Simulation and Particle Systems

Elements of Graphics
CS324e
Fall 2017
Student Presentation
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

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/s²)
Vectors

- Velocity and acceleration can both be represented as vectors
- Vectors have:
  - Direction
  - Magnitude
- Positive and negative values in the x and y axes determine the direction
- Magnitude determines the rate of change

(Mathinsight)
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;
  }
}
Discuss...

- What is the expected behavior of the previous program?
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 ($F \neq ma$ 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

```cpp
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 += velocity;
    if (y > (height - radius)) {
        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 {
    float x, y;
    float vx, vy;
    float r;

    Particle(float _x, float _y, float _vx, float _vy, float _r) {
        x = _x;
        y = _y;
        vx = _vx;
        vy = _vy;
        r = _r;
    }

    void applyForces(float _fx, float _fy) {
        vy += _fy;
        vx += _fx;
        y += vy;
        x += vx;
    }

    void display() {
        ellipse(x, 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
- None of these things have to be physically-based
Hands-on: Basic Forces and Particle Systems

Today’s activities:

1. Implement the code sample on velocity and acceleration. Play with different values of acceleration.
2. Implement the code sample on restitution and friction. Play with different values of friction.
3. Implement the base Particle class. Create a single particle that moves according to the `applyForces` method.
4. Extend the particle class to create the following behavior: a fountain of particles that continuously “respawns” after they are off the screen, or a fountain of particles that bounce against the sides of the screen.