CMSC 427: Chapter 12
Physically-Based Modeling

Reading: Not covered in our text.
Overview:
- Basic Physics
- Kinematics, Kinetics, Springs, and Integration
- Mass-Spring Systems

Overview
- Basic Physics
- Kinematics, Kinetics, Springs, and Integration
- Mass-Spring Systems
Physics: Basic Concepts

**Basic Issues:**

**Kinematics:** The study of motion (ignoring forces). How does acceleration affect velocity? How does velocity affect position?
- **Particle:** A point-mass. Body rotation ignored.
- **Rigid body:** Rotation of the body needs to be considered.

**Force:** Objects change motion only when forces are applied.
- **Contact vs. field forces:** Hitting a baseball vs. gravity or magnetism.
- **Torque:** Force that induces rotation.
- **Environmental sources:** Friction, buoyancy, drag/lift.

**Kinetics:** (also called Dynamics) The effect of force on motion.

**Non-rigid Objects:**
- **Joints and constraints:** Rag-doll physics, mass-spring systems.
- **Flexible objects:** Soft bodies, meshes, cloth, hair.

**Collisions:** Detection and response.

Rigid Body Properties

**Rigid Body Physics:** For objects under translation/rotation.

**Mass:** (scalar)
- The amount of matter.
- The degree of **resistance to change** in translational motion (inertial mass).

**Center of Mass** (or gravity): (point)
- Central point about which rotations occur.
- Need not lie within the body (if the body is nonconvex).

**Moment of Inertia:** (scalar)
- The resistance to rotational motion about a given axis (scalar form).
- There is a more complex physical quantity, called the **inertial tensor**, which encodes the moment of inertia for all possible rotation axes.

**Particle Physics:** Simpler than rigid-body physics.
- Center of mass = particle position.
- Rotation is ignored. We may ignore moment of inertia and torque.
Overview

• Basic Physics

• Kinematics, Kinetics, Springs, and Integration

• Mass-Spring Systems

Kinematics: Speed and Velocity

**Velocity:** Change in position over time.

**Average Velocity:** Let $s =$ position and $t =$ time. Velocity is the change in position $\Delta s$ over some time interval $\Delta t$:

$$v = \frac{\Delta s}{\Delta t}.$$  

**Instantaneous velocity:** Velocity as a function of time:

$$v = \lim_{\Delta t \to 0} \frac{\Delta s}{\Delta t} = \frac{ds}{dt}.$$  

**Acceleration:** Change in velocity over time.

**Average Acceleration:**

$$a = \frac{\Delta v}{\Delta t}.$$  

**Instantaneous acceleration:** In general, velocity varies with time:

$$a = \lim_{\Delta t \to 0} \frac{\Delta v}{\Delta t} = \frac{dv}{dt} = \frac{d^2s}{dx^2}.$$  

Chapter 12, Slide 5
Copyright © D. M. Mount and A. Varshney
Force

**Kinematics:**
- The study of motion in the absence of force.

**Kinetics:**
- How do we integrate forces with mass to determine motion?

**Force Basics:**
- **Contact force:** from impact, friction, buoyancy, pressure.
- **Field force:** from gravity and electromagnetism.
- **Newton’s Third Law:** Forces come in pairs (action and reaction).
  
  Usually compute just one, and the other is its negation.

**Examples:**
- Springs - useful for handling collisions.
- Friction - braking, skidding, sliding.
- Dampers - motion resistors
- Bouyancy - floating

Springs

**Springs:**
- Elements (usually particles) joined by elastic structures, called springs.
- Springs assumed to follow Hooke’s Law (given below).
- Complex mass-spring systems can be used to model complex objects, such as cloth.
Example: Modeling a Hanging Rope

Example: Modeling a rope as a spring system.
- Springs link particles. The more particles, the smoother the approximation.
- Endpoint locations are fixed. The middle particle positions are determined by a balance between gravity and spring forces.
- Springs try to maintain their rest lengths and so preserve the length of the string.

Springs

Hooke’s Law: for an ideal (linear) spring.
- Let \( p_1 \) and \( p_2 \) be two particles connected by a spring.
- Let \( L = \| p_1 p_2 \| \) be the distance between \( p_1 \) and \( p_2 \).
- Let \( u = \text{unit length directional vector} \) from \( p_1 \) to \( p_2 \).
- Then the spring force is (vector quantity):
  \[
  F_{\text{spring}} = k (L - L_{\text{rest}}) u,
  \]
  where:
  - \( L_{\text{rest}} \) = the length of the spring at rest, and
  - \( k \) = spring constant. (Units: force/length, e.g. lb/ft or Newton/m.)

Application:
- \( F_{\text{spring}} \) is applied to \( p_1 \).
- \(-F_{\text{spring}}\) is applied to \( p_2 \).
- \( L < L_{\text{rest}} \): repels the points.
- \( L > L_{\text{rest}} \): attracts the points.
Integration

integration:  
- The process of applying physical laws (in the form of differential equations) to determine the positions, orientations, and velocities of the objects in your scene over time.

numerical integration:  
- Approximating differential constraints in small time steps.
- Can deal with complex constraint systems.
- Stability is an issue. Vicious cycle:  
  • To increase accuracy, make time steps smaller →
  • Smaller time steps require more total steps →
  • More steps imply more accumulated error → less accuracy.
- Common methods:  
  • Euler integration (fast and simple).
  • Verlet integration (slower, but more accurate).

Stability

stability: Numerical integration involves many time steps, each incurring a small error. These can accumulate over time, resulting in wildly inaccurate results.

Example: Consider a particle that is orbiting a central point. Each step moves the particle perpendicular to the center. Over time the particle spirals outwards.
Euler Integration

### Good
Simple and well known, easy to implement.

### Bad
Not very accurate.

### How it Works
Consider the following vector quantities:
- \( F(t) \): Force acting on particle at time \( t \).
- \( a(t) \): Acceleration at time \( t \). Computed from force and mass:
  \[ a(t) = \frac{F(t)}{m} \]
  where \( m \) is mass.
- \( v(t) \): Velocity at time \( t \).
- \( s(t) \): Position at time \( t \).

### Euler update rules:
\[
\begin{align*}
\Delta s(t) & \equiv s(t + \Delta t) - s(t) \\
\Delta v(t) & \equiv v(t + \Delta t) - v(t) \\
\Delta a(t) & \equiv a(t + \Delta t) - a(t) \\
\end{align*}
\]

### Euler Integration: Pseudocode

#### Initialization:
- \( p.pos \leftarrow p.initialPosition() \)
- \( p.veloc \leftarrow p.initialVelocity() \)

#### Update:
(Compute state at time \( t + \Delta t \) from values at time \( t \))
- \( p.pos \leftarrow p.pos + \Delta t \cdot p.veloc \)
- \( p.veloc \leftarrow p.veloc + \Delta t \cdot p.force/p.mass \)
- \( p.force \leftarrow p.updateForces() \)

#### Advantages:
- Easy to implement.
- Very fast.

#### Disadvantages:
- Not very accurate: Errors can accumulate rapidly.
- Inconsistency: Velocities and positions can be inconsistent.
- Instability: Can be unstable for stiff equations.
Verlet Integration

Verlet Integration:
- More accurate than the Euler integration.
- The next step is based on the prior two steps.

Verlet Integration Rule:
- Let s(t) be current position and s(t−Δt) be previous position.
- Let a(t) be the current acceleration (determined by a(t) = F(t)/m).
- Update rule:
  \[ s(t + Δt) = 2 \cdot s(t) - s(t - Δt) + \frac{d^2}{dt^2} s(t) \Delta t^2 \]
  \[ = 2 \cdot s(t) - s(t - Δt) + a(t) \Delta t^2. \]
- For cloth, we may want to add some damping to this by decreasing the displacement. Give a small positive damping factor δ:
  \[ s(t + Δt) = s(t) + (1 - δ)(s(t) - s(t - Δt)) + a(t) \Delta t^2. \]

Advantages:
- More accurate than Euler integration.

Disadvantages:
- Need an Euler integrator to obtain the 2nd time step.
- Time step sizes need to be uniform.

Verlet Integration: Pseudocode

Initialization:
\[
p.\text{oldPos} \leftarrow p.\text{pos} \leftarrow p.\text{initialPosition}() \]

Update Rule: (Compute state at time t+Δt from prior two states)
\[
\text{temp} \leftarrow p.\text{pos} \\
p.\text{pos} \leftarrow p.\text{pos} + (1 - δ) \cdot (p.\text{pos} - p.\text{oldPos}) + p.\text{accel} \cdot (Δt^2) \\
p.\text{oldPos} \leftarrow \text{temp} \\
p.\text{force} \leftarrow p.\text{updateForces}() \\
p.\text{accel} \leftarrow p.\text{force} / p.\text{mass}
\]

Advantages:
- More accurate than Euler integration.

Disadvantages:
- Need an Euler integrator to obtain the 2nd time step.
- Time step sizes need to be uniform.
Overview

- Basic Physics
- Kinematics, Kinetics, Springs, and Integration
  - Mass-Spring Systems

Mass-Spring Systems

Mass-Spring System:
- Used for modeling string, cloth, hair, etc.
- Models local interactions between a collection of particles.
- Implemented by creating a network of spring forces, each of which links pairs of particles.

Image Michael Kass
Mass-Spring Systems: Types of Forces

**Structural forces:**
- Enforce invariant properties of the system.
- Ideally, these should be hard constraints, but it is easier to implement them as forces.
- Examples: Fixing the length of a string. A cloth should not pass through itself.

**Internal deformation forces:**
- Enforce deformational properties of the system (e.g., stiffness vs. flexibility).
- Example: Strings deform easily. A diving board is much stiffer.

**External forces:**
- Enforce environmental forces imposed on the system.
- Examples: Collisions with external objects, gravity, wind.

---

**Example: Hair**

Simple Hair Model:
- **Topology:** Linear sequence of particles.
- **Structural forces:** Enforce distances between particles, and anchor root to skin.
- **Deformation forces:** Proportional to the angle between segments to enforce stiffness.
- **External forces:** Gravity, wind.
Example: Cloth

Cloth Model:

**Topology:** Three types of springs. Let \([x, y]\) be a point of the mesh.

**Structural Constraints:** To: \([x \pm 1, y], [x, y \pm 1]\).
Resist stretching of the cloth.

**Shearing Constraints:** To: \([x \pm 1, y \pm 1]\).
Resist shearing, and preserve orthogonality of the mesh.

**Bending Constraints:** To: \([x \pm 2, y], [x, y \pm 2]\).
Resist bending, enforce stiffness.

Cloth Simulation

**Satisfying Constraints:**
- By Hooke’s Law, a spring exerts a force to restore its rest length.
- We will cheat, by computing a correction vector, which restores each spring to its exact rest length.
- Each vertex is subject to many such pulls, and thus its final position not at rest, but the average of these effects.

**Fixing Spring Constraint:** Given a spring \(s = (p_1, p_2)\)

```plaintext
Vector v ← p_2 - p_1 // vector from p_1 to p_2
float d ← ||v|| // distance from p_1 to p_2
float d_0 ← s.restLength // spring’s length at rest
Vector c ← (1 - d_0/d) · v // correction vector
Vector h ← c / 2 // half the correction vector
p_1 ← p_1 + h // adjust point positions
p_2 ← p_2 - h
```

Repeated for each spring. May be repeated a few times.
Cloth Simulation: Programming Tips

Implementation Tips:
- In addition to the spring constraints, particles are subject to other forces, which can be updated using Euler or Verlet integration.
- Each particle computes any collision with objects in the scene. Such a collision induces a force on the particle (i.e., increases its acceleration) in the direction of the object’s velocity.
- Gravity also induces a force, which increases the particle’s downward velocity.
- Some cloth particles may be defined as unmovable, which means that their positions never change. Used to anchor the cloth.

Rendering Tips:
- Render the cloth using GL_QUADS or GLTriangle_STRIP.
- To obtain smooth shading, compute each vertex normal as the average the normal vectors of the incident faces.
- Disable backface culling.

Summary

Summary:
- Basic Physics
- Kinematics, Kinetics, Springs, and Integration
- Mass-Spring Systems