Physics based animation

Lecture 05 - Mass spring systems

Grégory Leplâtre

g.leplatre@napier.ac.uk, room D32
School of Computing
Edinburgh Napier University

Semester 1 - 2016/2017

Objectives

- Reminder: spring dynamics

Objectives

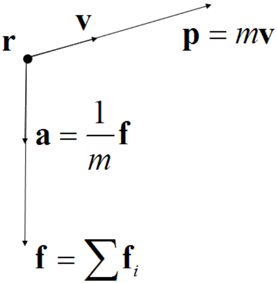
- Reminder: spring dynamics
- Applications: Mass spring systems

Objectives

- Reminder: spring dynamics
- Applications: Mass spring systems
- Limitations and alternatives

Outline

Reminder: particles



- ***r***: position
- ***v***: velocity
- ***a***: acceleration
- *m*: mass
- ***p***: momentum
- ***f***: force

Springs

Simple spring force (Hooke's law):

$$\boldsymbol{f}_{spring} = -k_s \boldsymbol{x}$$

Where:

- k_s is a constant describing the **stiffness** of the spring
- $\boldsymbol{x}$ represents the displacement

Dampers

Damping force between particles:

$$\boldsymbol{f}_{damp} = -k_d \boldsymbol{v}$$

Simulation

1. Once all forces have been computed, Newton's second law gives us the acceleration:

$$\boldsymbol{a}_n = \frac{1}{m}\boldsymbol{f}_n$$

Simulation

1. Once all forces have been computed, Newton's second law gives us the acceleration:

$$\boldsymbol{a}_n = \frac{1}{m}\boldsymbol{f}_n$$

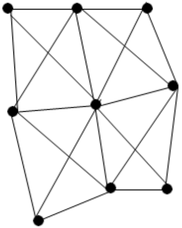
2. Using an appropriate integration method, we computer the **velocity** and **position** from the acceleration (here, using semi-implicit Euler):

$$v_{n+1} = v_n + ha_n$$

$$r_{n+1} = r_n + hv_{n+1}$$

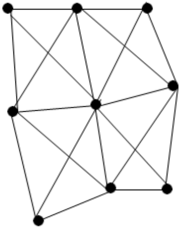
Outline

Cloth simulation with springs



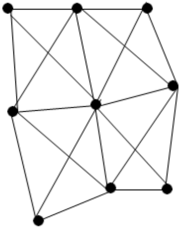
- Cloth can be simulated by a system of particles interconnected with **spring-dampers**.

Cloth simulation with springs



- Cloth can be simulated by a system of particles interconnected with **spring-dampers.**
- Each particle is also influenced by gravity

Cloth simulation with springs



- Cloth can be simulated by a system of particles interconnected with **spring-dampers**.
- Each particle is also influenced by gravity
- Other forces can also be added for more interesting behaviours:
  - Aerodynamic drag
  - bending resistance
  - plastic, elastic deformations and tearing.

Cloth simulation

1. Compute Forces
   - For each particle: apply gravity
   - For each spring-damper: compute and apply forces
   - For each triangle: compute and apply aerodynamic forces

Cloth simulation

1. Compute Forces
   - For each particle: apply gravity
   - For each spring-damper: compute and apply forces
   - For each triangle: compute and apply aerodynamic forces
2. Integrate
   - Using an appropriate integration method!

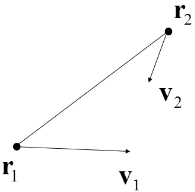
1. Gravity

Uniform gravity field:

$$\boldsymbol{f}_{gravity} = m\boldsymbol{g}_0$$

$$\boldsymbol{g}_0 = [0, -9.8, 0]\ m.s^{-2}$$

2. Spring-Dampers



- Spring constant: k_s
- Damping factor: k_d
- Rest length: l_0

2. Spring-Dampers

- Linear spring force in one dimension:
  $f_{spring} = -k_s x = -k_s(l_0 - l)$

2. Spring-Dampers

- Linear spring force in one dimension:
  $f_{spring} = -k_s x = -k_s(l_0 - l)$
- Linear damping force:
  $f_{damp} = -k_d v = -k_d(v_1 - v_2)$

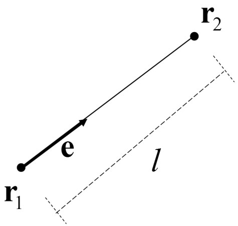
2. Spring-Dampers

- Linear spring force in one dimension:
  $f_{spring} = -k_s x = -k_s(l_0 - l)$
- Linear damping force:
  $f_{damp} = -k_d v = -k_d(v_1 - v_2)$
- spring-damper force: $f_{sd} = -k_s(l_0 - l) - k_d(v_1 - v_2)$

2. Spring-Dampers

- To compute the forces in 3D:
  - Turn 3D distances and velocities into 1D
  - Compute spring forces in 1D
  - Turn 1D force back into 3D force

2. Spring-Dampers



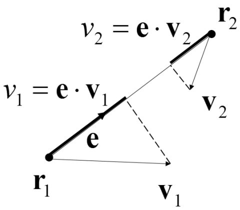
- Compute the unit vector $\boldsymbol{e}$ from $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$

$$\boldsymbol{e} = \frac{\boldsymbol{r}_2 - \boldsymbol{r}_1}{\|\boldsymbol{r}_2 - \boldsymbol{r}_1\|}$$

- Compute distance between the two points: l

$$\boldsymbol{e} = \|\boldsymbol{r}_2 - \boldsymbol{r}_1\|$$

2. Spring-Dampers

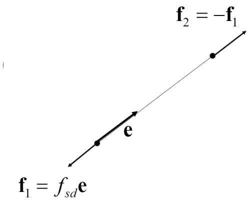


- Then compute 1D velocities:

$$v_1 = \boldsymbol{e} \bullet \boldsymbol{v_1}$$

$$v_2 = \boldsymbol{e} \bullet \boldsymbol{v_2}$$

2. Spring-Dampers



- Then compute 1D forces and map then back into 3D:

$$f_{sd} = -k_s(l_0 - l) - k_d(v_1 - v_2)$$

$$\boldsymbol{f_1} = f_{sd}\boldsymbol{e}$$

$$\boldsymbol{f_2} = -\boldsymbol{f_1}$$

3 - Aerodynamic force

Reminder from last lecture:

$$\boldsymbol{f}_{aero} = \frac{1}{2}\rho \|\boldsymbol{v}\|^2 c_d a \boldsymbol{e}$$

Where:

- ρ is the density of the medium (air, water, etc)
- c_d is the coefficient of drag of the object
- a is the cross sectional area of the object.
- $\boldsymbol{e} = -\frac{\vec{v}}{\|\vec{v}\|}$

3 - Aerodynamic force

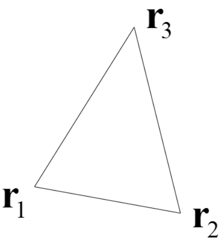
Further (major) simplification:

$$\boldsymbol{f}_{aero} = \frac{1}{2}\rho\|\boldsymbol{v}\|^2 c_d a \boldsymbol{n}$$

Where:

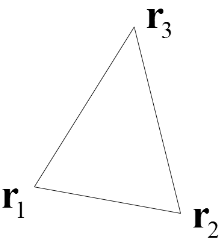
- Instead of opposing the velocity, the force pushes against the normal of the surface: $\boldsymbol{n}$

3. Aerodynamic force



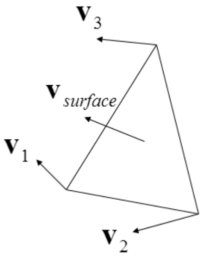
- In order to compute the force, a surface of application must be defined.
- Triangles are added to our particle network.
- In practice, the mesh would already exist and have been triangulated.

3. Aerodynamic force



- We need three things:
  - The velocity: $\boldsymbol{v}$
  - The area of the triangle: a
  - The normal of the surface: $\boldsymbol{n}$

3. Aerodynamic force



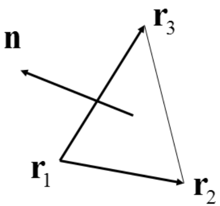
- For the **velocity**, we can average the velocities of the three particles:

$$\mathbf{v}_{surface} = \frac{\mathbf{v}_1 + \mathbf{v}_2 + \mathbf{v}_3}{3}$$

- If wind/air movement is present, it can be taken into account:

$$\mathbf{v} = \mathbf{v}_{surface} - \mathbf{v}_{air}$$

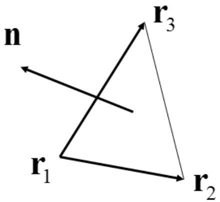
3. Aerodynamic force



- The **Normal** of the triangle:

$$\boldsymbol{n} = \frac{\boldsymbol{r}_2 - \boldsymbol{r}_1 \times \boldsymbol{r}_3 - \boldsymbol{r}_1}{\|\boldsymbol{r}_2 - \boldsymbol{r}_1 \times \boldsymbol{r}_3 - \boldsymbol{r}_1\|}$$

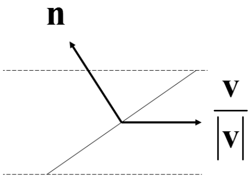
3. Aerodynamic force



- The **area** of the triangle:

$$a_0 = \frac{1}{2}\|\boldsymbol{r}_2 - \boldsymbol{r}_1 \times \boldsymbol{r}_3 - \boldsymbol{r}_1\|$$

3. Aerodynamic force



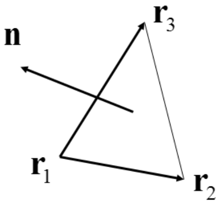
- The **area** of the triangle:

$$a_0 = \frac{1}{2}\|\boldsymbol{r}_2 - \boldsymbol{r}_1 \times \boldsymbol{r}_3 - \boldsymbol{r}_1\|$$

- The cross-sectional area (exposed to air flow):

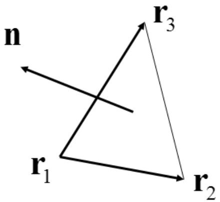
$$a = a_0 \frac{\boldsymbol{v} \bullet \boldsymbol{n}}{\|\boldsymbol{v}}\|$$

3. Aerodynamic force



- The force we have calculated applies to **one triangle.**
- Divide by three $\rightarrow$ force on each corner particle.

3. Aerodynamic force



- The force we have calculated applies to **one triangle.**
- Divide by three $\rightarrow$ force on each corner particle.

Recap

- Effective and efficient solution

Recap

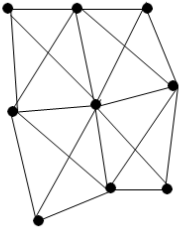
- Effective and efficient solution
- But we can take things further

Recap

- Effective and efficient solution
- But we can take things further
  - Collisions
  - Material stiffness
  - Different types of deformation
  - 3D systems: soft bodies
  - Alternatives?

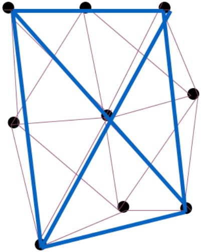
Bending Forces

Bending forces



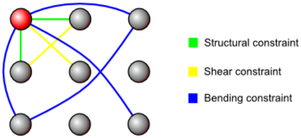
- If we arrange our cloth springs as they are in the picture, there will be nothing preventing the cloth from bending
- This may be find for simulating softer cloth, but for stiffer materials, we may want some resistance to bending

Bending forces



- A simple solution is to add more springs, arranged in various configurations, such as the one in the picture
- The spring constants and damping factors of this layer might need to be tuned differently...

Bending forces



- A simple solution is to add more springs, arranged in various configurations, such as the one in the picture
- The spring constants and damping factors of this layer might need to be tuned differently...

Deformation types

Deformation types

- An **elastic** deformation will **restore back** to its un-deformed state when all external forces are removed (such as the deformation in a spring, or in a rubber ball)

Deformation types

- An **elastic** deformation will **restore back** to its un-deformed state when all external forces are removed (such as the deformation in a spring, or in a rubber ball)
- A **plastic** deformation is a **permanent adjustment** of the material structure (such as the buckling of metal)

Plastic and elastic deformations

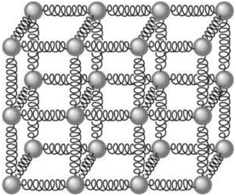
- We can add a simple **plastic deformation** rule to the spring-dampers. We do so by **modifying the rest length**
- Several possible rules can be used, but one simple way is to start by defining an **elastic limit** and **plastic limit**
  - The **elastic limit** is the **maximum deformation** distance allowed **before a plastic deformation occurs**.
  - If the elastic limit is reached, the rest length of the spring is adjusted so that meets the elastic limit
  - An additional plastic limit prevents the rest length from deforming beyond some value
  - The plastic limit defines the maximum distance we are allowed to move the rest length

Fracture and tearing

- We can also allow springs to break
- One way is to define a length (or percentage of rest length) that will cause the spring to break
- This can also be combined with the plastic deformation, so that **fracture occurs at the plastic limit**

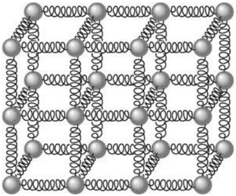
Outline

Taking things further



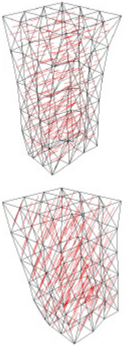
- Can be expanded to 3D structures (deformable/soft bodies)

Taking things further



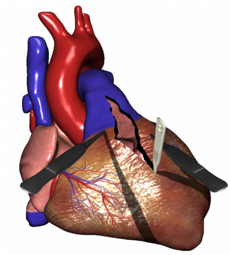
- Can be expanded to 3D structures (deformable/soft bodies)
- Volume preservation

Taking things further



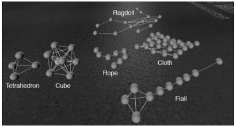
- Can be expanded to 3D structures (deformable/soft bodies)
- Volume preservation
- Anisotropy

Taking things further



- Can be expanded to 3D structures (deformable/soft bodies)
- Volume preservation
- Anisotropy
- Application to muscle deformations (cardiac surgery simulation by Jesper Mosegaard)

Taking things further



- Can be expanded to 3D structures (deformable/soft bodies)
- Volume preservation
- Anisotropy
- Application to muscle deformations (cardiac surgery simulation by Jesper Mosegaard)
- etc

Outline

Summary: Mass spring systems

- Pros:
  - Fast
  - GPU-friendly

Summary: Mass spring systems

- Pros:
  - Fast
  - GPU-friendly
- Cons
  - Difficult to map physical properties onto spring properties
  - Volume not considered
  - Stability issues depending on **integration method** used and size of **step**

Summary: Mass spring systems

- Pros:
  - Fast
  - GPU-friendly
- Cons
  - Difficult to map physical properties onto spring properties
  - Volume not considered
  - Stability issues depending on **integration method** used and size of **step**
- More advanced alternative: Finite Elements Method

Coming up

- Practical: spring simulation. Why simulate one when you could simulate a whole lattice (cloth)?!
- Next week: Solid bodies and **class test ...**
- Following week: **project pitch**
- Subsequently: **collisions**

References

- Bourguignon, D., & Cani, M. P. (2000). Controlling anisotropy in mass-spring systems. In Computer Animation and Simulation 2000 (pp. 113-123). Springer Vienna.
- da Silva, J. P., Giraldi, G. A., & Apolinário Jr, A. L. (2015). A new optimization approach for mass-spring models parameterization. Graphical Models, 81, 1-17.
- Hong, M., Jung, S., Choi, M. H., & Welch, S. W. (2006). Fast volume preservation for a mass-spring system. IEEE Computer Graphics and applications, 26(5), 83-91.
- Hutchinson, D., Preston, M., & Hewitt, T. (1996). Adaptive refinement for mass/spring simulations. In Computer Animation and Simulation'96 (pp. 31-45). Springer Vienna.
- Liu, T., Bargteil, A. W., O'Brien, J. F., & Kavan, L. (2013). Fast simulation of mass-spring systems. ACM Transactions on Graphics (TOG), 32(6), 214.
- Nealen, A., Muller, M., Keiser, R., Boxerman, E., & Carlson, M. (2006, December). Physically based deformable models in computer graphics. In Computer graphics forum (Vol. 25, No. 4, pp. 809-836). Blackwell Publishing Ltd.
- Nedel, L. P., & Thalmann, D. (1998, June). Real time muscle deformations using mass-spring systems. In Computer Graphics International, 1998. Proceedings (pp. 156-165). IEEE.