CSC317 Computer Graphics
Tutorial 8

November 13, 2024

Assignment 8: Mass Spring System

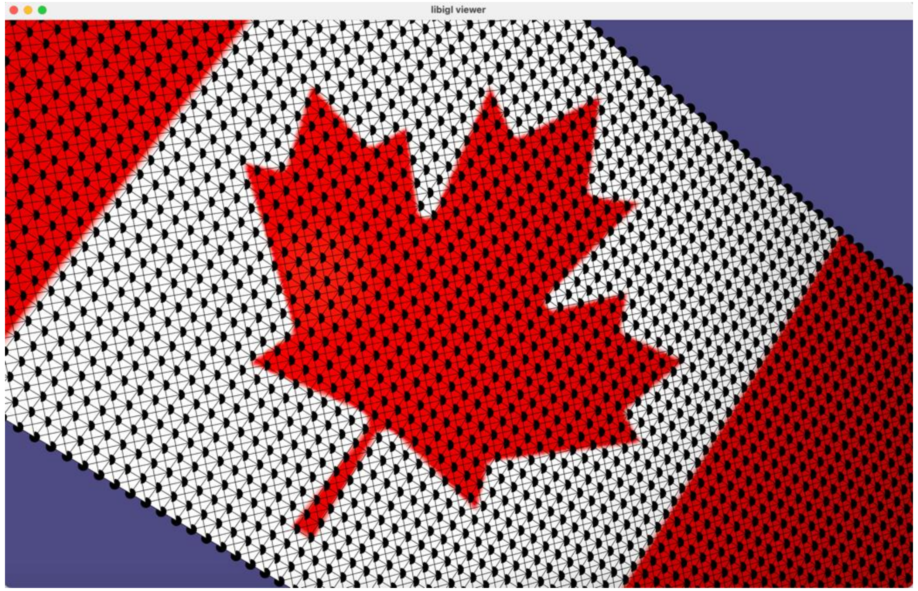
- Due Date: November **26** @ 11:59 pm
- Assignment description can be accessed through [course github page](), under "Lecture Schedule"

Week	Topic / Event
1	IntroductionDL,KS, RGBtoHSV,tutorial, Assignment 1 (Raster Images) Math Practice**waitlisted ? zip assignment and email to TAs** due 17/09
2	Lecture2DL,KS, Assignment 2 Ray Casting due 24/09
3	Lecture 3DL,KS, Assignment 3 Ray Tracing due 01/10
4	Lecture 4DL,KS, Assignment 4 Bounding Volume Hierarchy due 8/10
5	Lecture 5DL,KS, Assignment 5 Meshes due 22/10
6	**No Lecture, Thanksgiving**
	First In-Tutorial Test October 16th
7	Lecture 6,DL,KS, Assignment 6 Shader Pipeline due 5/11
8	**No Lecture, Reading Week !**
9	Lecture 7, DL, KS, Assignment 7 Kinematics due ~~12/11~~ 19/11
10	Lecture 8, Assignment 8 Mass-Spring Systems due ~~19/11~~ 26/11
11	Lecture 9
12	Second In-Class Test December 3rd

Assignment 8: Mass Spring System

- Goal: Animating deformable shapes

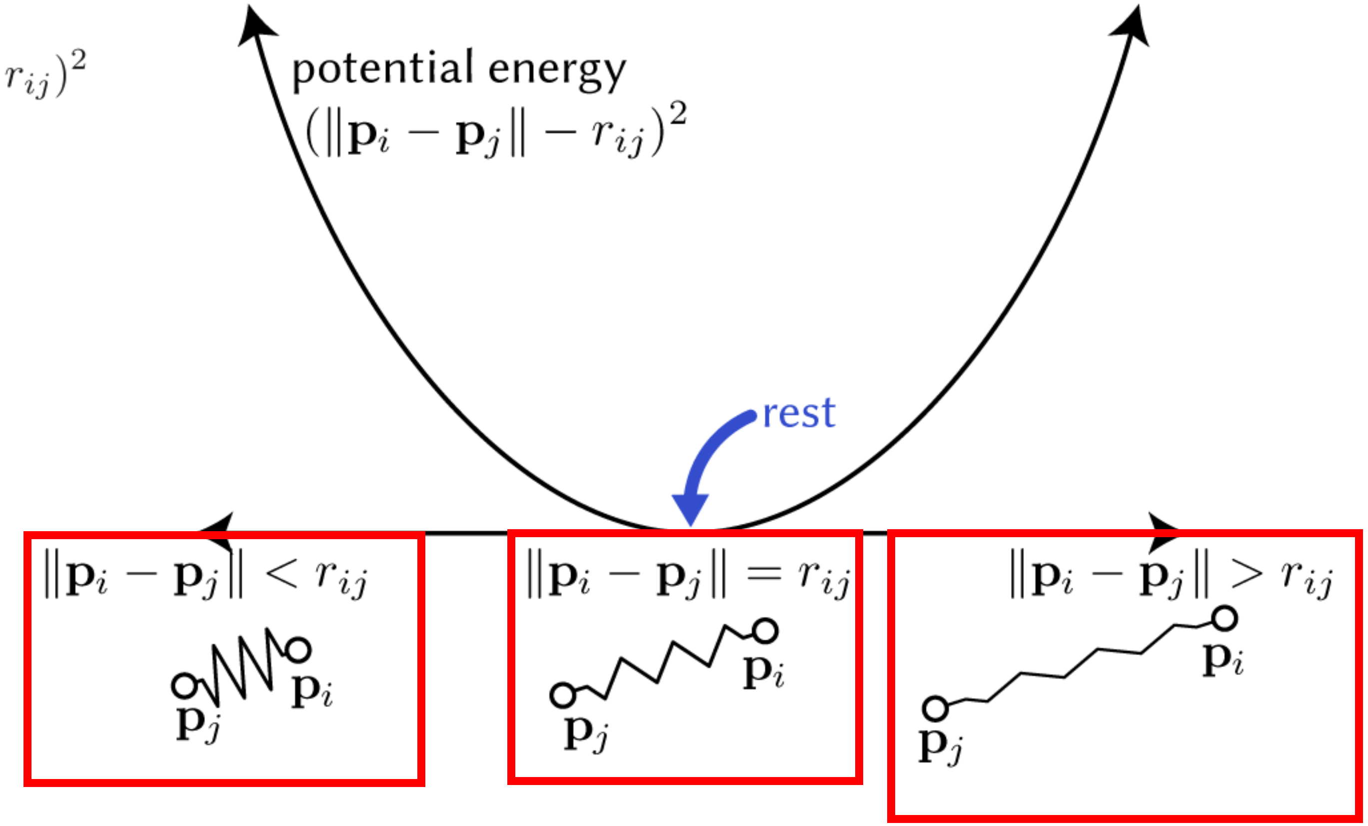
- Where does mass-spring system come into play?
  - We treat the system as a network of point masses and springs



Assignment 8: Mass Spring System

- Potential energy of a spring:

$$V(\mathbf{p}_i, \mathbf{p}_j) = \frac{1}{2}k(\|\mathbf{p}_i - \mathbf{p}_j\| - r_{ij})^2$$



Assignment 8: Mass Spring System

- Newton's second law: **f = ma**

$$\mathbf{a}_i^t = \ddot{\mathbf{p}}_i^t = \frac{\partial^2 \mathbf{p}_i(t)}{\partial t^2} = \frac{\dot{\mathbf{p}}_i^{t+\Delta t} - \dot{\mathbf{p}}_i^t}{\Delta t} = \frac{\mathbf{p}_i^{t+\Delta t} - \mathbf{p}_i^t}{\Delta t^2} - \frac{\mathbf{p}_i^t - \mathbf{p}_i^{t-\Delta t}}{\Delta t^2} = \frac{\mathbf{p}_i^{t+\Delta t} - 2\mathbf{p}_i^t + \mathbf{p}^{t-\Delta t}}{\Delta t^2}$$

- Relationship between potential energy and force:

Non-linear!

$$\mathbf{f}_{ij} = -\boxed{\frac{\partial V}{\partial \mathbf{p}_i}} \in \mathbf{R}^3.$$

Assignment 8: Mass Spring System

- Alternative setup: energy minimization

Potential Energy

Kinetic Energy

External Force

$$\mathbf{p}^{t+\Delta t} = \underset{\mathbf{p}}{\operatorname{argmin}} \underbrace{\left(\sum_{ij} \frac{1}{2} k(\|\mathbf{p}_i - \mathbf{p}_j\| - r_{ij})^2\right) + \frac{\Delta t^2}{2}\left(\sum_i m_i \left(\frac{\mathbf{p}_i - 2\mathbf{p}_i^t + \mathbf{p}_i^{t-\Delta t}}{\Delta t^2}\right)^2\right) - \left(\sum_i \mathbf{p}_i^\top \mathbf{f}_i^{\text{ext}}\right)}_{E(\mathbf{p})}$$

Assignment 8: Mass Spring System

- Setting up the system of equations:

$$\begin{aligned}\tilde{E}(\mathbf{p}) &= \frac{k}{2}\operatorname{tr}\left((\mathbf{A}\mathbf{p}-\mathbf{d})^{\top}(\mathbf{A}\mathbf{p}-\mathbf{d})\right) + \frac{1}{2\Delta t^2}\operatorname{tr}\left(\left(\mathbf{p}-2\mathbf{p}^{t}+\mathbf{p}^{t-\Delta t}\right)^{\top}\mathbf{M}\left(\mathbf{p}-2\mathbf{p}^{t}+\mathbf{p}^{t-\Delta t}\right)\right) - \operatorname{tr}\left(\mathbf{p}^{\top}\mathbf{f}^{\mathrm{ext}}\right)\\ &= \frac{1}{2}\operatorname{tr}\left(\mathbf{p}^{\top}(k\mathbf{A}^{\top}\mathbf{A} + \frac{1}{\Delta t^2}\mathbf{M})\mathbf{p}\right) - \operatorname{tr}\left(\mathbf{p}^{\top}(k\mathbf{A}^{\top}\mathbf{d} + \frac{1}{\Delta t^2}\mathbf{M}(2\mathbf{p}^{t}-\mathbf{p}^{t-\Delta t}) + \mathbf{f}^{\mathrm{ext}})\right) + \text{ constants}\end{aligned}$$

- Re-arranging:

$$\mathbf{p}^{t+\Delta t} = \underset{\mathbf{p}}{\operatorname{argmin}}\ \frac{1}{2}\operatorname{tr}\left(\mathbf{p}^{\top}\mathbf{Q}\mathbf{p}\right) - \operatorname{tr}\left(\mathbf{p}^{\top}\mathbf{b}\right)$$

$$\frac{\partial \operatorname{tr}\left(\mathbf{x}^{\top}\mathbf{y}\right)}{\partial \mathbf{x}} = \mathbf{y}$$

- Taking the derivative:

$$\mathbf{Q}\mathbf{p} = \mathbf{b}.$$

$$\frac{\partial \frac{1}{2}\operatorname{tr}\left(\mathbf{x}^{\top}\mathbf{Y}\mathbf{x}\right)}{\partial \mathbf{x}} = \mathbf{Y}\mathbf{x}$$

signed_incidence_matrix_dense

```
// Inputs:
//   n  number of vertices (#V)
//   E  #E by 2 list of edge indices into rows of V
// Outputs:
//   A  #E by n signed incidence matrix
```

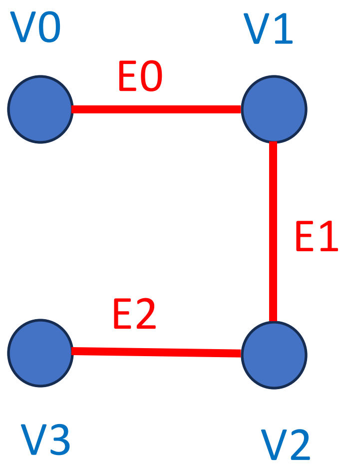
$$\mathbf{A}_{ek} = \begin{cases} +1 & \text{if } k = i \\ -1 & \text{else if } k == j \\ 0 & \text{otherwise.} \end{cases}$$

$$\mathbf{Q} := (k\mathbf{A}^\top \mathbf{A} + \frac{1}{\Delta t^2}\mathbf{M})$$
$$\mathbf{b} := k\mathbf{A}^\top \mathbf{d} + \mathbf{y} \in \mathbf{R}^{n \times 3}$$

signed_incidence_matrix_dense

```
// Inputs:
//   n  number of vertices (#V)
//   E  #E by 2 list of edge indices into rows of V
// Outputs:
//   A  #E by n signed incidence matrix
```

$$\mathbf{A}_{ek} = \begin{cases} +1 & \text{if } k = i \\ -1 & \text{else if } k == j \\ 0 & \text{otherwise.} \end{cases}$$



E (3 x 2):
0, 1
1, 2
2, 3

A (3 x 4):
1, -1, 0, 0
0, 1, -1, 0
0, 0, 1, -1

fast_mass_springs_precomputation_dense

```
// Precompute matrices and factorizations necessary for the "Fast Simulation of
// Mass-Spring Systems" method.
//
// Inputs:
//   V  #V by 3 list of vertex positions
//   E  #E by 2 list of edge indices into rows of V
//   k  spring stiffness
//   m  #V list of masses
//   b  #b list of "pinned"/fixed vertices as indices into rows of V
//   delta_t  time step in seconds
// Outputs:
//   r  #E list of edge lengths
//   M  #V by #V mass matrix
//   A  #E by #V signed incidence matrix
//   C  #b by #V selection matrix
//   prefactorization  LLT prefactorization of energy's quadratic matrix
```

fast_mass_springs_precomputation_dense

- r: list of edge lengths
- M: diagonal matrix of masses

$$\mathbf{M}_{ij} = \begin{cases} m_i & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

- A: signed incidence matrix (already computed)
- C: selection matrix (set to 1 for every pinned vertex)
- prefactorization: **prefactorization.compute(Q)** does the required "inverse"

$$\mathbf{Q} := (k\mathbf{A}^\top \mathbf{A} + \frac{1}{\Delta t^2}\mathbf{M}) + \mathit{pinned\ vertex\ penalty\ term}$$

Example Selection Matrix

- Select vertices 0, 2, 4:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V0 \\ V1 \\ V2 \\ V3 \\ V4 \end{bmatrix} = \begin{bmatrix} V0 \\ V2 \\ V4 \end{bmatrix}$$

Selection Matrix C   Input Vertices   Output

fast_mass_springs_step_dense

```
// Conduct a single step of the "Fast Simulation of Mass-Spring Systems" method.
//
// Inputs:
//   V  #V by 3 list of **rest** vertex positions
//   E  #E by 2 list of edge indices into rows of V
//   k  spring stiffness
//   b  #b list of indices of fixed vertices as indices into rows of V
//   delta_t  time step in seconds
//   fext  #V by 3 list of external forces
//   r  #E list of edge lengths
//   M  #V by #V mass matrix
//   A  #E by #V signed incidence matrix
//   C  #b by #V selection matrix
//   prefactorization  LLT prefactorization of energy's quadratic matrix
//   Uprev  #V by 3 list of previous vertex positions (at time t-Δt)
//   Ucur  #V by 3 list of current vertex positions (at time t)
// Outputs:
//   Unext #V by 3 list of next vertex positions (at time t+Δt)
```

fast_mass_springs_step_dense

- Note the differences between:
  - Uprev = positions at t-dt
  - Ucur = positions at t
  - V = positions at rest

$$\mathbf{y} := \frac{1}{\Delta t^2}\mathbf{M}(2\mathbf{p}^t - \mathbf{p}^{t-\Delta t}) + \mathbf{f}^{\text{ext}} + \textit{pinned vertex penalty term}$$

$$\mathbf{b} := k\mathbf{A}^\top\mathbf{d} + \mathbf{y} \in \mathbf{R}^{n\times 3}$$

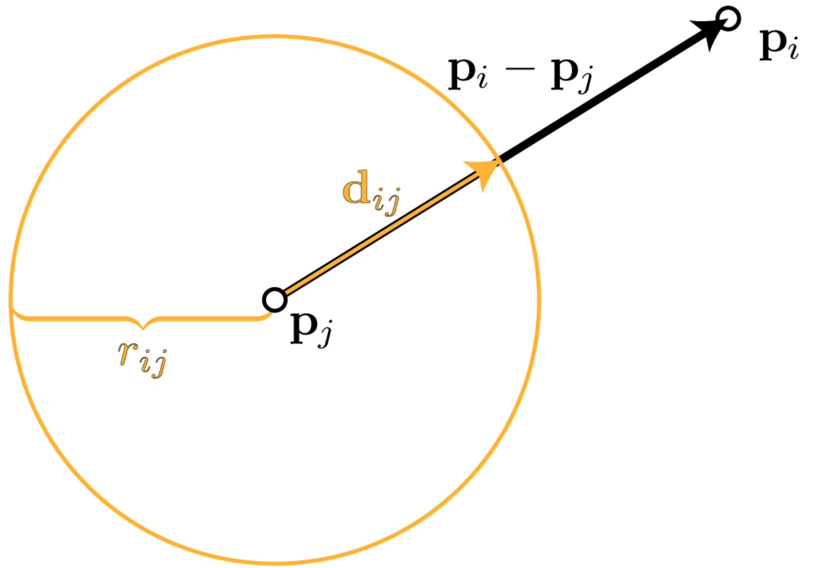
- Careful: do not create a new variable named b (**Q**p = b)
  - Already used for the list of pinned vertices

- Notice that y is constant for each iteration, only b changes (since d changes)
  - i.e. construct y once before the iterations

fast_mass_springs_step_dense

- Remember that we need 50 iterations of a local-global solve

- For each iteration:
  - find d (directions), that attempts to preserve current edge length $\mathbf{v} = \mathbf{A}\mathbf{p} \leftrightarrow \mathbf{v}_{ij} = \mathbf{p}_i - \mathbf{p}_j.$
    - i.e. normalize d then multiply by edge lengths e
  - then find p using $\mathbf{p} = \mathbf{Q}^{-1}\mathbf{b}$



$$\frac{\partial \operatorname{tr}\left(\mathbf{x}^\top \mathbf{y}\right)}{\partial \mathbf{x}} = \mathbf{y}$$

Pinned vertices

$$\frac{\partial \frac{1}{2}\operatorname{tr}\left(\mathbf{x}^\top \mathbf{Y}\mathbf{x}\right)}{\partial \mathbf{x}} = \mathbf{Y}\mathbf{x}$$

- Need to differentiate the energy:

$$\frac{w}{2}\operatorname{tr}\left((\mathbf{C}\mathbf{p}-\mathbf{C}\mathbf{p}^{\text{rest}})^\top(\mathbf{C}\mathbf{p}-\mathbf{C}\mathbf{p}^{\text{rest}})\right) = \frac{1}{2}\operatorname{tr}\left(\mathbf{p}^\top(w\mathbf{C}^\top\mathbf{C})\mathbf{p}\right) - \operatorname{tr}\left(\mathbf{p}^\top w\mathbf{C}^\top\mathbf{C}\mathbf{p}^{\text{rest}}\right) + \text{constant}$$

- Construct C from variable name b (list of pinned vertices)
- Once you have the derivative:
  - The quadratic term gets added to Q (first RHS term; in precomputation)
  - The linear term gets added to b (second RHS term; in step)
- Follow the derivation of Q and b
- Remember input V is the list of **rest** vertex positions

Sparse Versions

- Copy-paste dense code and change all the dense matrix datatypes
- All large dense matrices should be converted to sparse
- Vectors stay dense
- Use Eigen’s *setFromTriplets*
  - Use *std::vector* for constructing the list of triplets
  - *ijv.emplace_back(i,j,v);* means adding entry with row# i, col# j, and value v
- DO NOT add zero entries!
- DO NOT create a dense matrix then convert it to sparse!