Computer Graphics

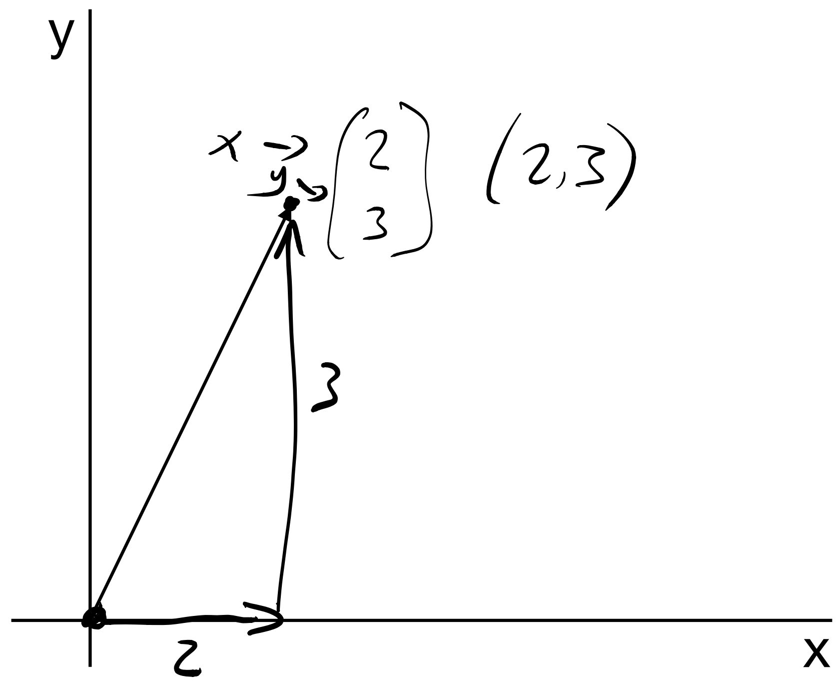
Lecture 2

Triangle Meshes - Geometry

Announcements

- Monday's lecture is pre-recorded (flipped).
- Class will be spent working on problems in groups on Discord.
  - We'll start in Zoom for announcements then go to Discord for remainder of class.
- HW0 and A0 are due Wednesday.

Vectors

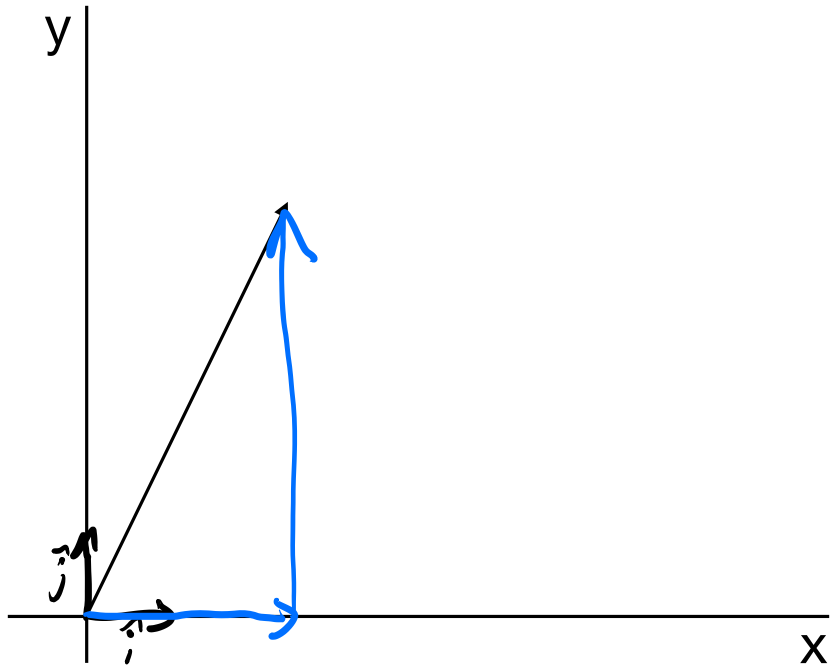


The Canonical Basis

$$\begin{bmatrix}2\\3\end{bmatrix} \to 2\cdot\hat{i} + 3\hat{j}$$

$$\hat{i} = \begin{bmatrix}1\\0\end{bmatrix}$$

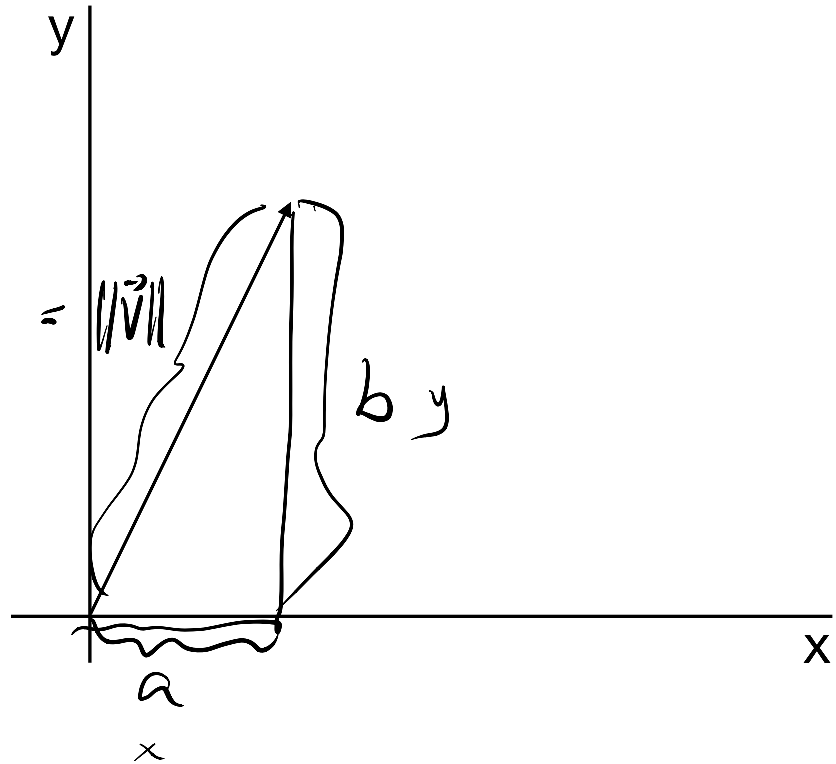
$$\hat{j} \begin{bmatrix}0\\1\end{bmatrix}$$



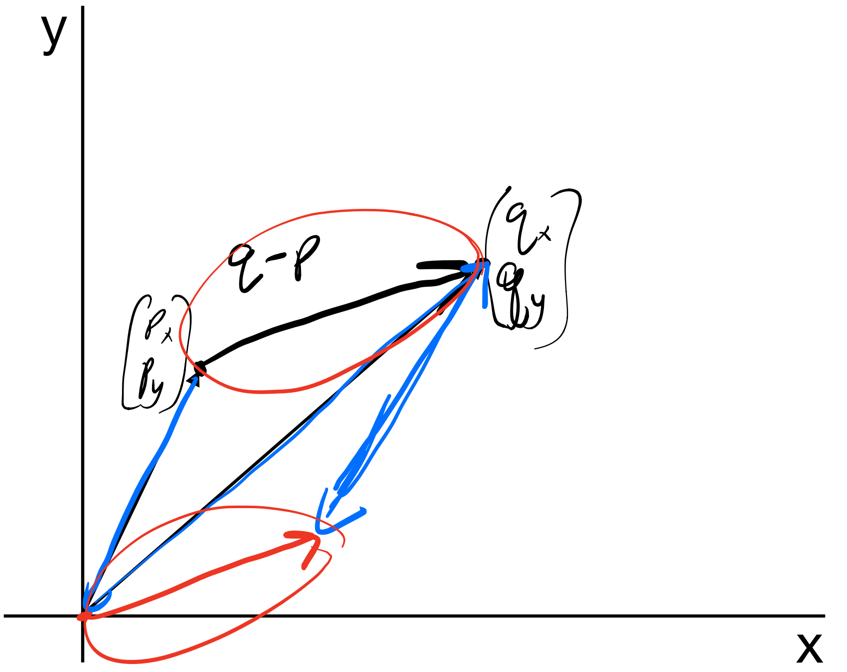
Magnitude (length)

$$\sqrt{a^2 + b^2} = \|\vec{v}\|$$

$$\sqrt{\sum(v_i^2)}$$



The vector between two points



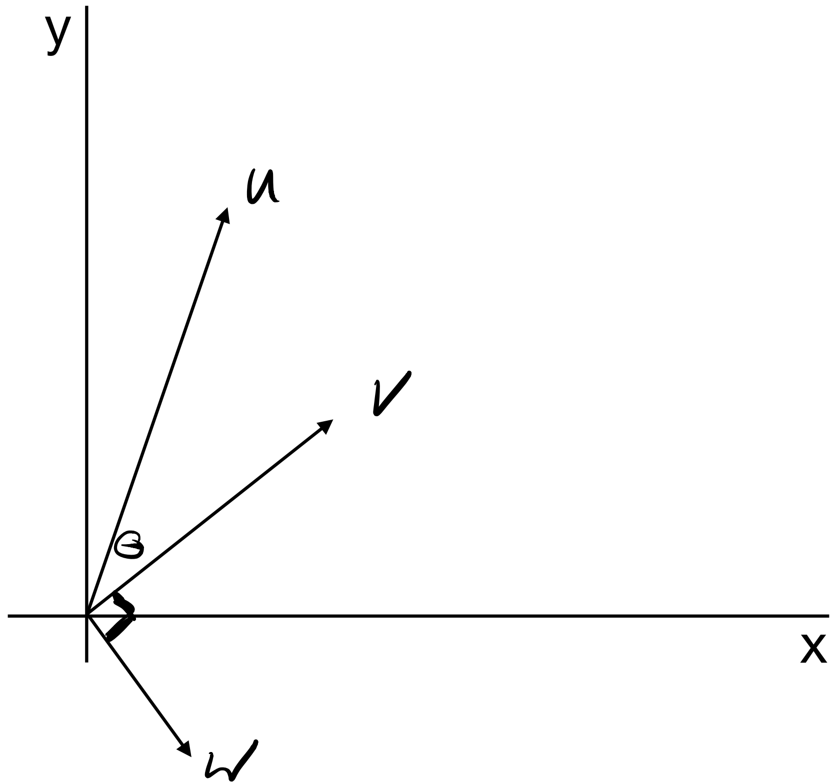
The dot product

$u \cdot v = \|u\| \cdot \|v\| \cos\theta$

$v \cdot w = \quad \cdot \cos 90$

0

$= 0$

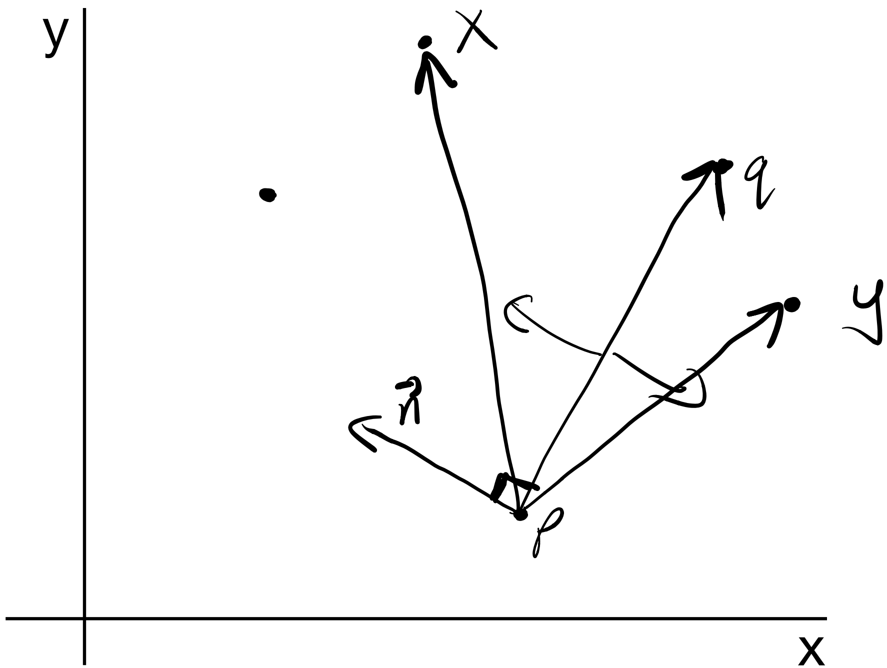


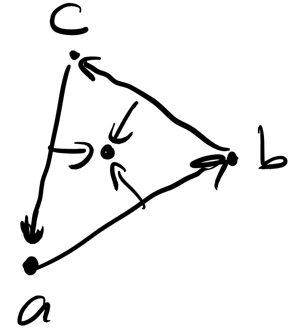
Point-in-Triangle

$n \cdot (x - p) > 0$

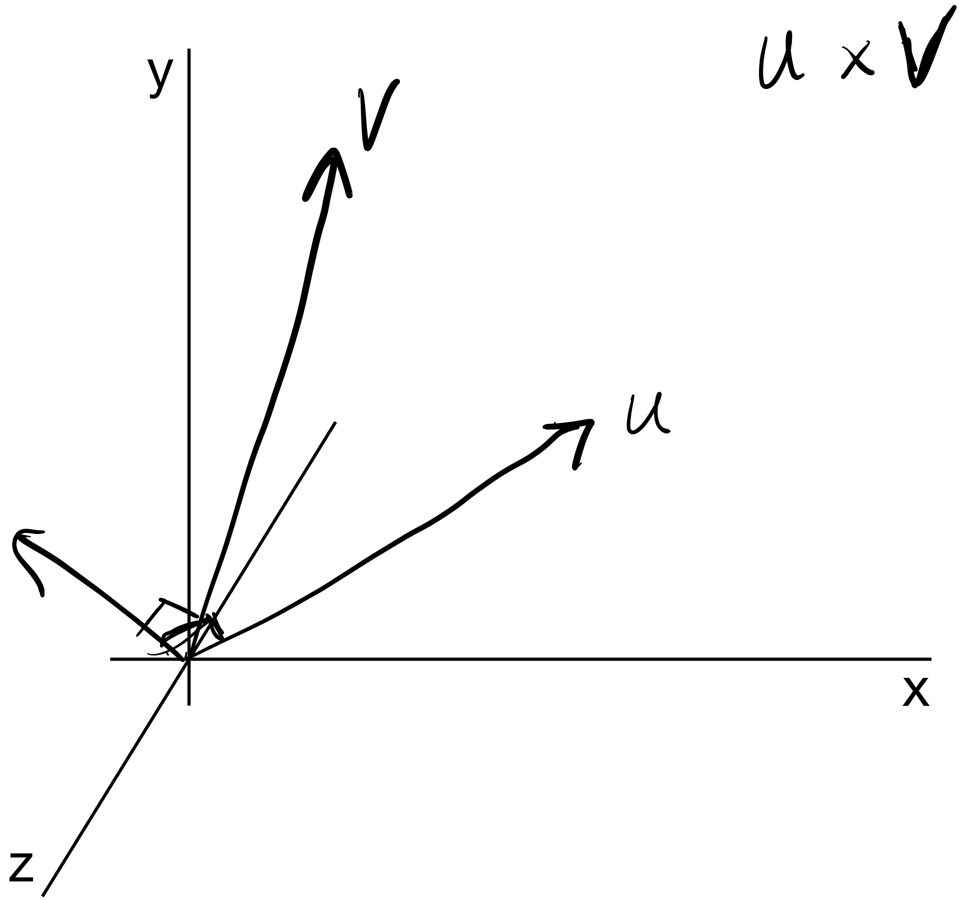
$n \cdot (y - p) < 0$

$= 0$





The cross product



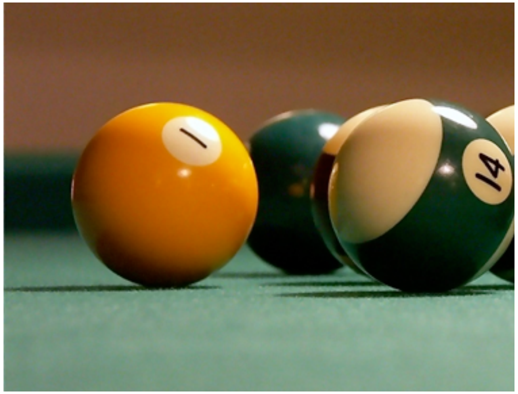
Modeling

Pseudocode for graphics:

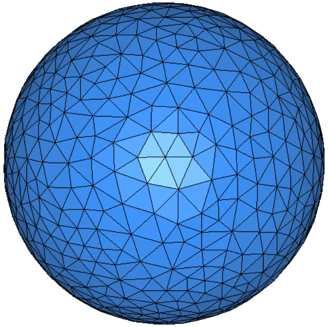
```
Create a model of a scene
Render an image of the model
```

Modeling a Sphere

- Center point and radius
- Triangle mesh



spheres



approximate sphere

which is better?

what does "better" mean?

This is a choice of data structures.

What's important to us? Let's brainstorm.

Space complexity / efficiency:

rendering speed / time complexity

accuracy

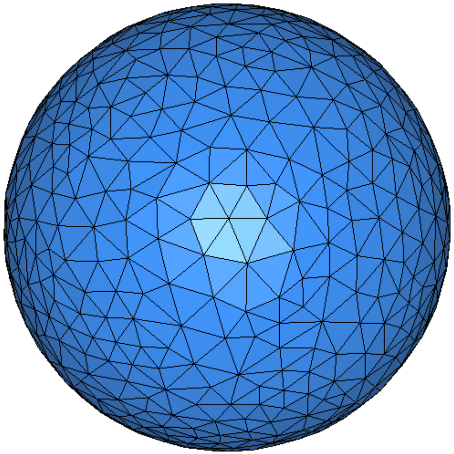
generality

Modeling

- This is really a choice of data structures.
  What's important to us?
  - What can the data structure represent?
    Here: **generality** and **manipulability** for modeling.
  - Space complexity: how memory-efficient is the representation?
  - Time complexity
    Here: efficient **operations needed for rendering**
    - Intersect rays with object (image-order)
    - Project all points on object down to 2D (object-order)

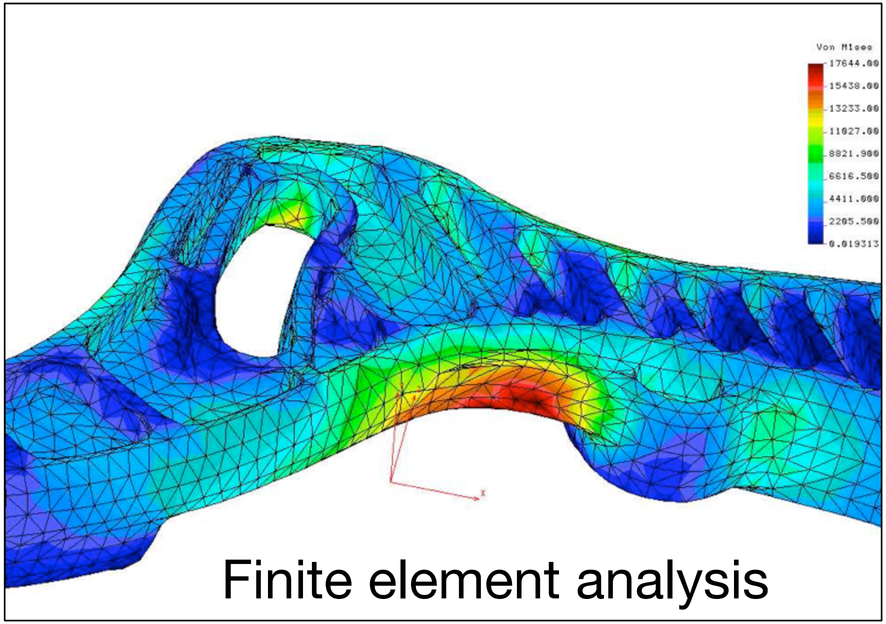
Meshes - Advantages

- Made of very simple *primitives* (usually triangles)



Meshes - Advantages

- Approximate arbitrary geometry
- Enables storage of surface properties



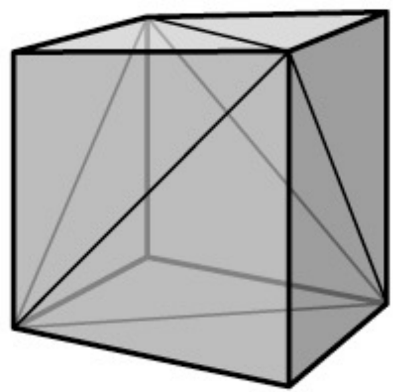
Finite element analysis

Meshes - Advantages

- Makes for cool architecture

Ottawa Convention Center

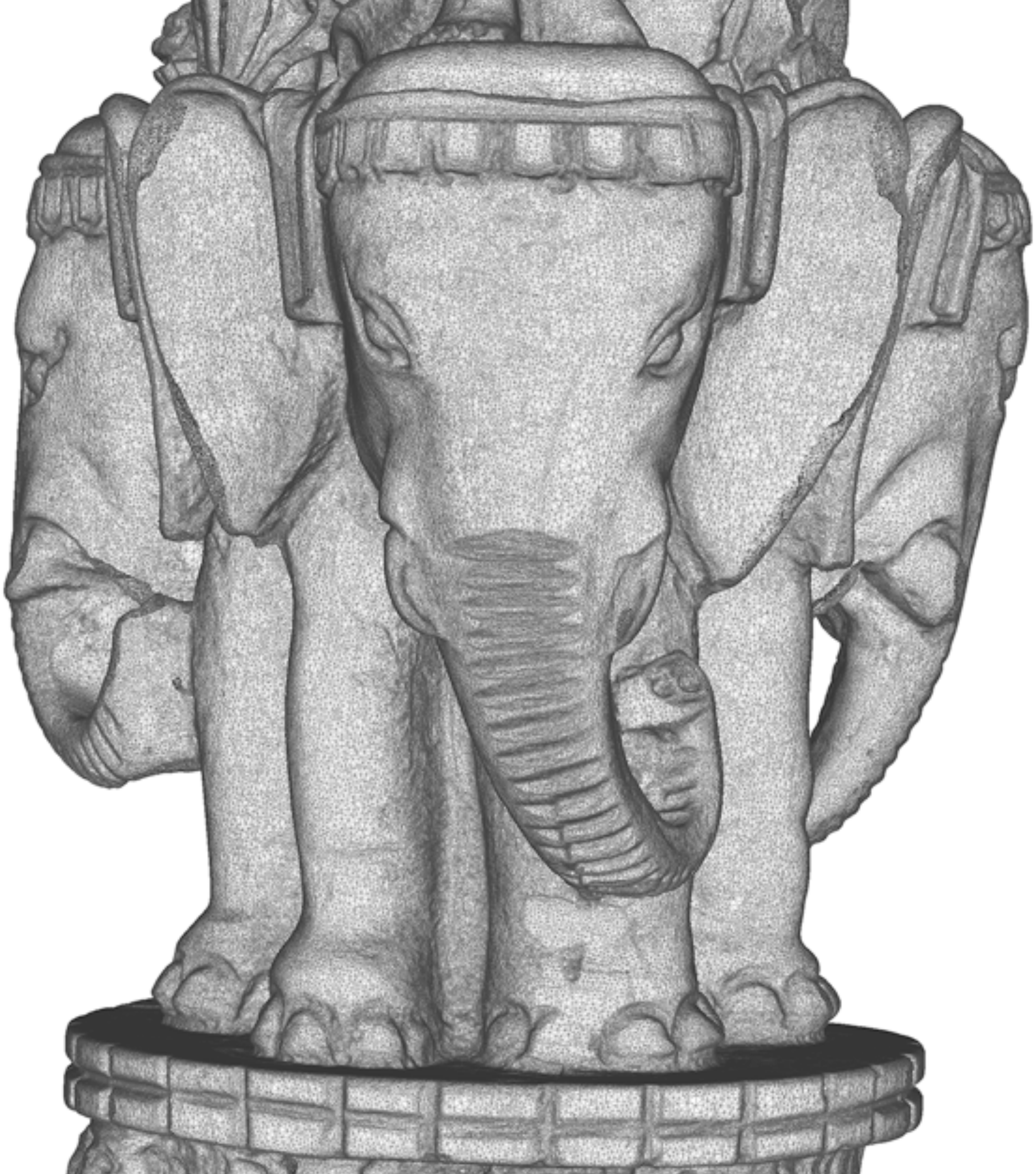
A small mesh

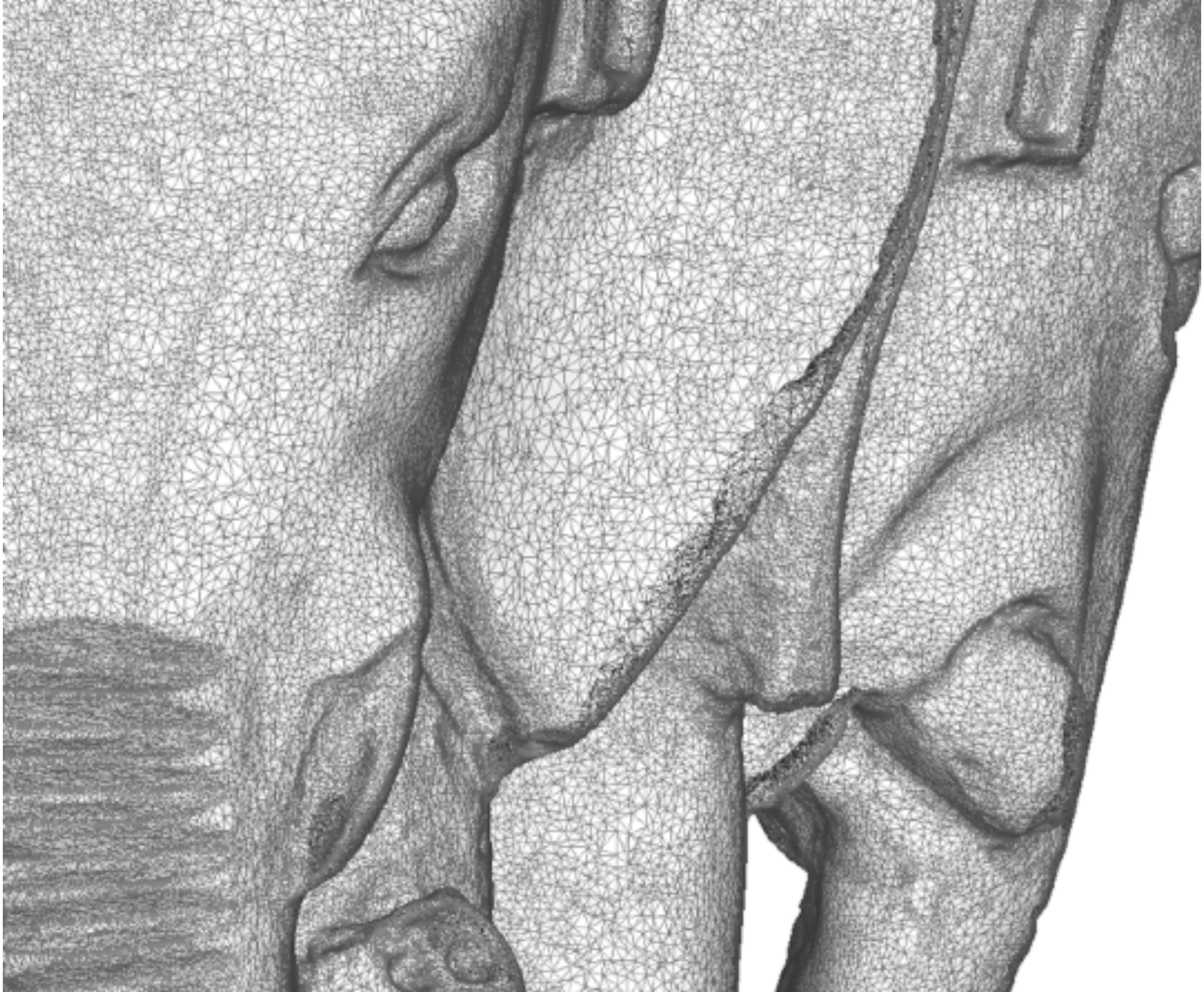


12 triangles, 8 vertices

A large mesh

Traditional Thai sculpture
scan by XYZRGB, inc.
Image by MeshLab project





A large mesh

- 10 million triangles
- Generated from a high-resolution 3D scan

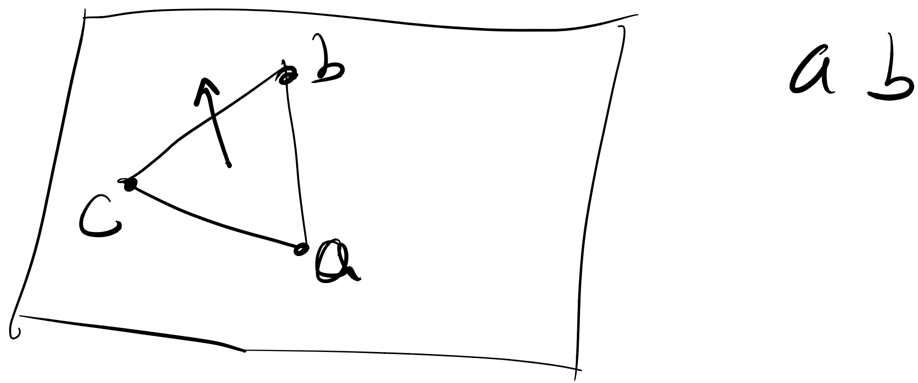
Let's talk about triangles

- Defined by three vertices
- Live in the plane containing those vertices
- Vector normal to plane is the triangle’s normal
- Conventions (for this class; not everyone agrees):
  - vertices are counter-clockwise as seen from the “outside” or “front”
  - surface normal points towards the outside (“outward facing normals”)

Aside: why not quadrilaterals? Other polygons?

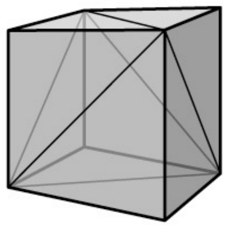
Let's talk about triangles

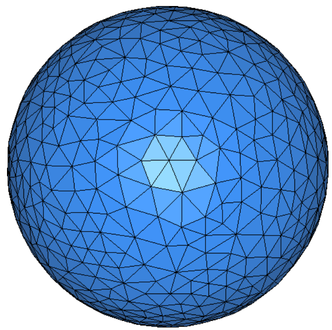
- A triangle is face up on a table - how do I represent it?



Triangle Meshes

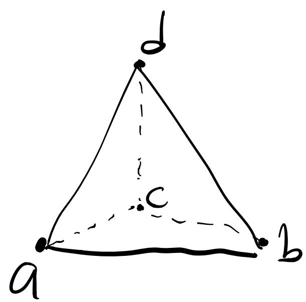
- A bunch of triangles in 3D space that are **connected together** to form a surface
- Geometrically, a mesh is a piecewise planar surface
  - almost everywhere, it is planar
  - exceptions are at the edges where triangles join
- Often, it's a piecewise planar **approximation of a smooth surface**
  - in this case the creases between triangles are artifacts—we don't want to see them

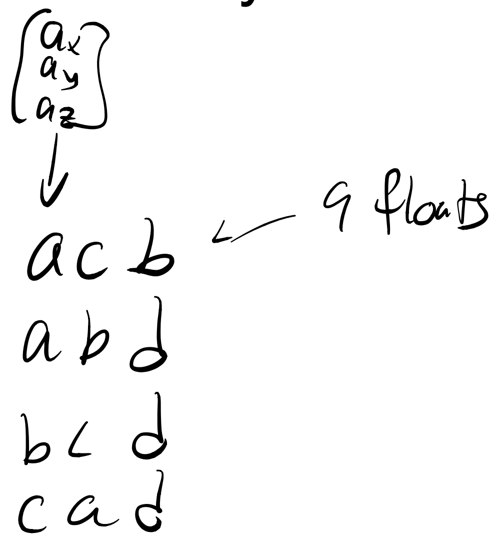




Representing Triangle Meshes

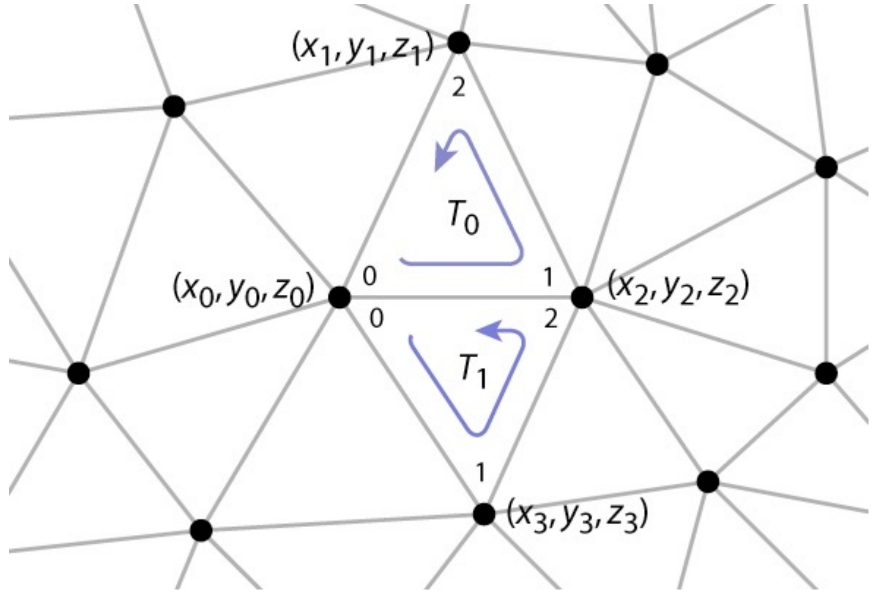
- How do we represent these in memory?
- Example: a tetrahedron





Separate Triangles

	[0]	[1]	[2]
tris[0]	x_0, y_0, z_0	x_2, y_2, z_2	x_1, y_1, z_1
tris[1]	x_0, y_0, z_0	x_3, y_3, z_3	x_2, y_2, z_2
	⋮	⋮	⋮

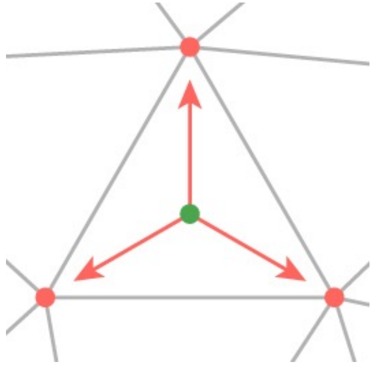


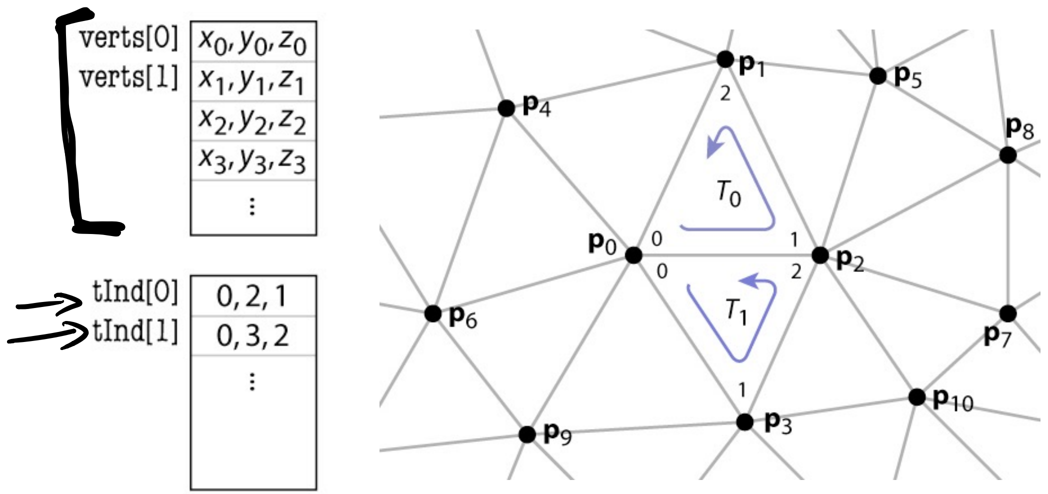
Problems:

- Wastes space
- Repeated floats with different round-off creates problems:
  - Cracks in the mesh
  - Finding neighbors may fail

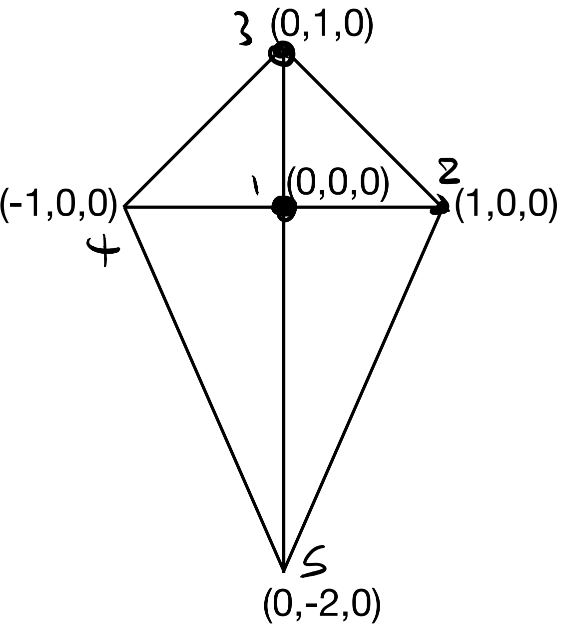
Indexed Triangle Set (A1)

- Vertices are listed once, without duplicates
- Each Triangle stores **indices** of its vertices





Problems: Kite Mesh



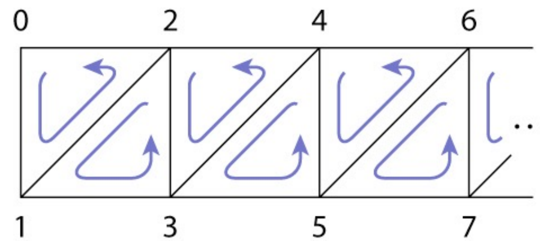
vertices:		triangles:
(1)	0, 0, 0	123
(2)	1, 0, 0	134
(3)	0 1 0	145
(4)	-1 0 0	152

1. Find your group number on Canvas
2. Get on the Discord server
3. Join your Group's voice channel
4. Open the Google Doc pinned in your group's text channel
5. Open today's problems
   - linked from #in-class-text
   - also P02 on course webpage schedule
6. Meet your group members
7. Solve Problems 1 and 2

(5) 0 -2 0)

Triangle Strips

- Takes advantage of mesh properties:
  - Each triangle is usually adjacent to previous
  - Next triangle reuses previous two vertices
  - Every subsequence of 3 vertices is a triangle



Vertex sequence

0, 1, 2, 3, 4, 5, 6, 7, …

leads to triangle sequence:

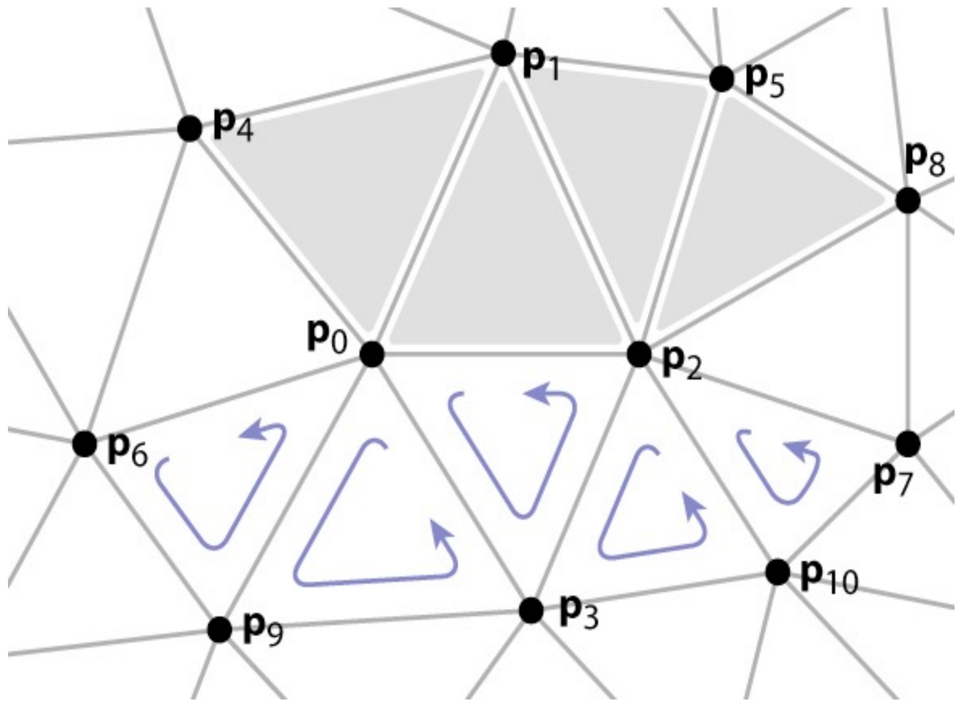
(0 1 2), (2 1 3), (2 3 4), (4 3 5), (4 5 6), (6 5 7), …

For long strips, about one index per triangle!

Triangle Strips

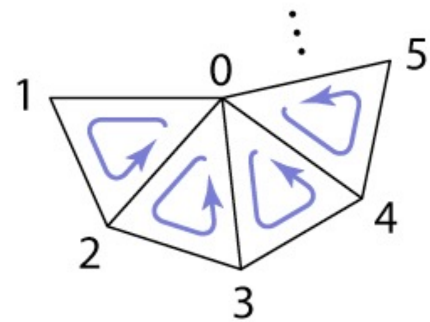
verts[0]	x_0, y_0, z_0
verts[1]	x_1, y_1, z_1
	x_2, y_2, z_2
	x_3, y_3, z_3
	⋮

tStrip[0]	6, 0, 4, 1, 2, 5, 8
tStrip[1]	6, 9, 0, 3, 2, 10, 7
	⋮



Triangle Fans

- Same idea as triangle strips, but keep oldest index rather than newest
  - Every sequence of three vertices is a triangle
  - Same benefits as triangle strips



What else?

- Indexed triangle sets are good for rendering, but not great for mesh **processing**.
- What if we want to efficiently find:
  - all triangles containing a vertex?
  - all triangles adjacent to a triangle?
  - the triangle across a particular edge of a triangle?
- You can augment the mesh data structure to store more. See Section 12.1.4.

Problems 3-4