Physics-Based Animation

Jan 30, 2014

Administrative Items

Scheduling

Next week: Discussion of 2 meshing papers on Tues, NO CLASS THURS.

Feb 11 & 13: Collisions and constraints – Will pick these ASAP.

Feb 18 & 20: Reading week

Presentation scheduling:

Feb 25 & 27: Rods and Hair

March 4 & 6: Fracture and Cutting

Presentations

Grades posted in LEARN.

Changes for next round:

1. Peer feedback.
2. Clear grading rubric.
3. Send me your slides afterwards, and I will provide more detailed feedback.

Proposals

Describe your project, summarize the methods you will use.

Break the project into 3-5 concrete steps/milestones.

1-2 pages, including references.

I can try to suggest references if you tell me your topic.

E.g. Shells simulator:

- Implement in-plane elastic/stretching forces – explicit integration.
- Implement bending force – explicit integration (“Discrete Shells”).
- Add collisions with simple objects (spheres, planes).
- Convert to implicit integration for stability/speed.
- Add a physical damping model.

Proposals

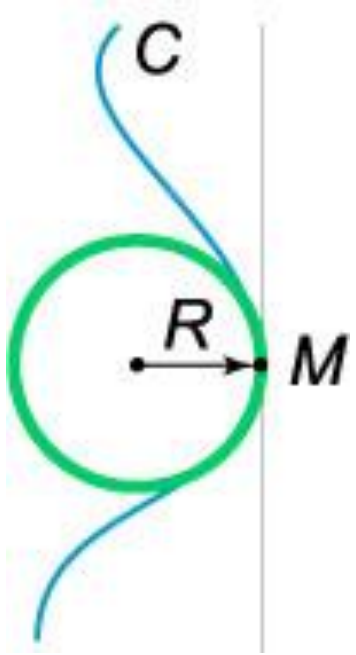
You can pursue novel extensions after demonstrating basic techniques.

If you're quite ambitious, the *Symposium on Computer Animation* deadline coincides with the project deadline (April 15).

A Bit of Curvature

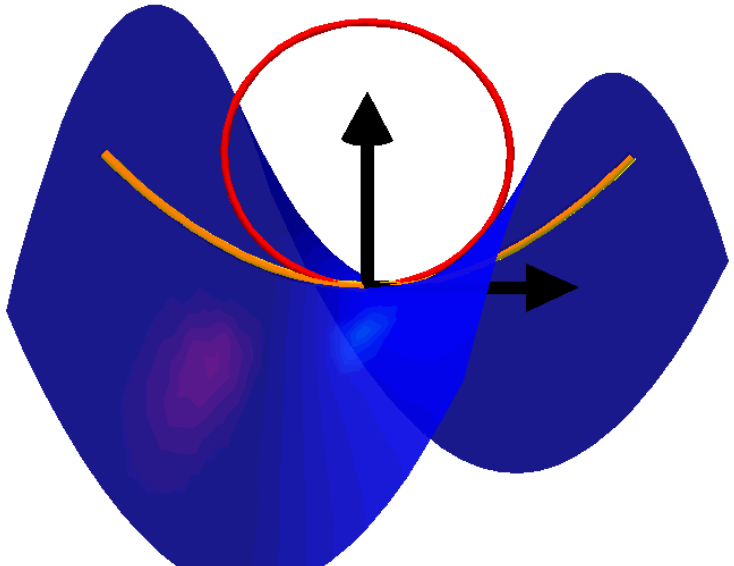
Curvature of Surfaces

- Curvature is $\kappa = \frac{1}{R}$ for a circle fit to a curve (per Richard's talk).



- In 3D, pick a plane, slice the surface with it, and fit a circle to the curve in that plane. Many possible curvatures!

- There will always be a max and a min curvature: the *principle* curvatures: $\kappa_1,\ \kappa_2$.

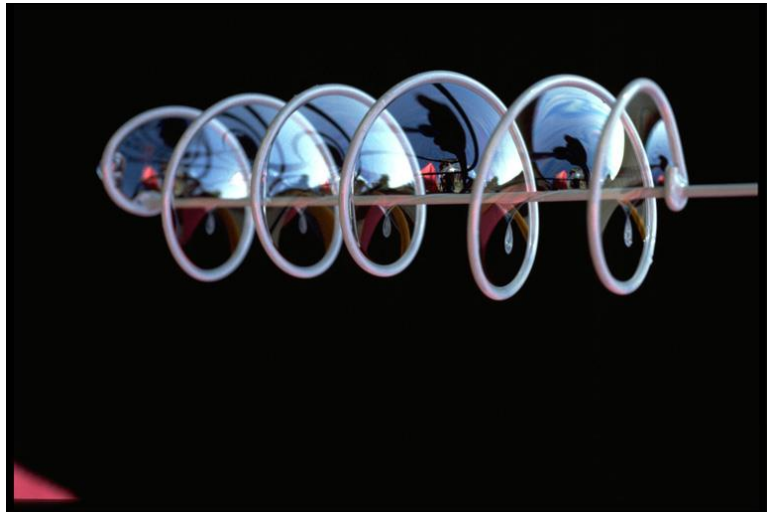


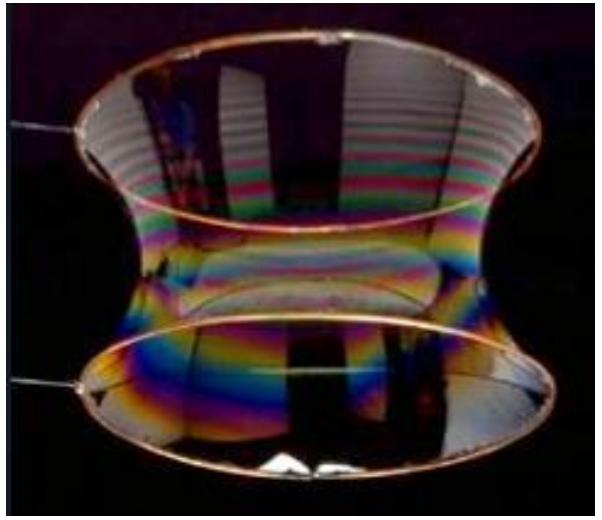
Mean Curvature

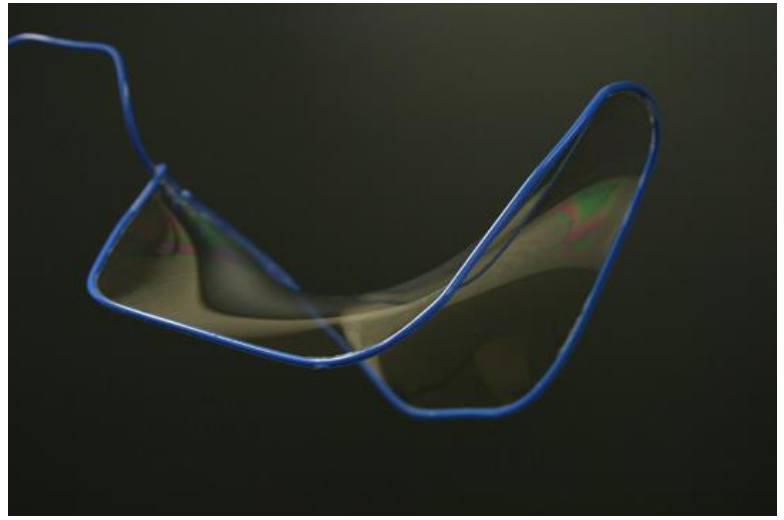
…is what it sounds like: $H = \frac{1}{2}(\kappa_1 + \kappa_2)$

Surfaces with zero *mean curvature* are called *minimal surfaces*.

Loosely speaking, they minimize area and behave “like soap films”.





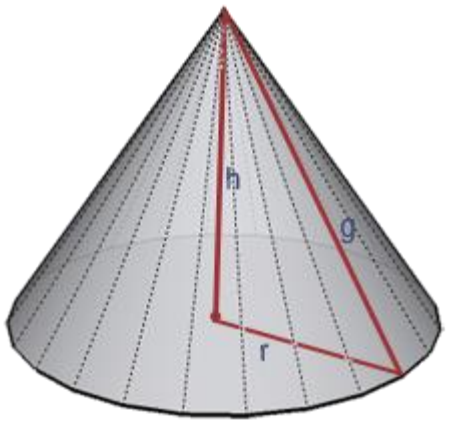


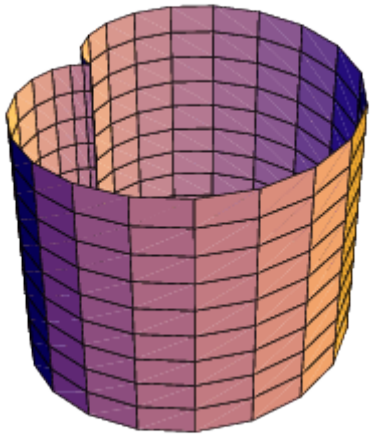
Gaussian Curvature

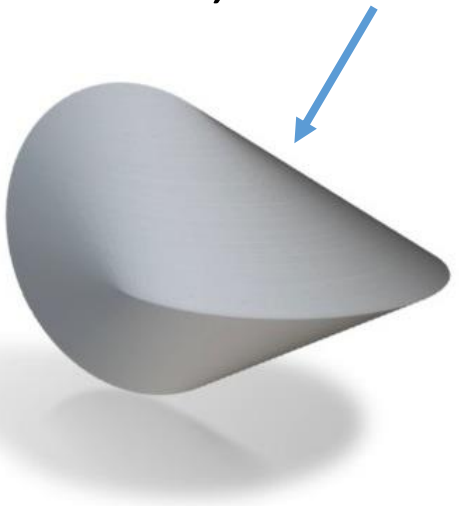
The product of principal curvatures: $\mathrm{K} = \kappa_1 \kappa_2$

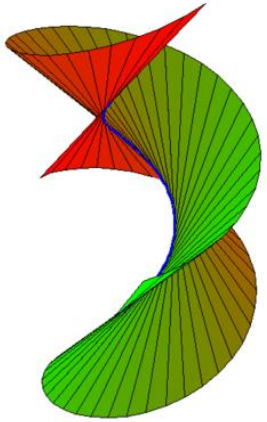
Surfaces with zero *Gaussian curvature* are called *developable surfaces.*

Planes, conical surfaces, (generalized) cylinders, “oloids”, …







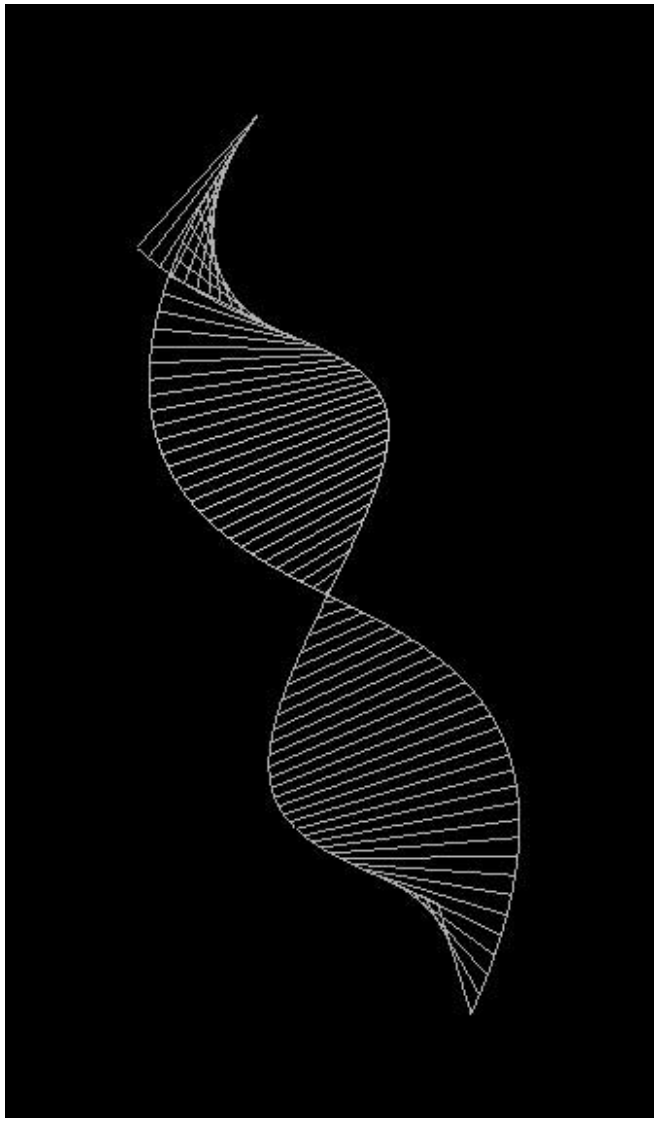


Ruled Surface

There is always a *straight* line through any point on the surface.

Note: Developable surfaces are ruled, but ruled does not imply developable.

e.g, Hyperboloid is not developable.



Isometric deformations

Deformations which do not stretch the surface.

i.e., distance between points (measured *within* the surface) do not change.

For shells, this corresponds to (hypothetically) infinite stretching/membrane stiffness.

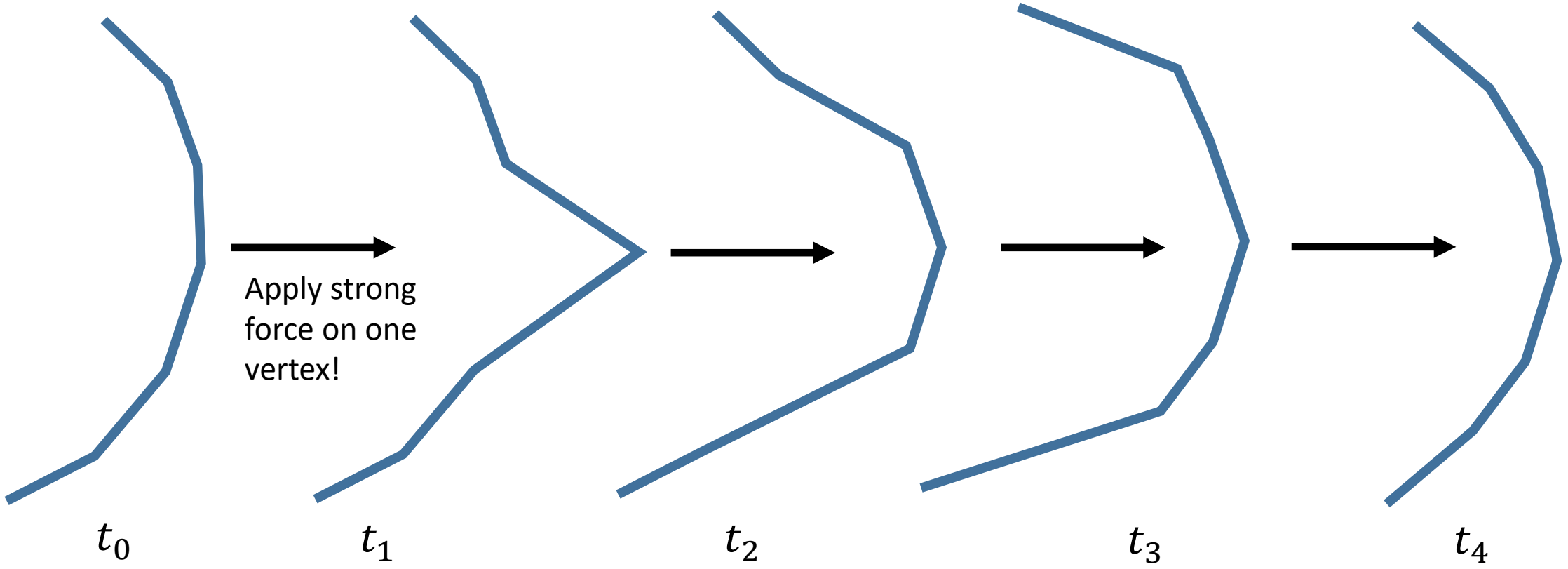
More on differential geometry for graphics, see Mark Pauly's slides:
http://www.pmp-book.org/download/slides/Differential_Geometry.pdf

A Bit of Time Integration

Propagation of deformations

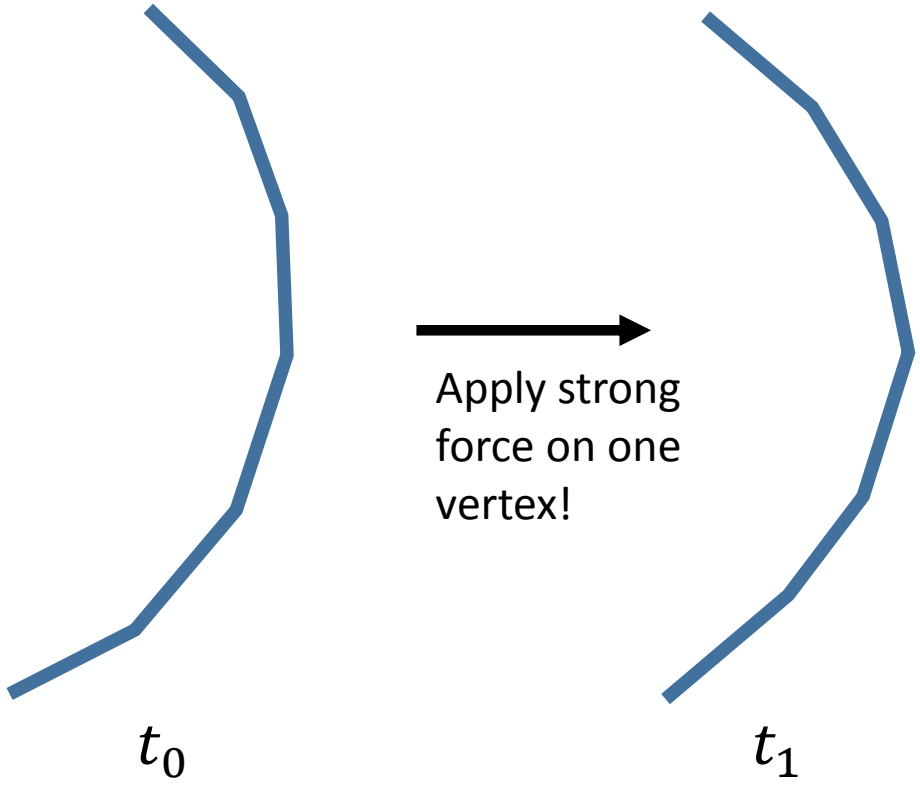
Explicit integration: Uses only *local information* from the current step.

∴Deformations propagate slowly through the mesh, one step at a time.



Propagation of deformations

Implicit integration: Solves a set of (globally coupled) equations for the end-of-step positions; allows deformations to propagate “instantly”.



(But recall, each step is far more expensive.)

A Bit about Autodiff

Automatic differentiation is not…

Symbolic differentiation:

Given a function, symbolic differentiation finds the *function* that is its exact analytical derivative.

(e.g., Maple, Mathematica, etc.)

e.g., Given expression for $f(x)$, you get back an expression for $f'(x)$.

Automatic differentiation is not…

Numerical differentiation:

Given a function and its arguments, numerical differentiation evaluates the function at multiple points to *approximate* the derivative there.

(e.g., finite differences.)

e.g., Given $f(x)$ and $x = 2$, you get back an *approximation* of $f'(2)$.

Automatic differentiation

Given a function, and its arguments, *evaluates* the derivative, up to numerical precision.

As each operation of f is applied, autodiff evaluates the corresponding derivatives.

e.g. Given $f(x)$ and $x = 2$, you get back $f'(2)$ (without F.D. errors).

Automatic differentiation

For example, if f performs a multiplication, $x \cdot x$ for $x = 2$, autodiff evaluates…

$$\frac{d}{dx}(x \cdot x) = \left(\frac{d}{dx}x\right) \cdot x + x \cdot \left(\frac{d}{dx}x\right) = x + x = 2 + 2 = 4$$

This allows the final result to be accumulated, op-by-op. (“Forward accumulation”.)

In C++, you might have a “number” class, that overloads the various operators, such that behind the scenes it evaluates the necessary derivatives/chain rules.

Paper Discussion