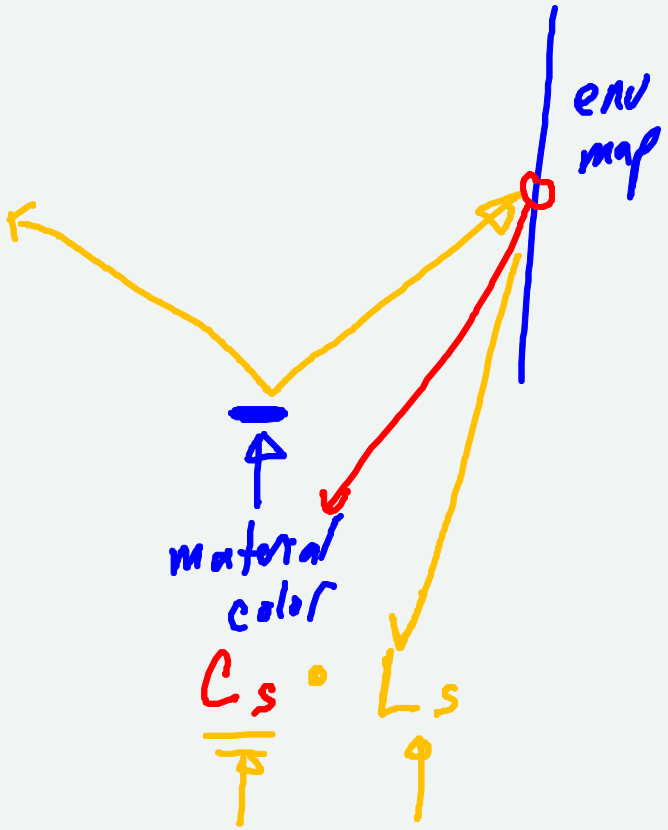
Lecture 25
Shape Deformation

Last Time: Graphics Performance

- Early Z
- Deferred Shading
- Using Environment Maps for Complex Lighting
- Texture Use and Re-Use (Atlases)
- Avoiding State Changes (big objects)
  - Matrix Palettes
  - Skinning



Terminology: Multi-Pass Rendering

We draw the set of objects multiple times

1. Draw a pass to make a map
    - Dynamic Environment Maps
    - Shadow Maps
2. Draw a pass to prepare the frame buffer
    - Early Z (draw without shading)
3. Draw passes that "add colors"

OK, But What Can I Do

In Graphics Town you can:

1. Be careful about texture usage (use an Atlas!)
2. Use Environment Maps
3. Try Skinning and Morphing (built into THREE)
4. Try implementing complex deformations (good shaders practice)

Animation by Transformation

Translate or rotate... scale

1. Change each vertex
    - compute N vertices
    - transmit N vertices between CPU and GPU
2. Change a transformation
    - change 1 number (maybe 12 for a matrix)
    - send 1 matrix to GPU

Downside: limited things we can do (with simple transformations)

A better motivation for skinning...

More Generally...

Make one shape

Deform it to other shapes

- easier to animate
- easier to model/control

Animation by Deformation

Advantages:

1. performance (don't need to compute every vertex)
    - no need to send mesh to graphics hardware each frame
    - per-vertex computation with limited data
2. authoring (artists don't have to sculpt every vertex)
    - design base shape and make coarse adjustments
3. storage (don't need to remember every vertex in every pose)
4. re-use (apply deformations to different base shapes)

Deformation-based shape control

1. Shape Interpolation (Morphing)
2. Non-Linear (complex) deformations
3. Grid Deformers (Free-Form Deformations)
4. Cages

5. Skeletons ← Skinning

Idea 1: Shape Interpolation (Morphing)

Create multiple copies of the mesh

- each copy is a **morph target**

Vertices interpolate between targets

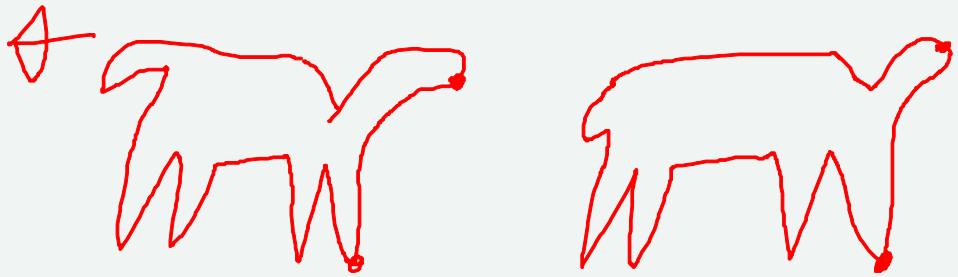
- blend their positions in each target
- $p = w_1 p_1 + w_2 p_2 + w_3 p_3$ for each vertex

Send all meshes to the hardware

Each frame only changes the weights

Downsides

1. Need to make all the meshes
2. Meshes need to correspond
3. In-between values may not be meaningful
4. Control by blending (not always easy)



In THREE

Build in to THREE!

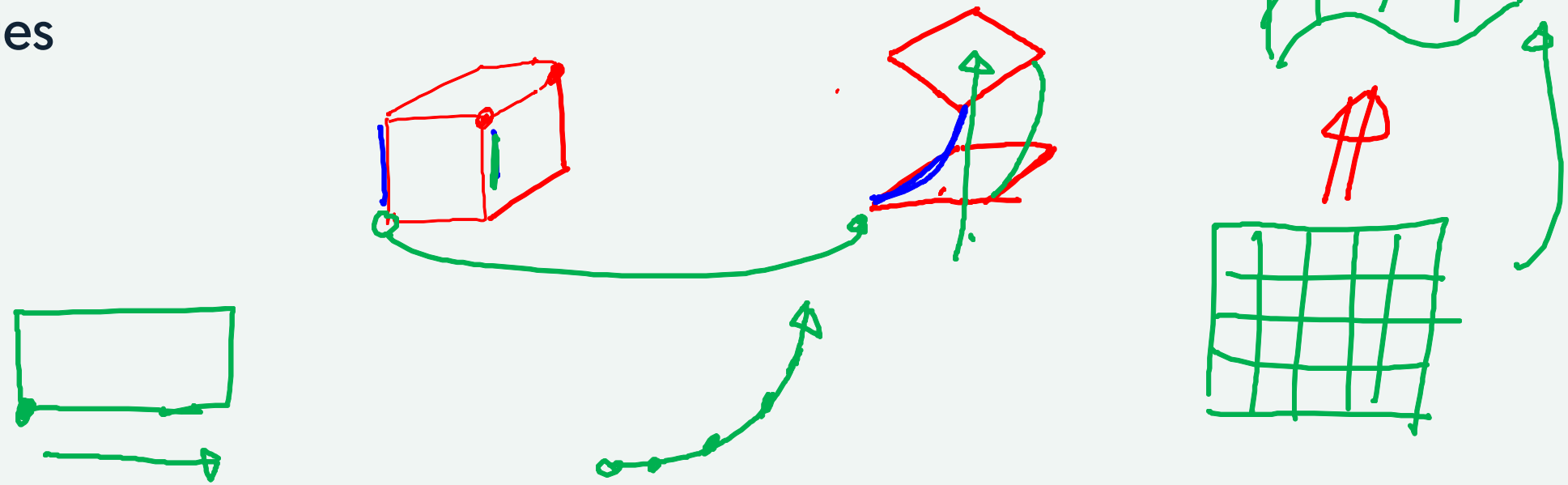
(see that weird blobby thing in the graphics town demo)

#2

Non-Linear Deformations

- Bend/Twist/Other
- Lattice / Free-Form Deformation
- Cages

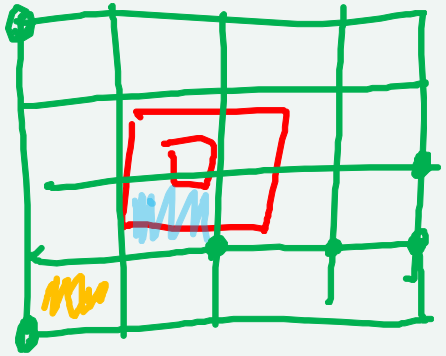
$f(x) = Fx$

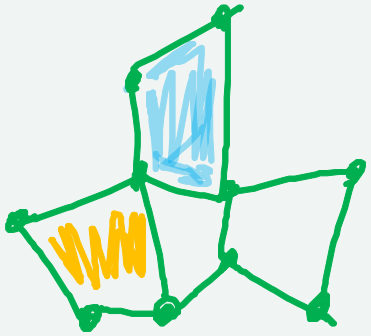


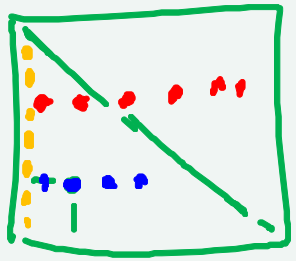
Deformations as space transformations

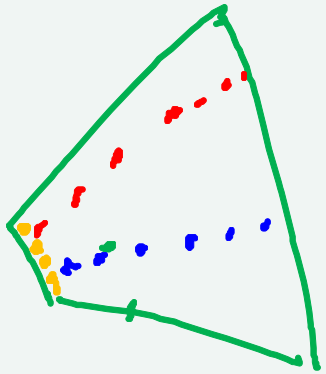
$$x', y', z' = f(x, y, z)$$

Grid Deformers





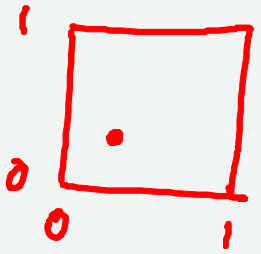


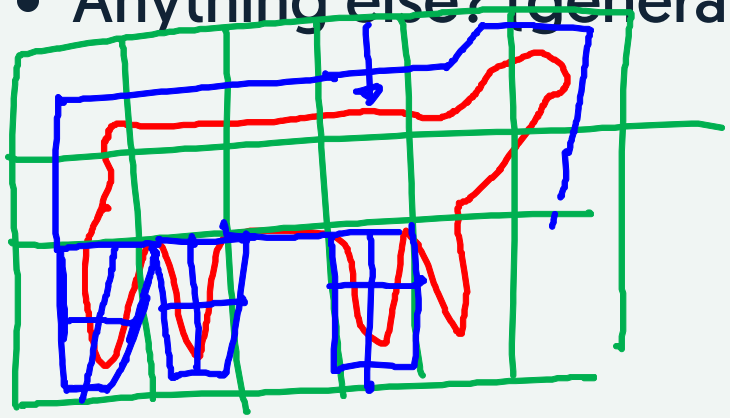


Free-Form Deformations (FFD)

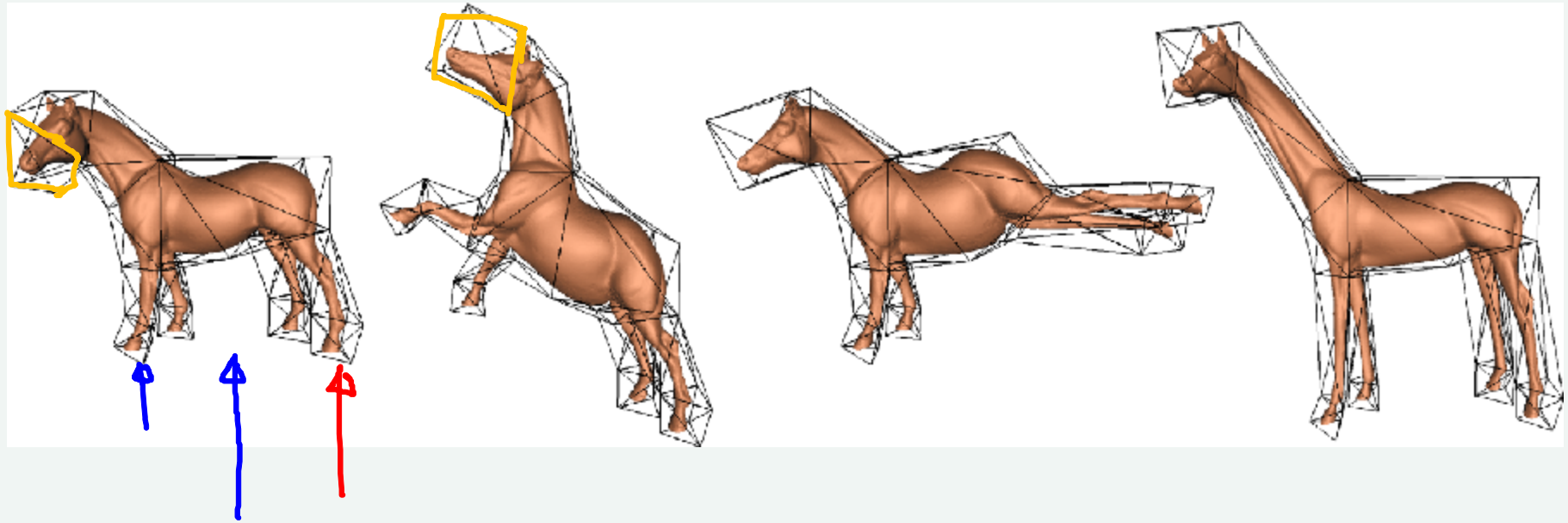
The trick: coordinates in irregular shapes

- Triangles (Barycentric)
- Squares (XY)
- Anything else? (generalized barycentric)

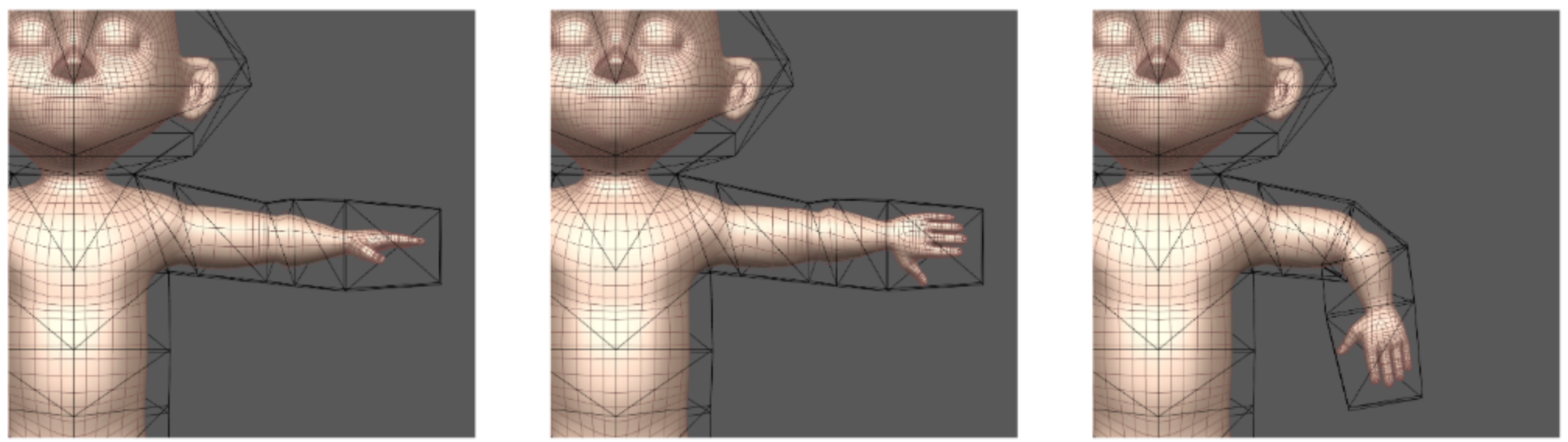




Cages (Coordinate-based Deformations)



Harmonic Coordinates (Pixar)

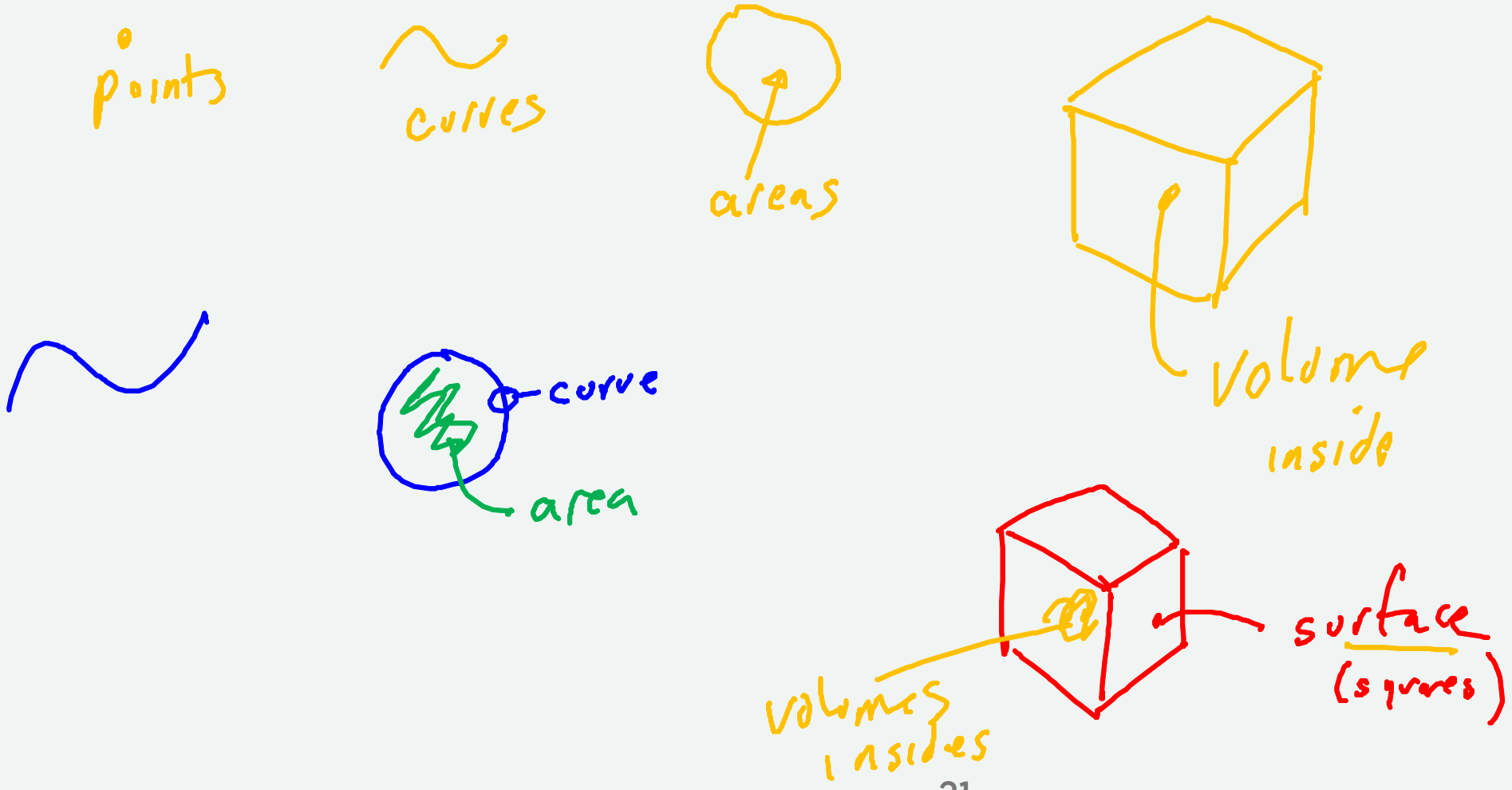


https://graphics.pixar.com/library/HarmonicCoordinatesB/paper.pdf

More generally...

How do we make smooth shapes?

Curves vs. Surfaces vs. Solids



Curves in 3D

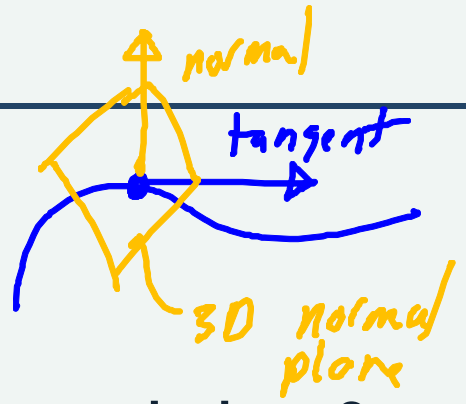
Everything we learned in 2D

just another dimension

dimensions are independent in polynomial curves

Curves in 3D

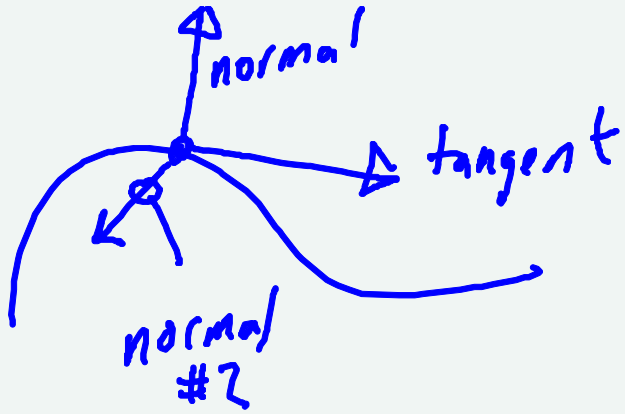
- Tangent Vector
- Normal Plane



How do we orient an object in the normal plane?

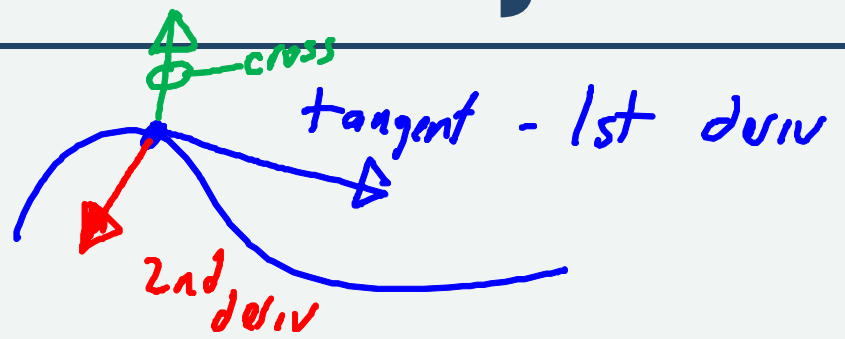
We need a **frame** - a coordinate system that moves along curve

It needs to be consistent

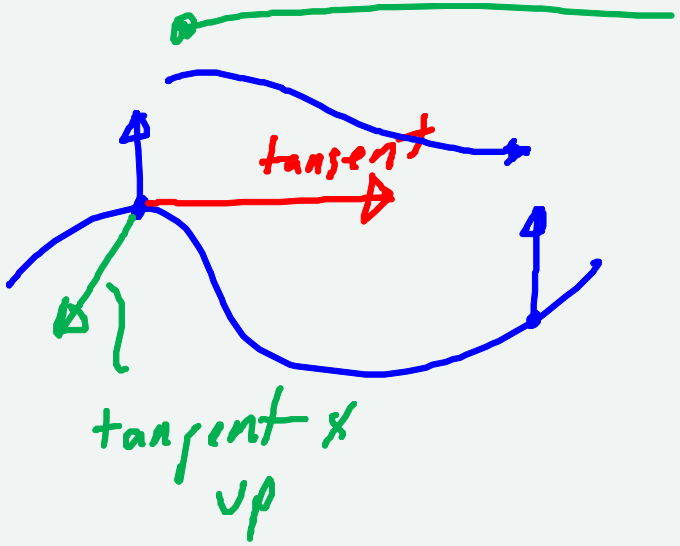


Roller Coasters (Trains in 3D)

The ghost of 559 past...



1. Frenet Frame
    - Tangent
    - Normal Vector (direction of 2nd derivative)
    - Bi-Normal (cross product of 1st two)
    - **what if one vanishes?** (straight segment)
2. Interpolate Up Vector



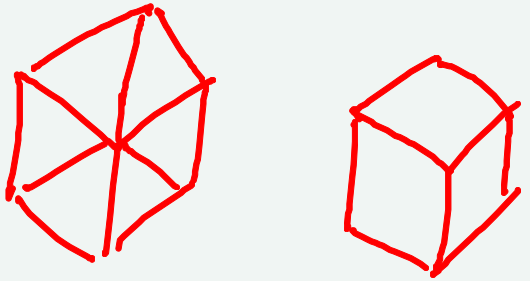
Solid Modeling

A (non-infinite) surface (area) is bounded by a curve

A curve may (or may not) bound a surface (area)

A (non-infinite) solid is bounded by a surface

A surface may (or may not) bound a solid



If you want a solid, be careful (that's a different class)

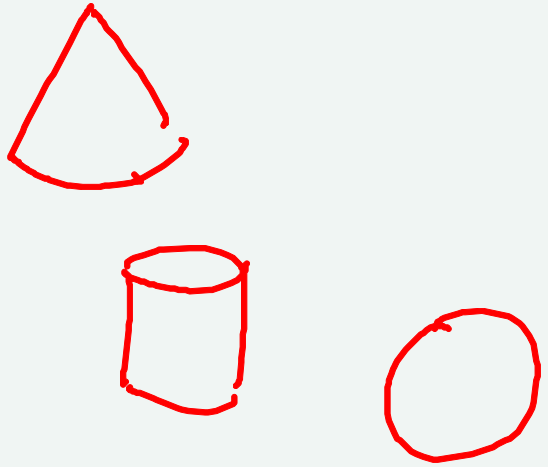
Surface Modeling

Flat surfaces (or piecewise flat)

- polygons
- triangles
- meshes

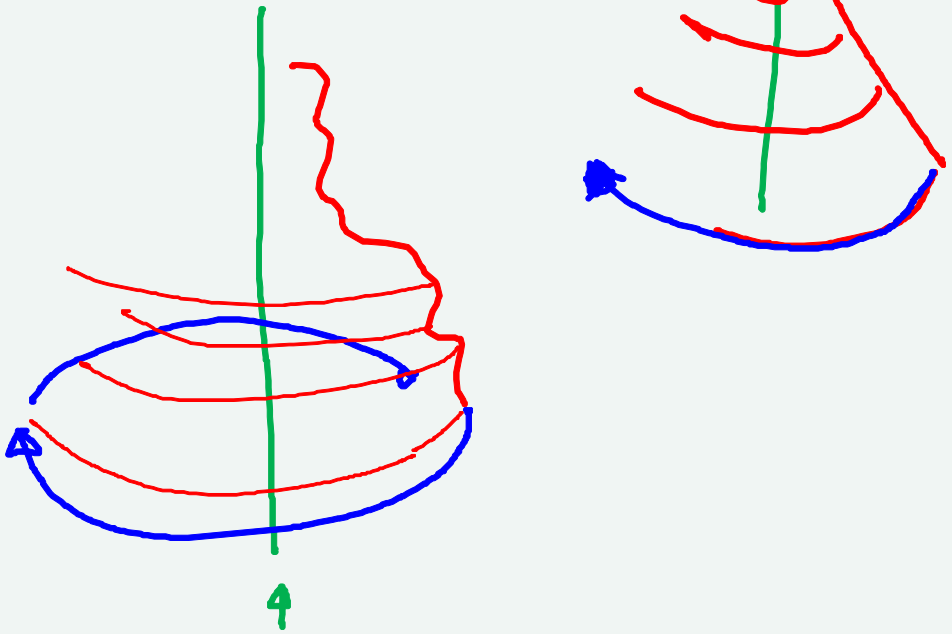
Standard shapes

- cone
- cylinder
- sphere (ball is volume)
- and many more...



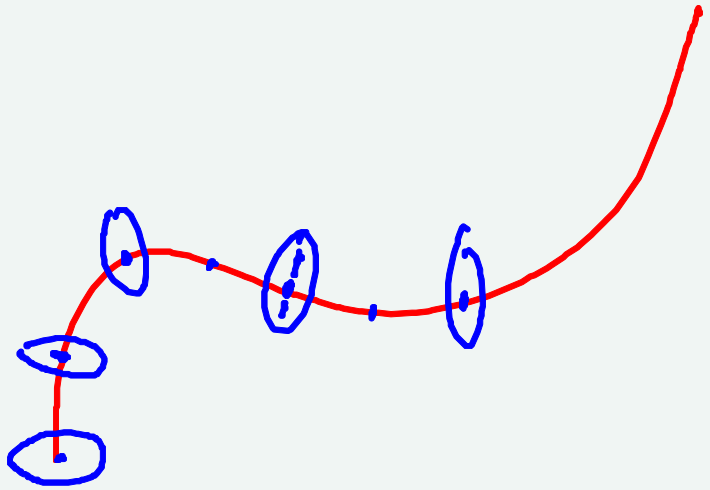
Surface of Revolution

1. Define a 2D Shape
2. Revolve it around an axis



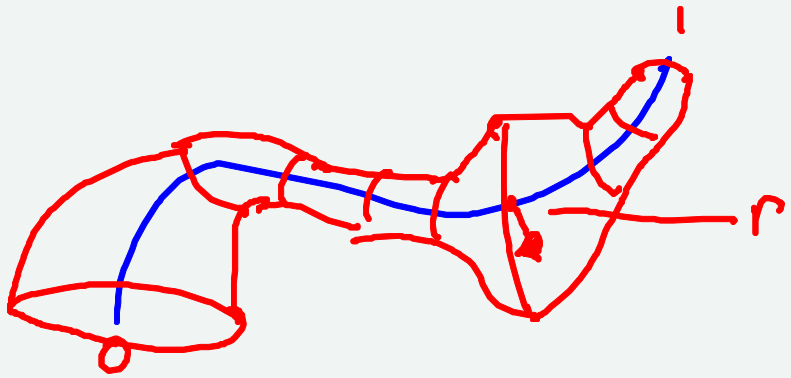
Generalized Cylinders (1) Tubes

1. Define a spine (function of t)
2. Give a radius



Generalized Cyliders (2) Cones

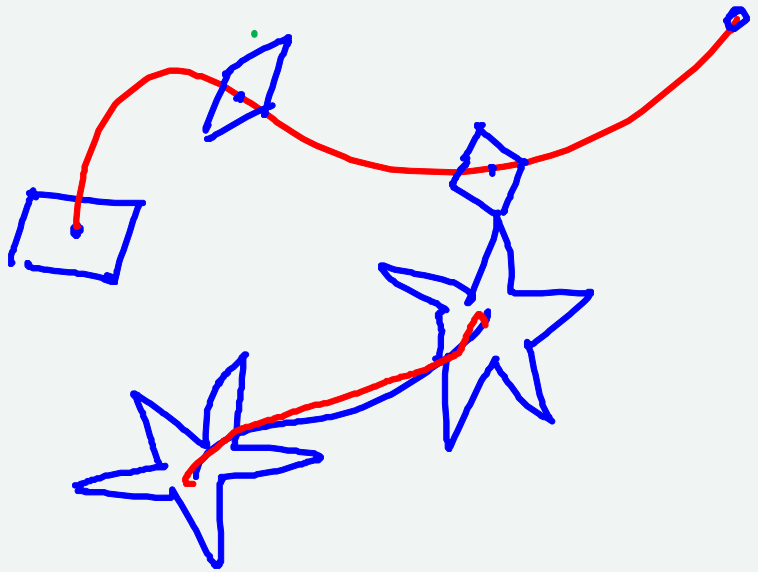
1. Define a spine (function of t) $t \to x, y, z \to$
2. Define a radius (function of t) $t \to r$



Generalized Cylinders (3) Sweeps

1. Define a spine
2. Define a cross-section shape

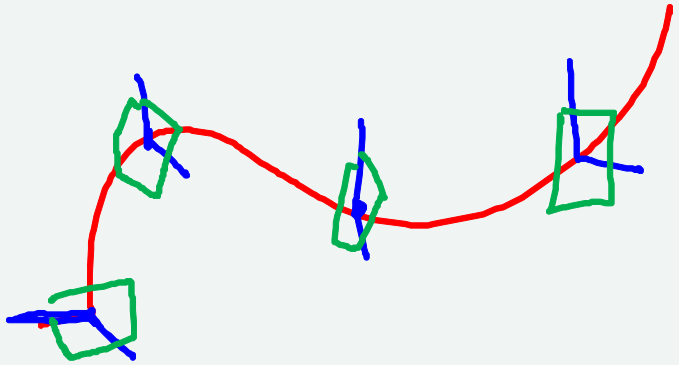
Tube = w/ cross section shape



Fancy Sweeps

2D Shape interpolation along spine

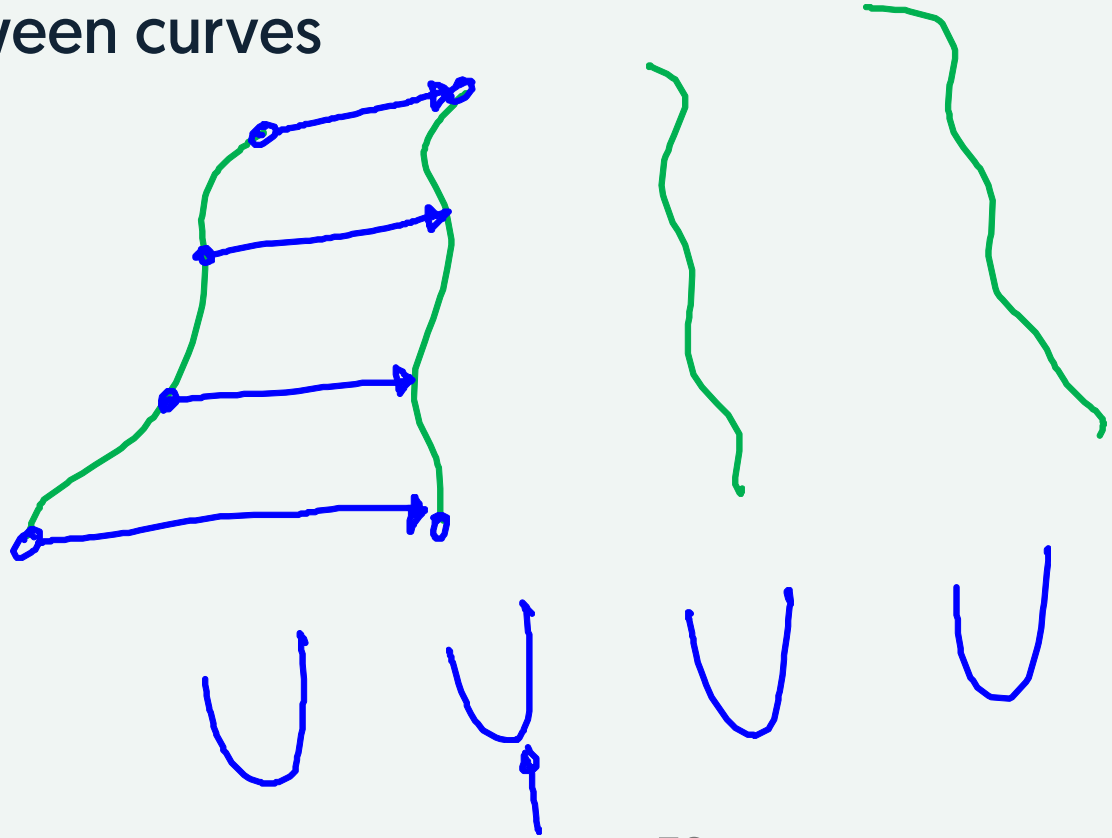
Requires good 3D curves



Lofting and Other Shape Methods

Define surfaces by curves

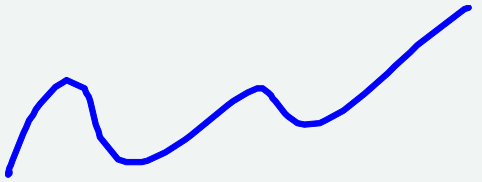
Interpolate between curves



Free Form Surfaces: Approaches

Same as curves

- Parametric: $(x, y, z) = \mathbf{f}(u, v)$
- Implicit: $f(x, y, z) = 0$
- Procedural
- Subdivision



common way for surfaces