  y += vel;
  if (y > (height - r)) {
    vel = -vel;
  }
}
```

# Particle Systems

- System dictating movement of particles within the world
- Simulation of water, smoke, fire, clouds, dust, cloth, crowds, galaxies etc



(PyParticles)

(Houdini)

# Example Particle Class

```
class Particle {
  PVector pos;
  PVector vel;
  float r;

  Particle(float x, float y, float vx, float vy, float r) {
    pos = new PVector(x, y);
    vel = new PVector(vx, vy);
    this.r = r;
  }

  void applyForces(float fx, float fy) {
    vel.x += fx;
    vel.y += fy;
    pos.x += vel.x;
    pos.y += vel.y;
  }

  void display() {
    ellipse(pos.x, pos.y, r, r);
  }
}
```

# Discuss…

- What properties do Particle objects have?
- How can we use Particle objects in our main draw loop?

# Extending the Particle Class

- Additional rules can create increasingly complex behaviors and visualizations:
    - Continuous generation of particles
    - Changes to particle appearance
    - Application of additional forces or functions on the particles
- Note that these things don't have to be physically-based!

# Instapoll Question: Forces

- Name at least 2 forces that affect particle movement

# Hands-on: Forces and Particle Systems

- Today's activities:
  1. Implement the base Particle class. Create a single particle that moves according to the `applyForces` method
  2. Extend the particle class to create one of the following behaviors: 1) a fountain of particles that continuously "respawns" after they are off the screen, or 2) a fountain of particles that bounce against the sides of the screen