

Computer Graphics

Lecture 3

Modeling
Triangle Meshes: Geometry



Announcements

- Tomorrow's and Friday's classes: watch video(s) ahead, work on Problems in class.
 - Tomorrow: 35 minutes of video
 - Friday: ~27+20 minutes of video (+8 optional but helpful minutes)
 - for Friday, a laptop is not required but one per group might be useful for the in-class problems (so can test your OBJ files)

Goals

- Understand the advantages and disadvantages of modeling objects using triangle meshes.
- Know how contiguous meshes of triangles can be represented using **separate triangle sets**, **indexed triangle sets**, **triangle strips**, and **triangle fans**.

Modeling

Pseudocode for graphics:

Create a model of a scene

Render an image of the model

Modeling

Pseudocode for graphics:

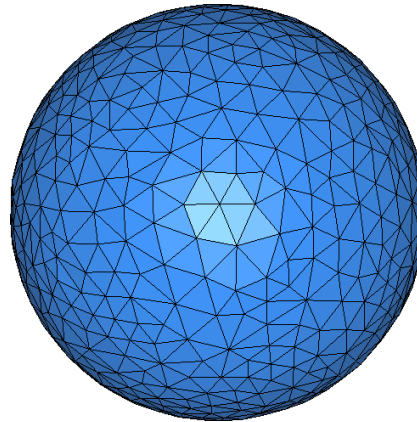
```
Create a model of a scene
```

```
Render an image of the model
```

Modeling a Sphere

Recall two possibilities:

- Center point and radius
- Triangle mesh



**approximate
sphere**

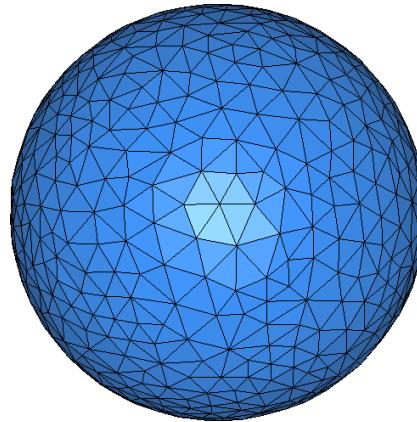


spheres

Modeling a Sphere

Recall two possibilities:

- Center point and radius
- Triangle mesh



**approximate
sphere**



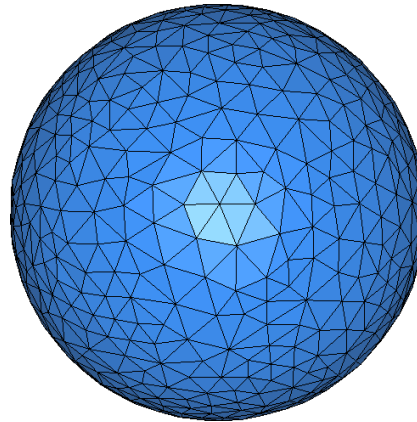
spheres

which is better?

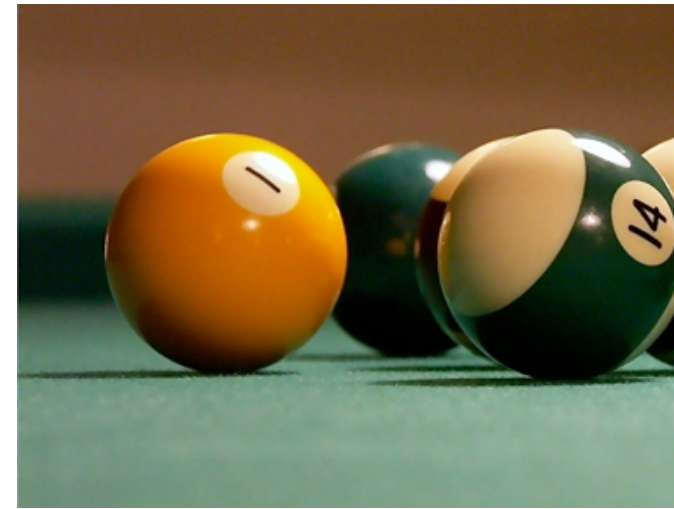
Modeling a Sphere

Recall two possibilities:

- Center point and radius
- Triangle mesh



**approximate
sphere**



spheres

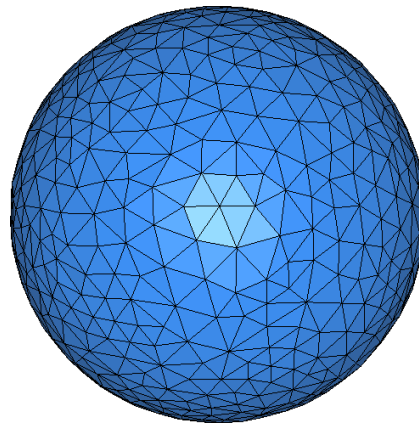
which is better?

what does "better" mean?

Modeling a Sphere

Recall two possibilities:

- Center point and radius
- Triangle mesh



**approximate
sphere**



spheres

which is better?

This is a choice of data structures.

what does "better" mean?

What's important to us?

Modeling

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

center + radius

triangle mesh

Modeling

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

center + radius

vs

triangle mesh

precision/accuracy

flexibility/generalizability

+ Storage

+ ease of editing

- compute (?)

ease of storing other properties (than geom.)

+ physics-friendly

+ ease of modeling

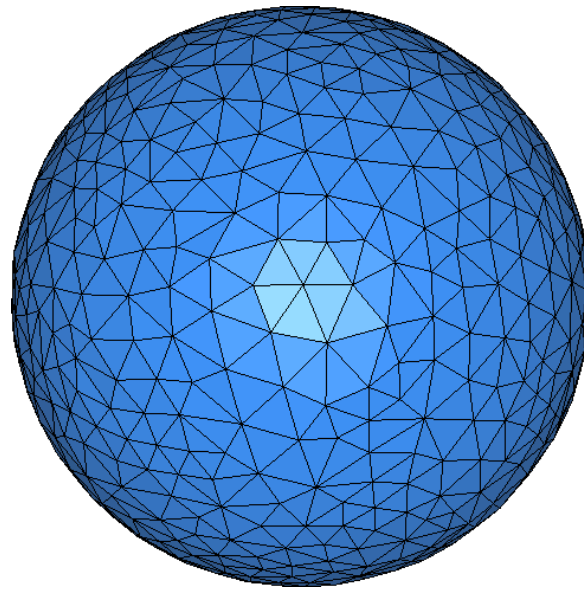
rendering efficiency

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)

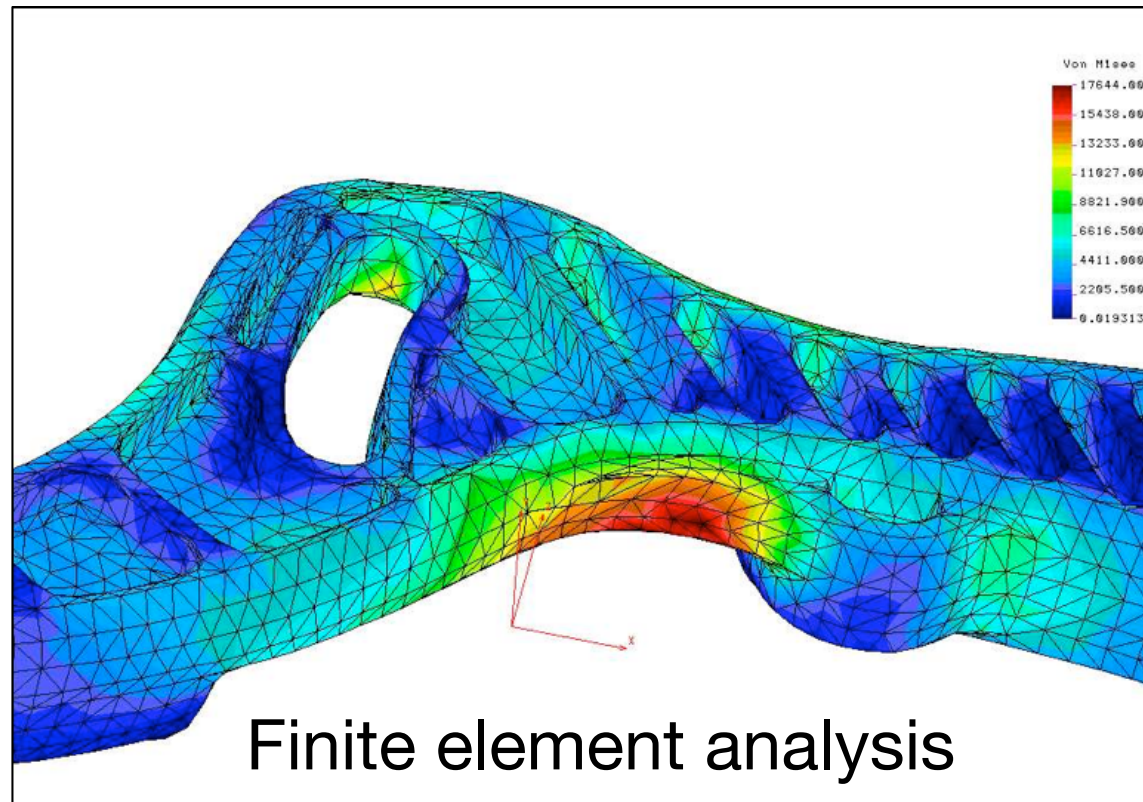
Meshes - Advantages

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



Meshes - Advantages

- Approximate arbitrary geometry
- Enables storage of surface properties beyond geometry



Meshes - Advantages

- Makes for cool architecture

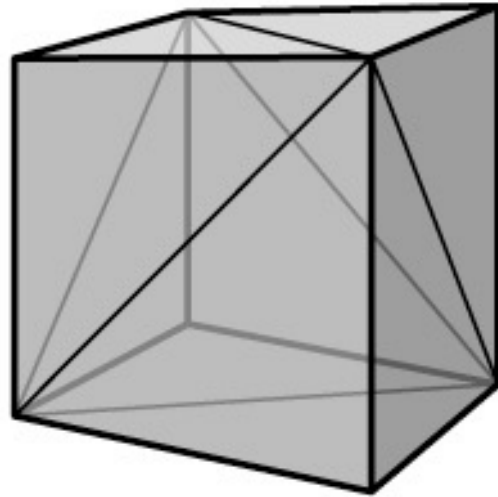


Ottawa Convention Center



Amazon Spheres Complex

A small mesh

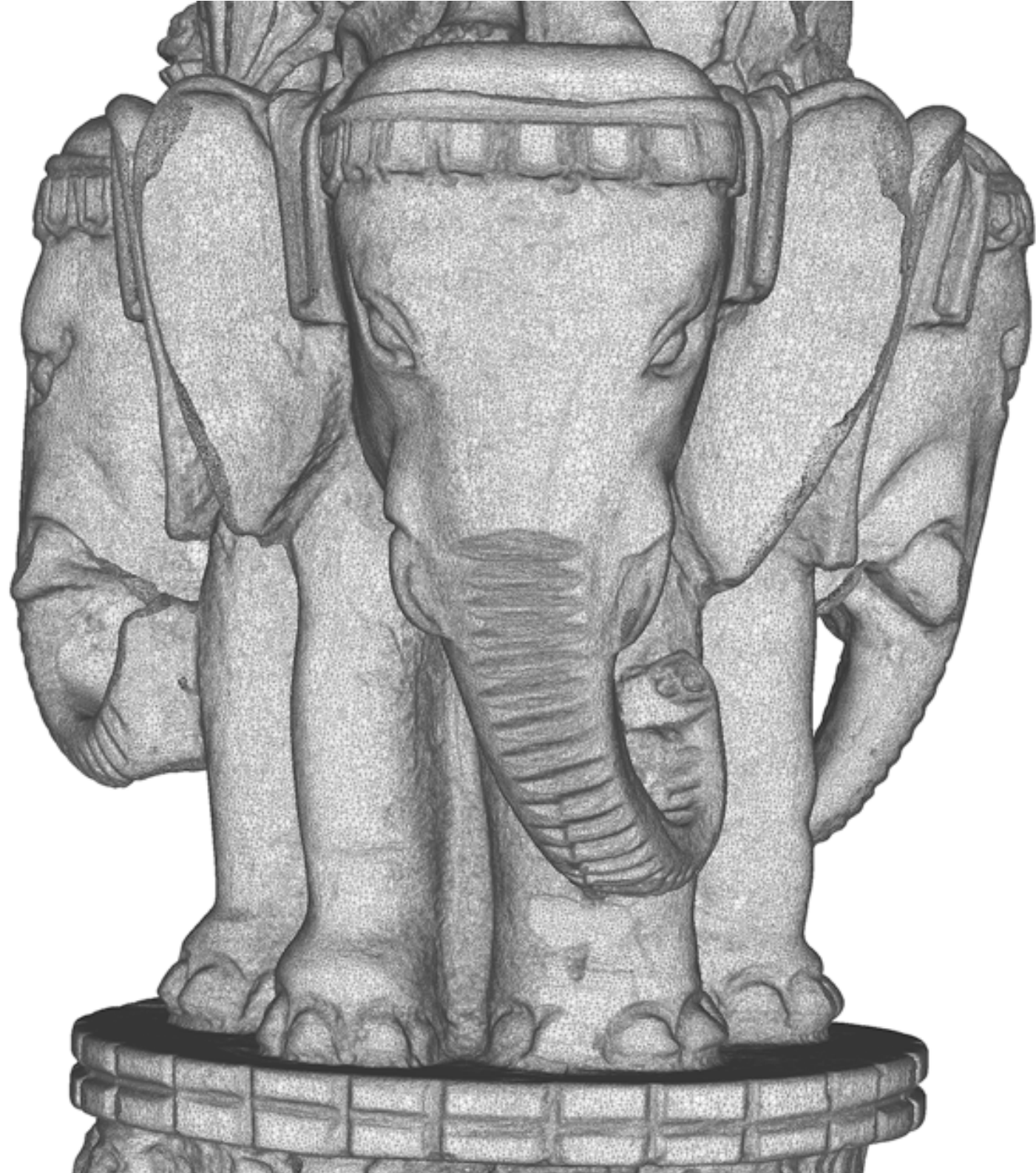


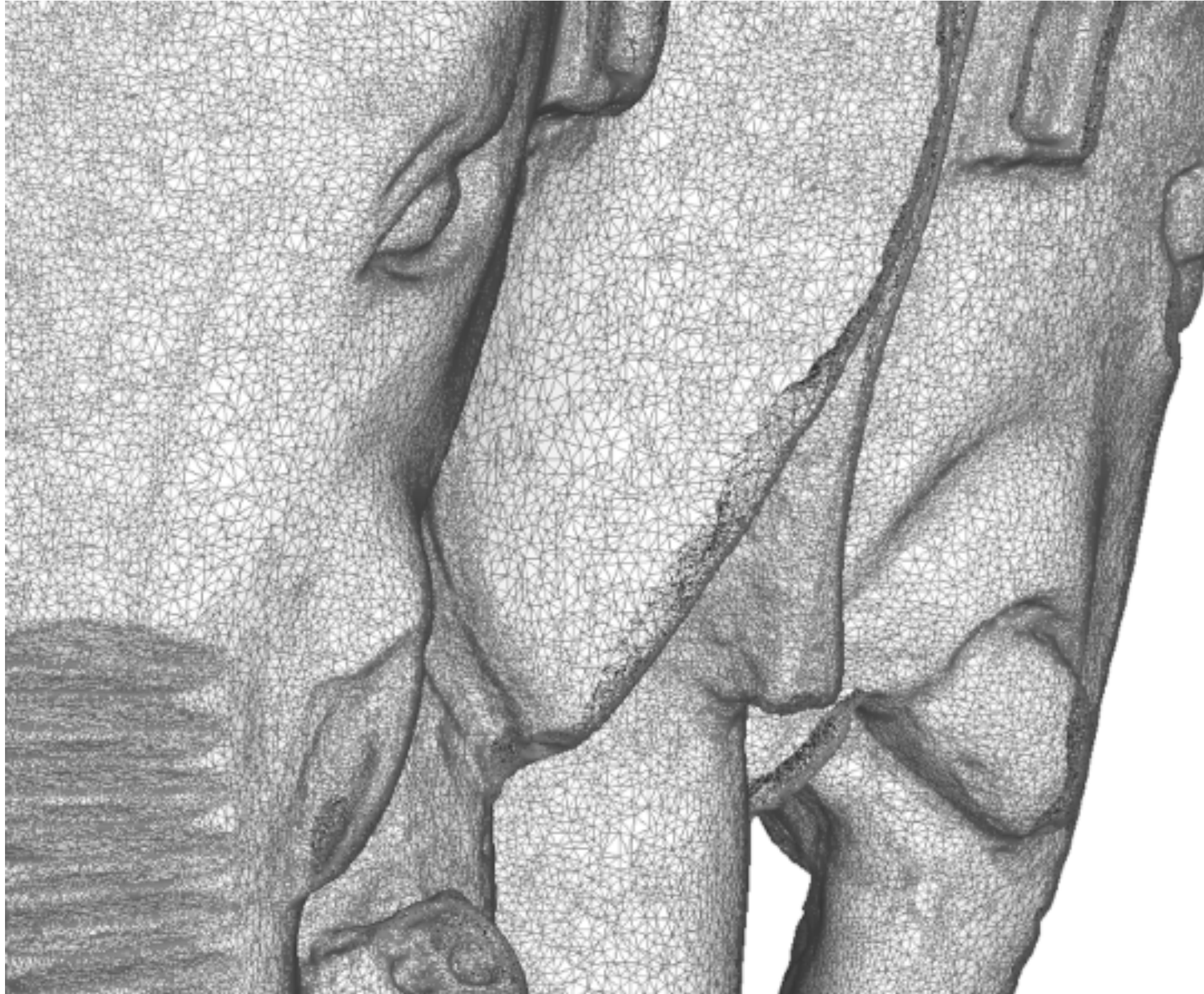
12 triangles, 8 vertices

A large mesh

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

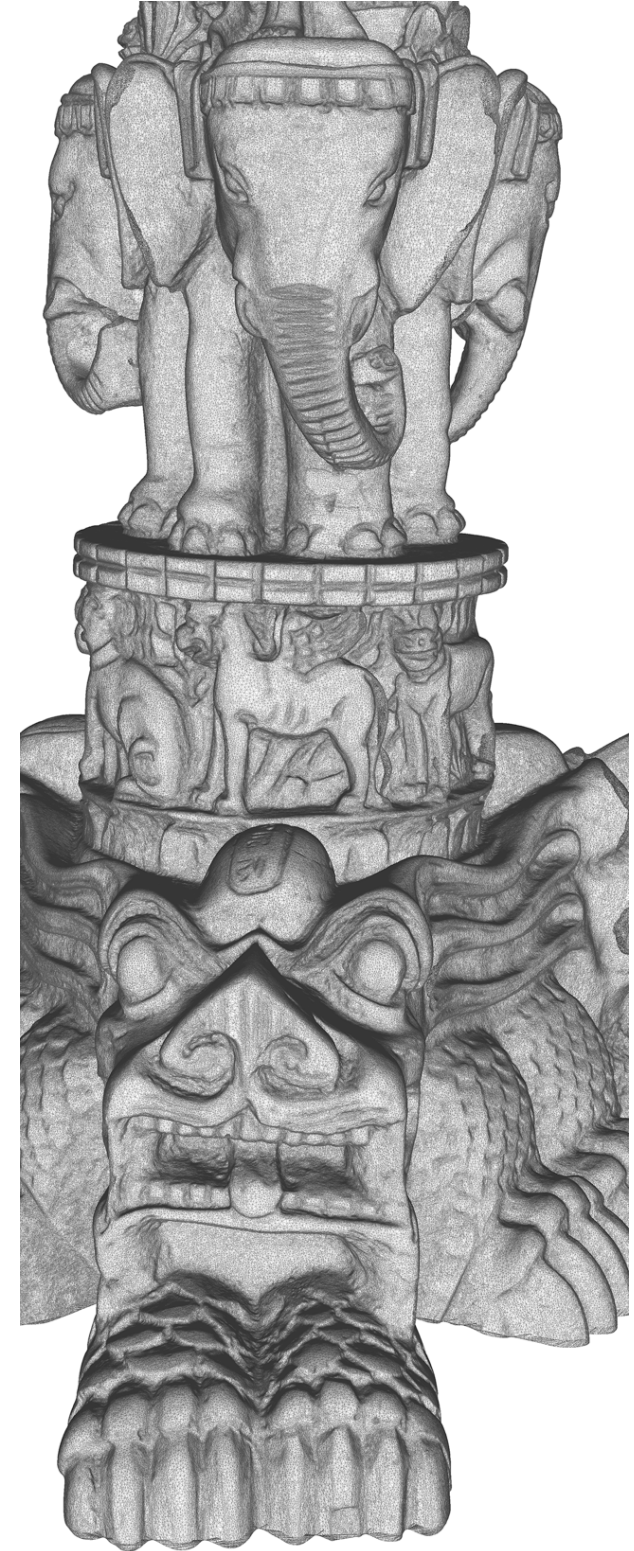






A large mesh

- 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”)

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”)

**Take a minute to consider:
why not quadrilaterals? Other polygons?**

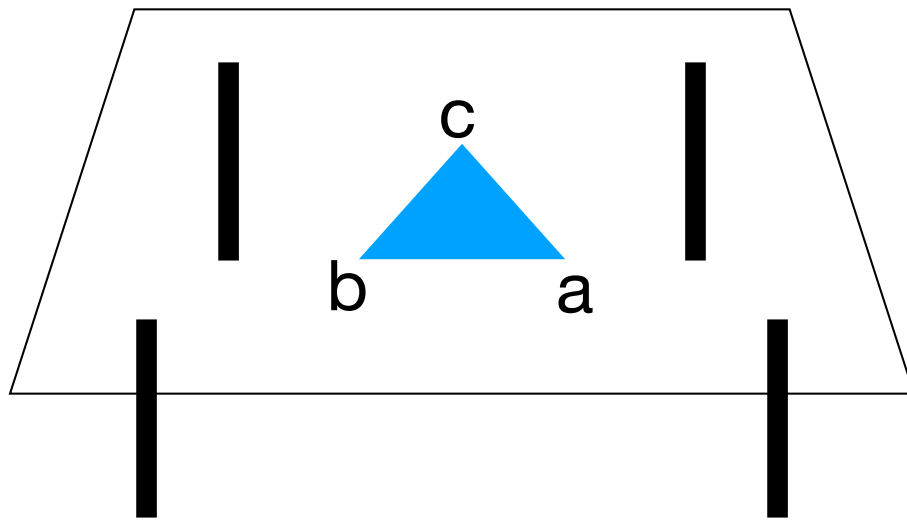
Why not use other polygons?

Why not use other polygons?

- Some systems do!
 - More common in modeling than rendering.
- Triangles are nice:
 - simplest possible polygon (makes rendering code easier!)
 - 3 vertices are always coplanar
 - always convex
 - any other polygon can be **triangulated**

Let's talk about triangles

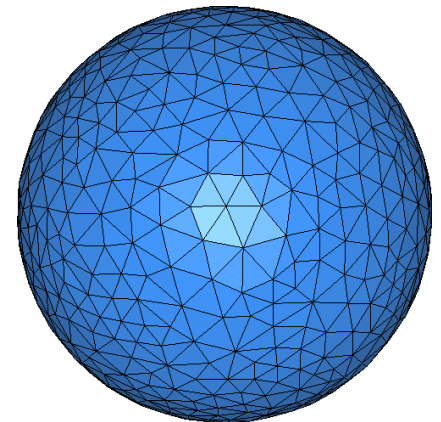
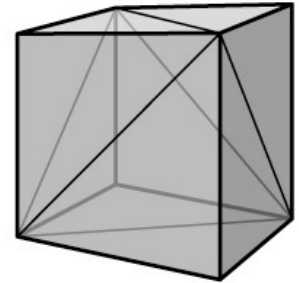
The triangle below sits **face down** on the table - which of the following does **not** describe the triangle?



- A. abc
- B. bca
- C. cab
- D. acb

Triangle Meshes

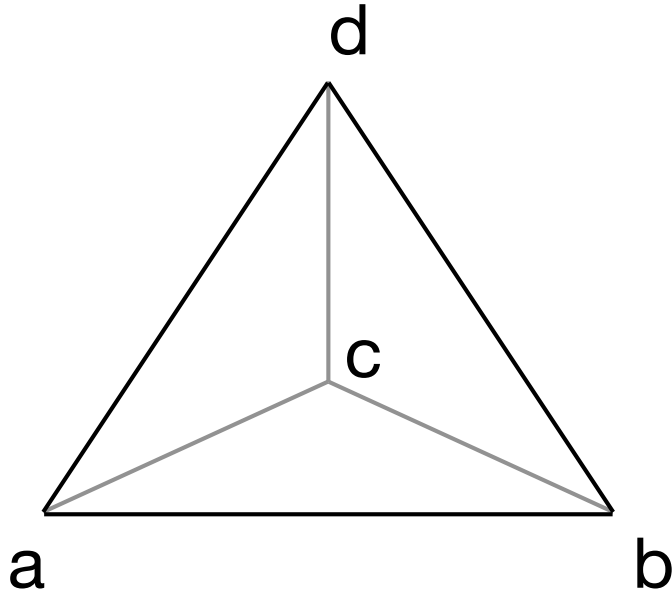
- A bunch of triangles in 3D space that are **connected together** to form a surface
- Geometrically, a mesh is a piecewise planar surface
 - almost everywhere, it is planar
 - exceptions are at the edges where triangles join
- Often, it's a piecewise planar **approximation of a smooth surface**
 - in 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?

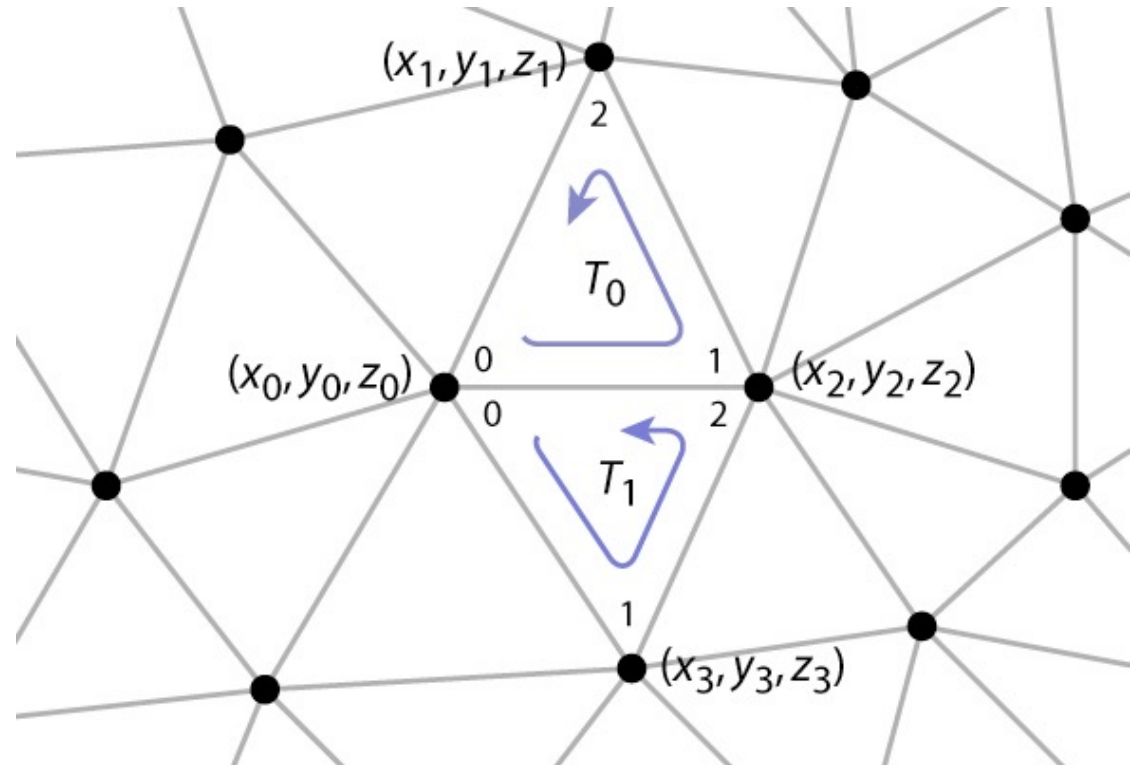
Example: a tetrahedron



illusion disambiguation: *c* is *behind* the triangle *abd*

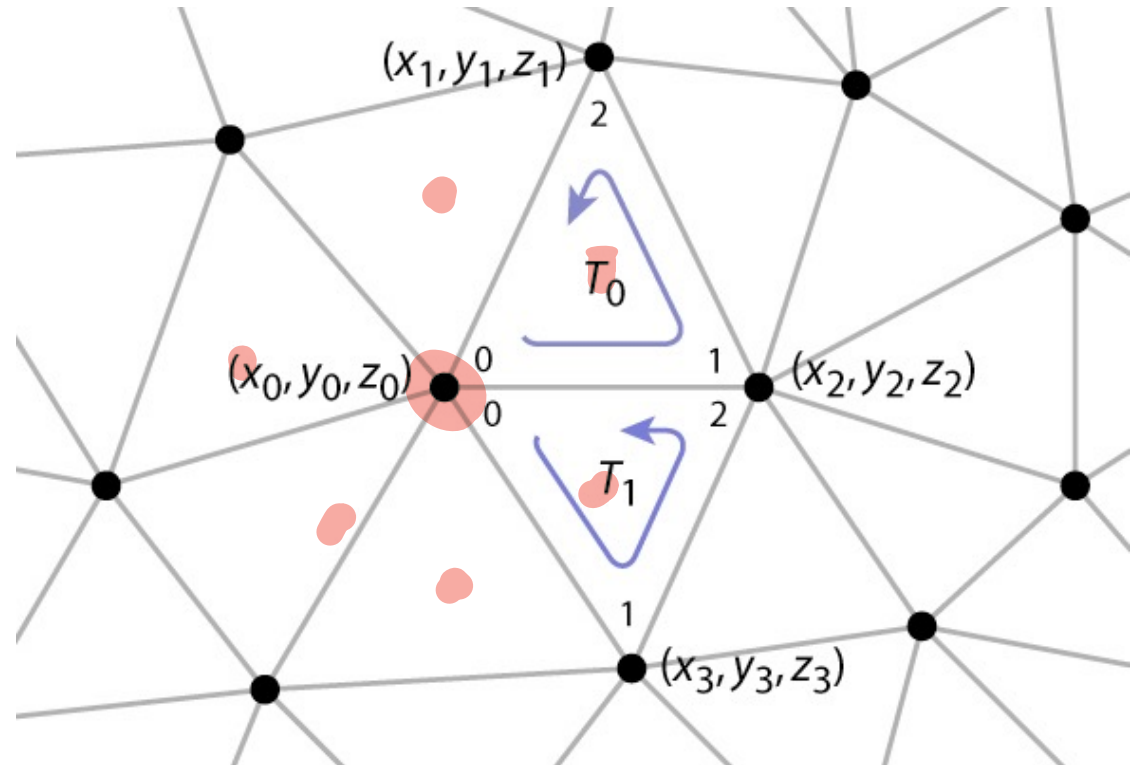
Separate Triangles

	[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
	\vdots	\vdots	\vdots



Separate Triangles

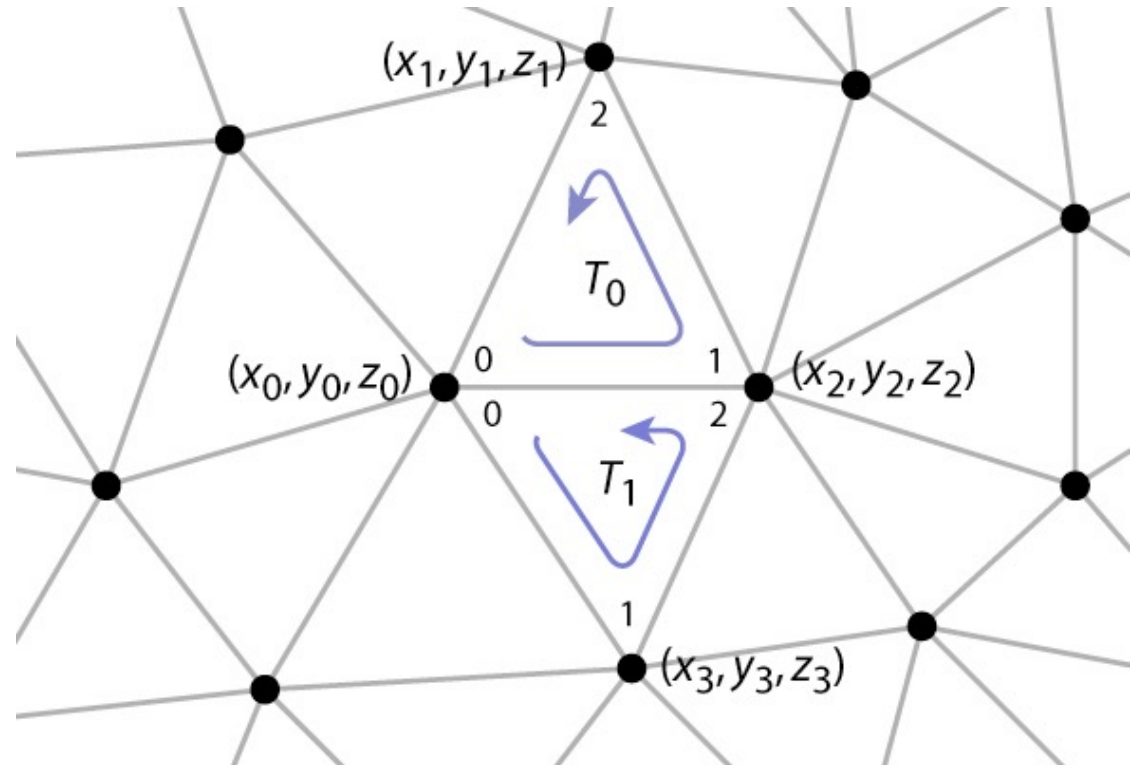
	[0]	[1]	[2]
tris[0]	x_0, y_0, z_0	x_2, y_2, z_2	x_1, y_1, z_1
tris[1]	x_0, y_0, z_0	x_3, y_3, z_3	x_2, y_2, z_2
	\vdots	\vdots	\vdots



Problems:

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
	\vdots	\vdots	\vdots

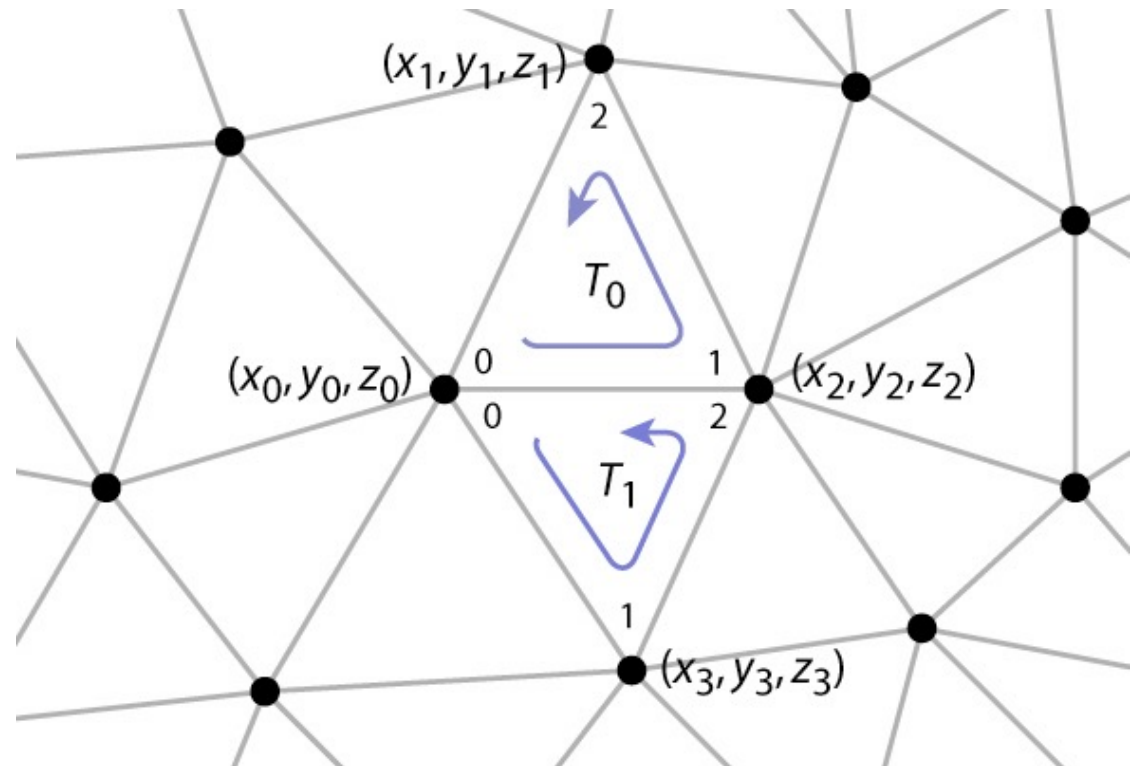


Problems:

- Wastes space

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
	\vdots	\vdots	\vdots






Problems:

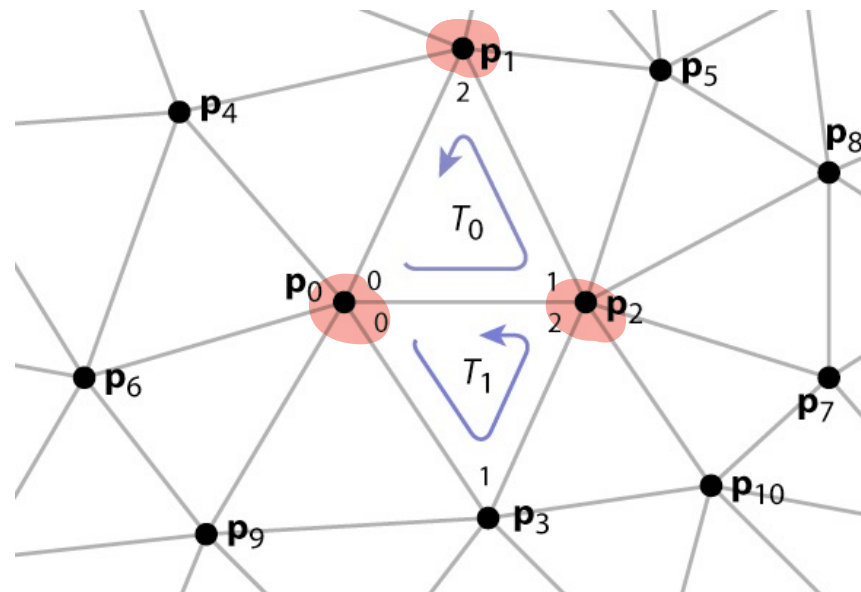
- Wastes space
- Repeated floats with different round-off creates problems:
 - Cracks in the mesh
 - Finding neighbors may fail

Indexed Triangle Set (A1)

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

	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
		\vdots

tInd[0]	0, 2, 1
tInd[1]	0, 3, 2
	\vdots

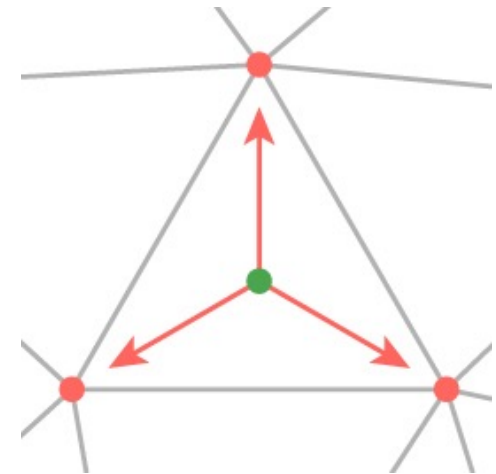
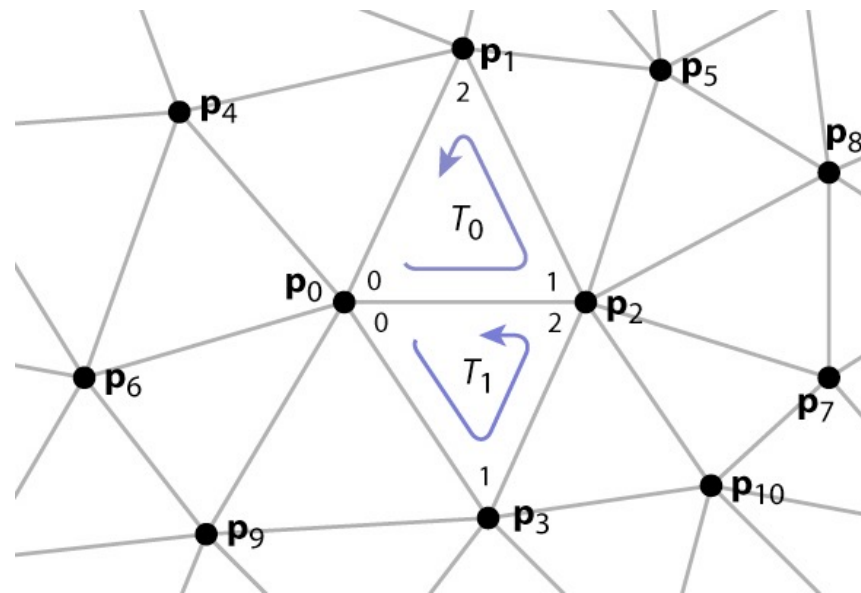


Indexed Triangle Set (A1)

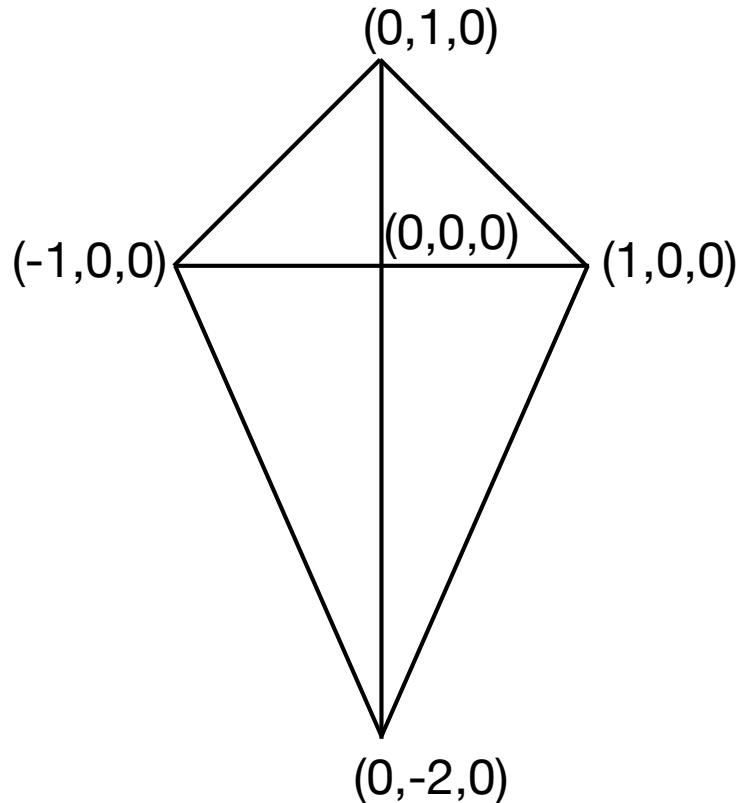
- Vertices are listed once, without duplicates
- Each Triangle stores **indices** of its vertices

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
	\vdots

tInd[0]	0, 2, 1
tInd[1]	0, 3, 2
	\vdots



Problems: Kite Mesh



Represent this surface using:

1. Separate triangles.
2. Indexed triangle set.

Note: all the triangles are facing towards you in the view shown.

Group work task: Complete Problems 1 and 2

Storage Cost?

- Always depends on the geometry, but for contiguous surface meshes, indexed triangle sets usually give large space savings.
 - Exercise: verify this on the tetrahedron example

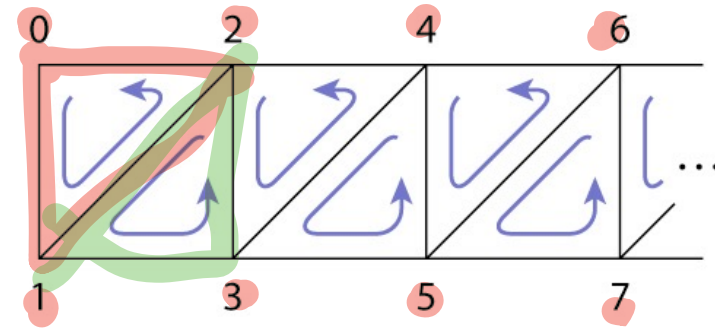
Triangle Strips

- Takes advantage of mesh properties:

- Each triangle is usually adjacent to previous

- Next triangle reuses previous two vertices

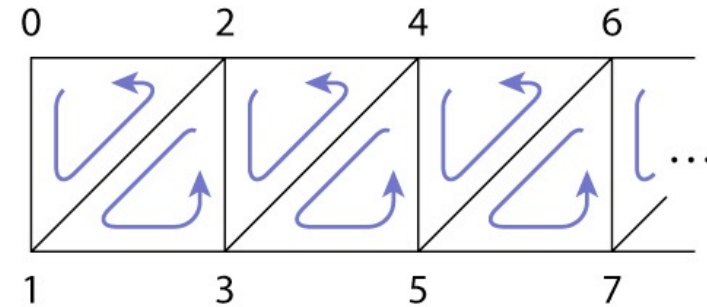
- Every subsequence of 3 vertices is a triangle



For long strips, about one index per triangle!

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

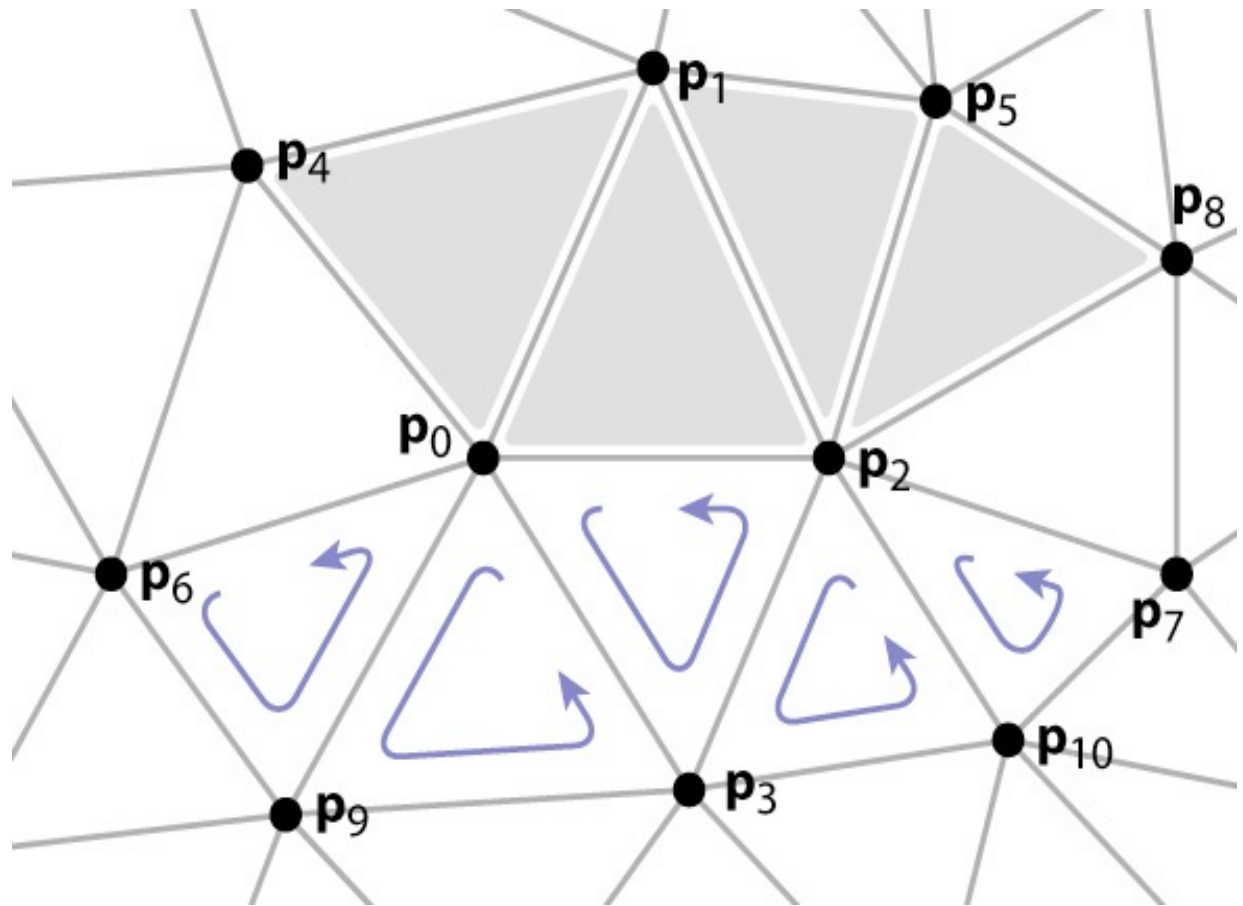
verts[0]	x_0, y_0, z_0
verts[1]	x_1, y_1, z_1
	x_2, y_2, z_2
	x_3, y_3, z_3
	\vdots

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

Triangle Strips

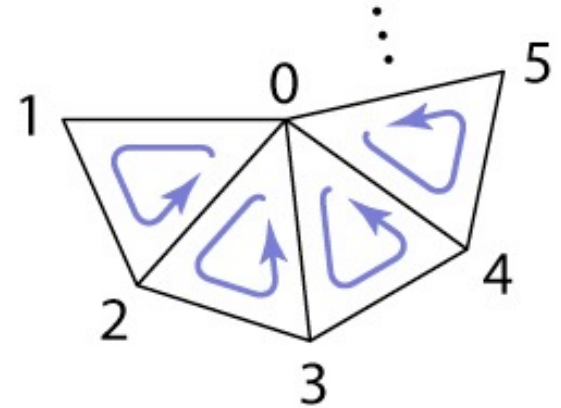
verts[0]	x_0, y_0, z_0
verts[1]	x_1, y_1, z_1
	x_2, y_2, z_2
	x_3, y_3, z_3
	\vdots

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



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



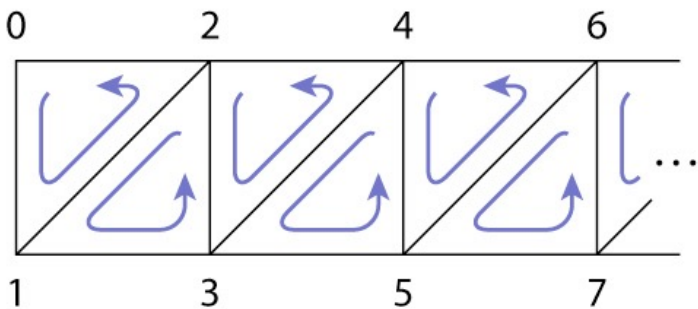
$$0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5$$
$$(0 \ 1 \ 2) (0 \ 2 \ 3) (0 \ 3 \ 4) (0 \ 4 \ 5)$$

What else?

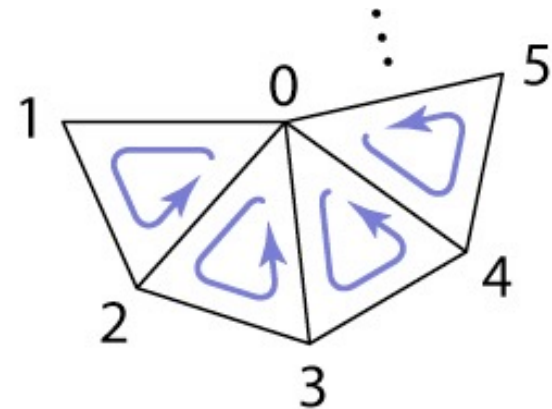
- Indexed triangle sets are good for rendering, but not great for mesh **processing**.
- What if we want to efficiently find:
 - all triangles containing a vertex?
 - all triangles adjacent to a triangle?
 - the triangle across a particular edge of a triangle?
- You can augment the mesh data structure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

Problems 3-5

Triangle Strip:



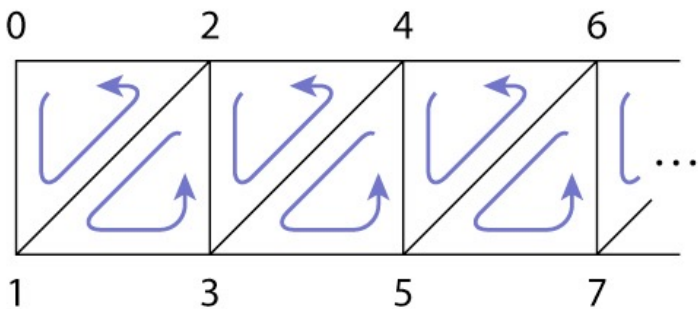
Triangle Fan:



Group work task: Complete Problems 3-5

Problems 3-5

Triangle Strip:



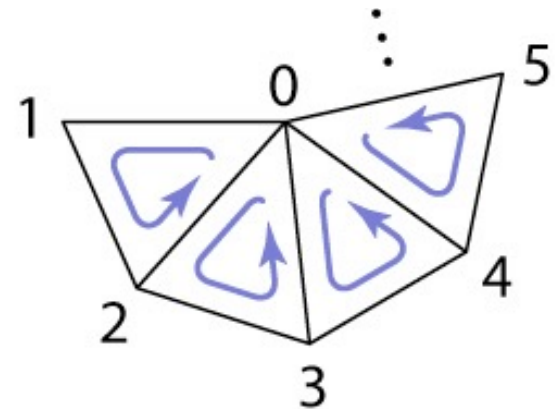
Vertex sequence

0, 1, 2, 3, 4, 5, 6, 7, ...

leads to triangle sequence:

(0 1 2), (2 1 3), (2 3 4), (4 3 5), ...

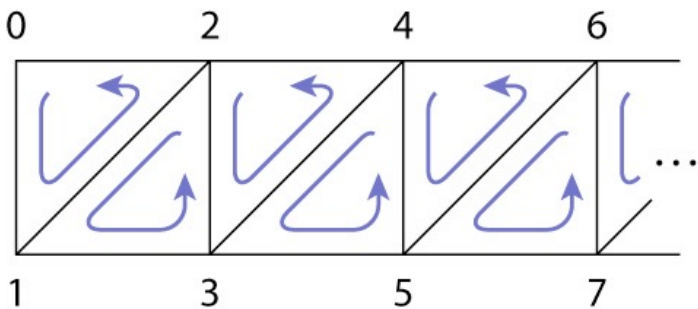
Triangle Fan:



Group work task: Complete Problems 3-5

Problems 3-5

Triangle Strip:



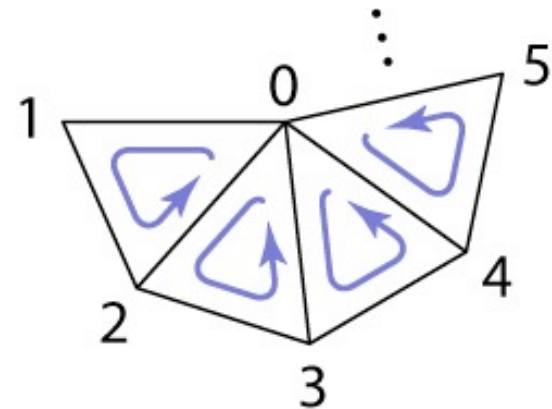
Vertex sequence

0, 1, 2, 3, 4, 5, 6, 7, ...

leads to triangle sequence:

(0 1 2), (2 1 3), (2 3 4), (4 3 5), ...

Triangle Fan:



Vertex sequence

0, 1, 2, 3, 4, 5, ...

leads to triangle sequence:

(0 1 2), (0 2 3), (0 3 4), (0 4 5), ..

Group work task: Complete Problems 3-5