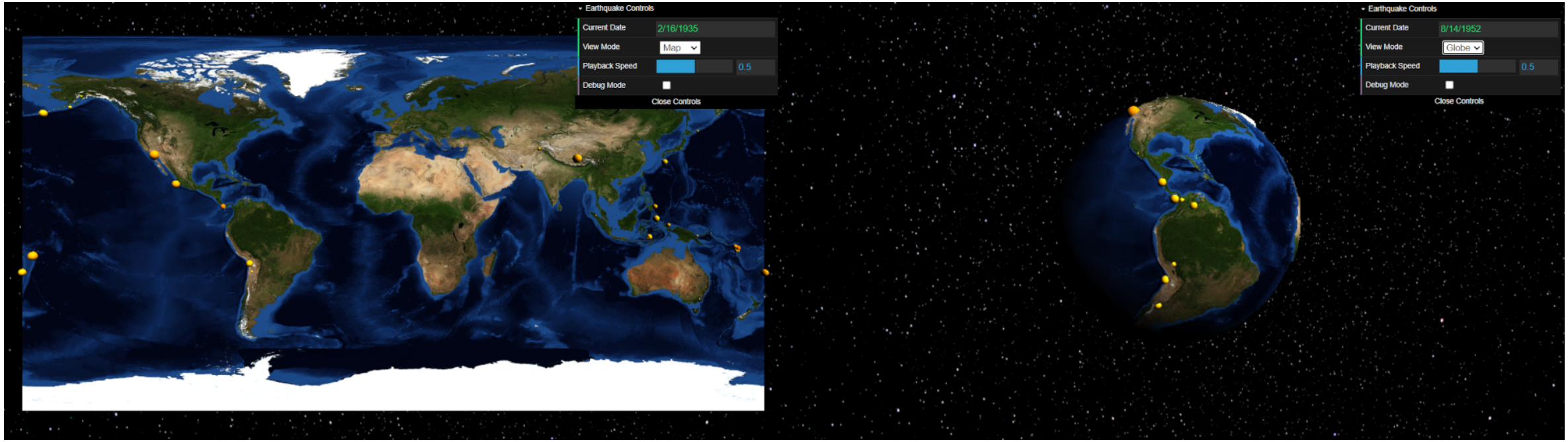
UNIVERSITY OF MINNESOTA
Driven to Discover®

3D Polygonal Modeling

CSCI 4611: Programming Interactive Computer Graphics and Games

Evan Suma Rosenberg | CSCI 4611 | Fall 2022

Assignment 3: Earthquake Visualization



Cool Datasets

- **Centennial Earthquake Catalog**
  https://earthquake.usgs.gov/

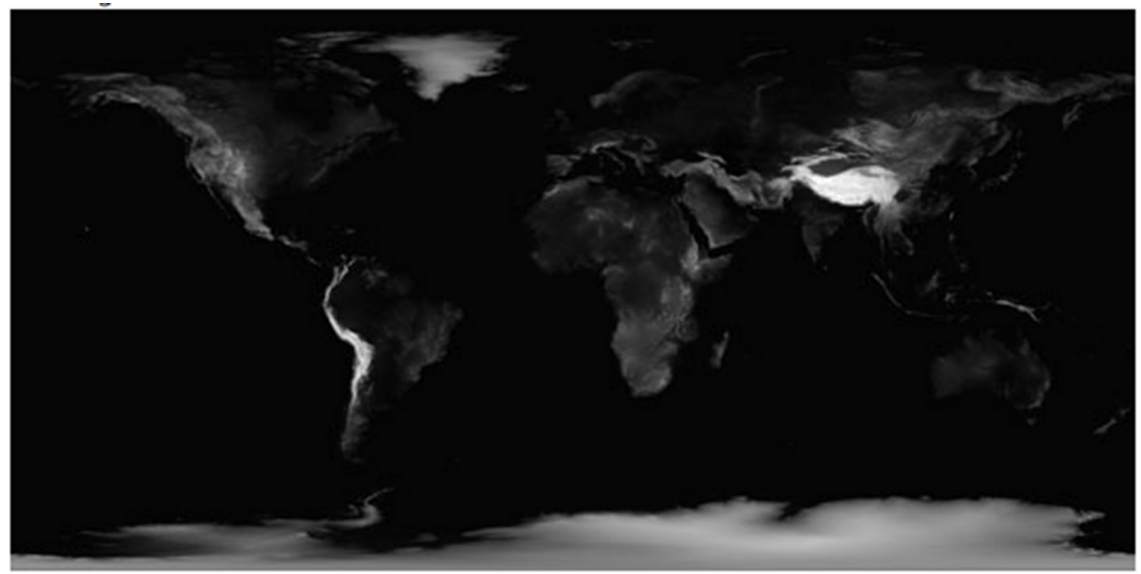
- **NASA Visible Earth: Blue Marble**
  https://visibleearth.nasa.gov/collection/1484/blue-marble

Centennial Earthquake Catalog

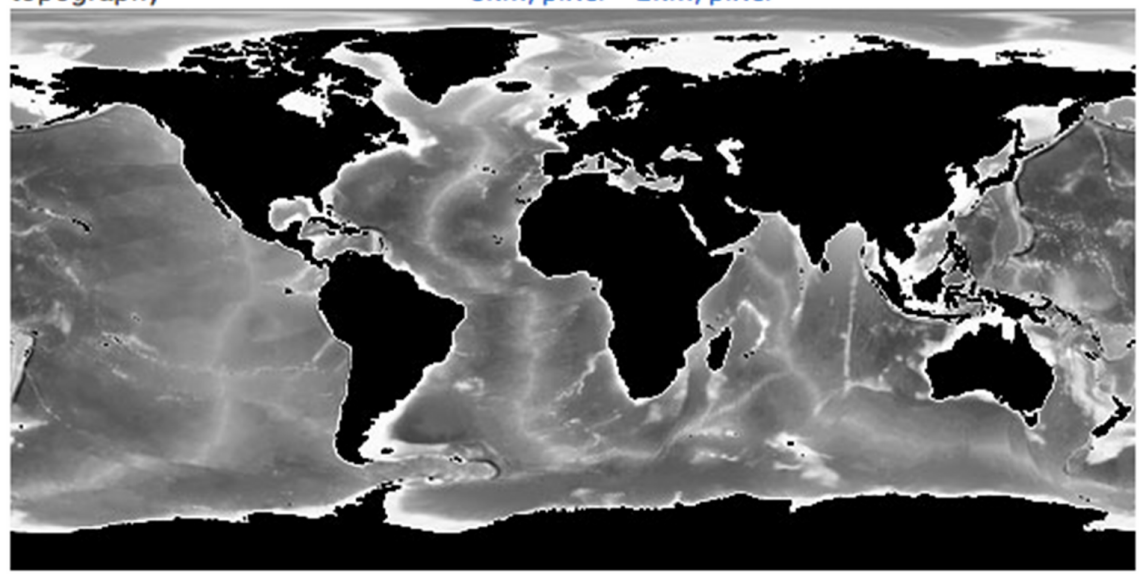
- Public domain
- More than 13,000 earthquakes
- Longitude, latitude, magnitude, data, time, etc.
- “For recent years (1964–present) a cut-off magnitude of 5.5 has been chosen for the catalog, and the catalog is complete down to that threshold.”

NASA Visible Earth: Blue Marble

- Monthly images of Earth with clouds removed
- Lowest resolution available is 5400x2700 (8km/px)
- Highest resolution is 500m/px
- Topography and bathymetry
- Also public domain



topography 8km/pixel 2km/pixel



bathymetry 8km/pixel 2km/pixel

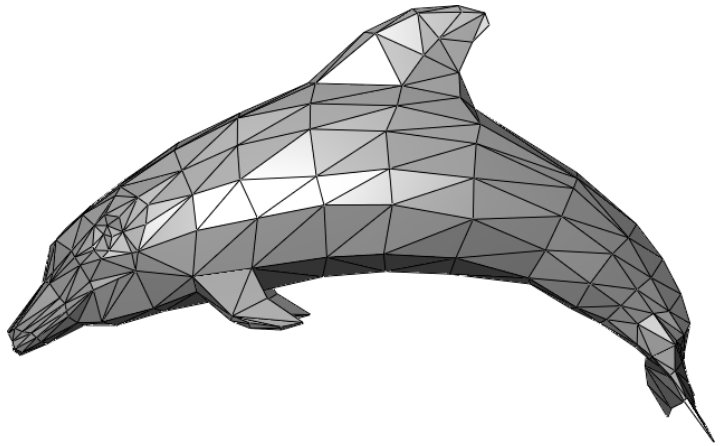
Assignment 3 Objectives

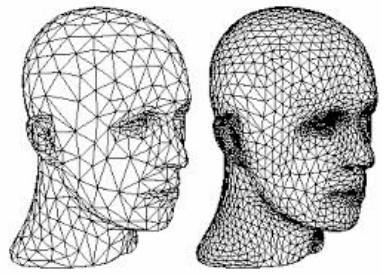
- Create polygon models for a planar Earth and spherical Earth
- Texture map each mesh with the satellite imagery of the Earth
- Visualize earthquakes and their magnitude at the correct time and location on the Earth
- Linearly interpolate (lerp) between the points and vertices that make up each model

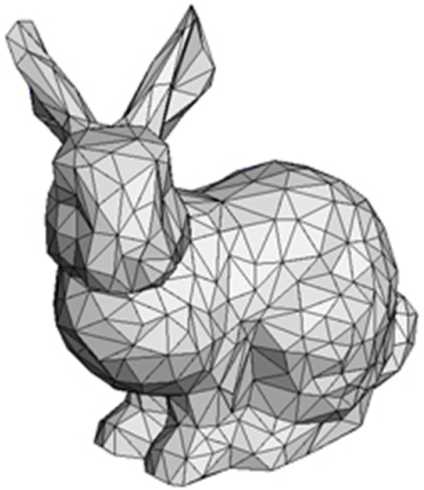
Live Demo

https://csci-4611-fall-2022.github.io/Builds/Assignment-3

Triangular Meshes







Drawing Triangles in 2D (OpenGL)

```
beginShape(TRIANGLES);
vertex(30, 75);
vertex(40, 20);
vertex(50, 75);
vertex(60, 20);
vertex(70, 75);
vertex(80, 20);
endShape();
```

```
beginShape(TRIANGLE_STRIP);
vertex(30, 75);
vertex(40, 20);
vertex(50, 75);
vertex(60, 20);
vertex(70, 75);
vertex(80, 20);
vertex(90, 75);
endShape();
```

```
beginShape(TRIANGLE_FAN);
vertex(57.5, 50);
vertex(57.5, 15);
vertex(92, 50);
vertex(57.5, 85);
vertex(22, 50);
vertex(57.5, 15);
endShape();
```

1. Tell the graphics engine how to interpret the vertex data that comes next.
2. List the coordinates of the vertices.

Drawing Triangles in 2D (OpenGL)

```
beginShape(TRIANGLES);
vertex(30, 75);
vertex(40, 20);
vertex(50, 75);
vertex(60, 20);
vertex(70, 75);
vertex(80, 20);
endShape();
```

```
beginShape(TRIANGLE_STRIP);
vertex(30, 75);
vertex(40, 20);
vertex(50, 75);
vertex(60, 20);
vertex(70, 75);
vertex(80, 20);
vertex(90, 75);
endShape();
```

```
beginShape(TRIANGLE_FAN);
vertex(57.5, 50);
vertex(57.5, 15);
vertex(92, 50);
vertex(57.5, 85);
vertex(22, 50);
vertex(57.5, 15);
endShape();
```

1. Triangle Primitive Type

2. Vertex Stream

Drawing Triangles in 3D (OpenGL)

https://www.khronos.org/opengl/wiki/Primitive

Triangle Primitive Types

- **GL_TRIANGLES:** Vertices 0, 1, and 2 form a triangle. Vertices 3, 4, and 5 form a triangle. And so on.
- **GL_TRIANGLE_STRIP:** Every group of 3 adjacent vertices forms a triangle. The face direction of the strip is determined by the winding of the first triangle. Each successive triangle will have its effective face order reversed, so the system compensates for that by testing it in the opposite way. A vertex stream of n length will generate n-2 triangles.
- **GL_TRIANGLE_FAN:** The first vertex is always held fixed. From there on, every group of 2 adjacent vertices form a triangle with the first. So with a vertex stream, you get a list of triangles like so: (0, 1, 2) (0, 2, 3), (0, 3, 4), etc. A vertex stream of *n* length will generate *n-2* triangles.

Vertex Stream

(x1, y1, z1)
(x2, y2, z2)
(x3, y3, z3)

Problem: Efficiency

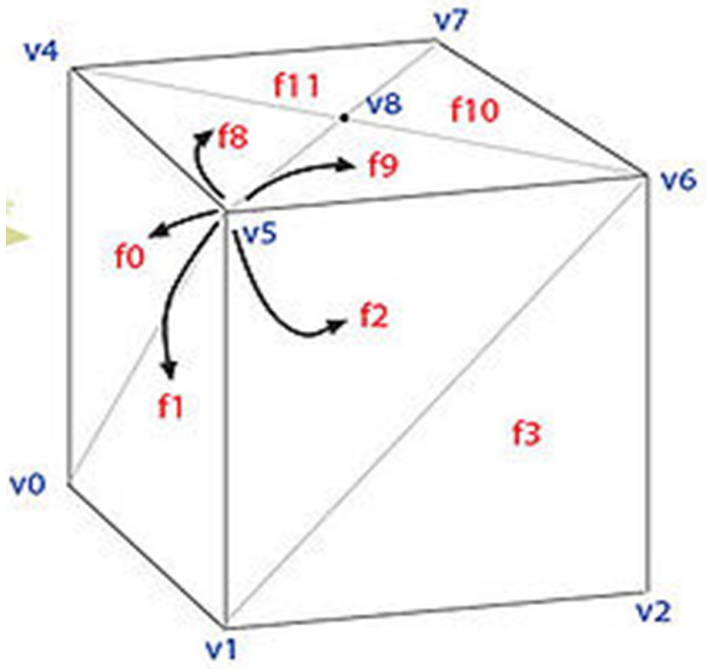
- **Our triangle meshes might be quite large.**
- **Converting into triangle strips and fans helps.**

  However, this is tedious in some cases, and it adds complexity to the data structures we need to store meshes.

- **The real inefficiency in GL_TRIANGLES is that adjacent triangles usually share vertices.**

  If we interpret each set of 3 vertices as a triangle, that means we need to repeat all the shared vertices!

Solution: Indexed Triangles



Vertex Buffer

- Lists x,y,z position of each vertex once.

0	0,0,0
1	1,0,0
2	1,1,0
3	0,1,0
4	0,0,1
5	1,0,1
6	1,1,1
7	0,1,1
8	0.5,0.5,0
9	0.5,0.5,1

Index Buffer

- Each consecutive set of 3 indices defines a triangle.
- The indices are like pointers into the vertex buffer.

0	0
1	5
2	4
3	0
4	1
5	5
6	1
7	6
8	5
9	1
10	6
11	2
12	2
13	7
14	6

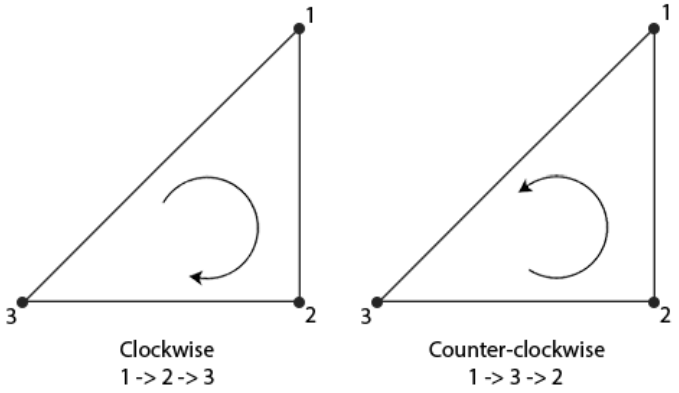
15	2
16	3
17	7
18	3
19	4
20	7
21	3
22	0
23	4
24	8
25	4
26	5
27	8
28	5
29	6

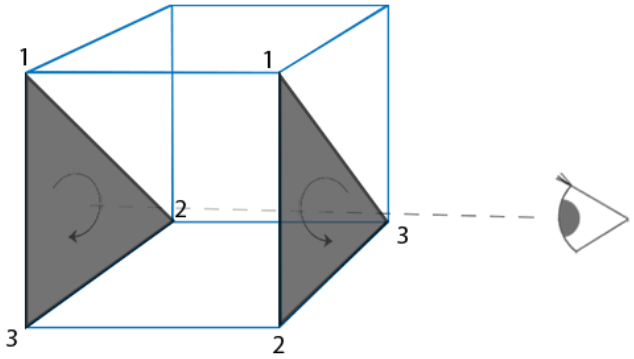
30	8
31	6
32	7
33	8
34	7
35	4
36	9
37	4
38	5
39	9
40	5
41	6
42	9
43	6
44	7

45	9
46	7
47	4

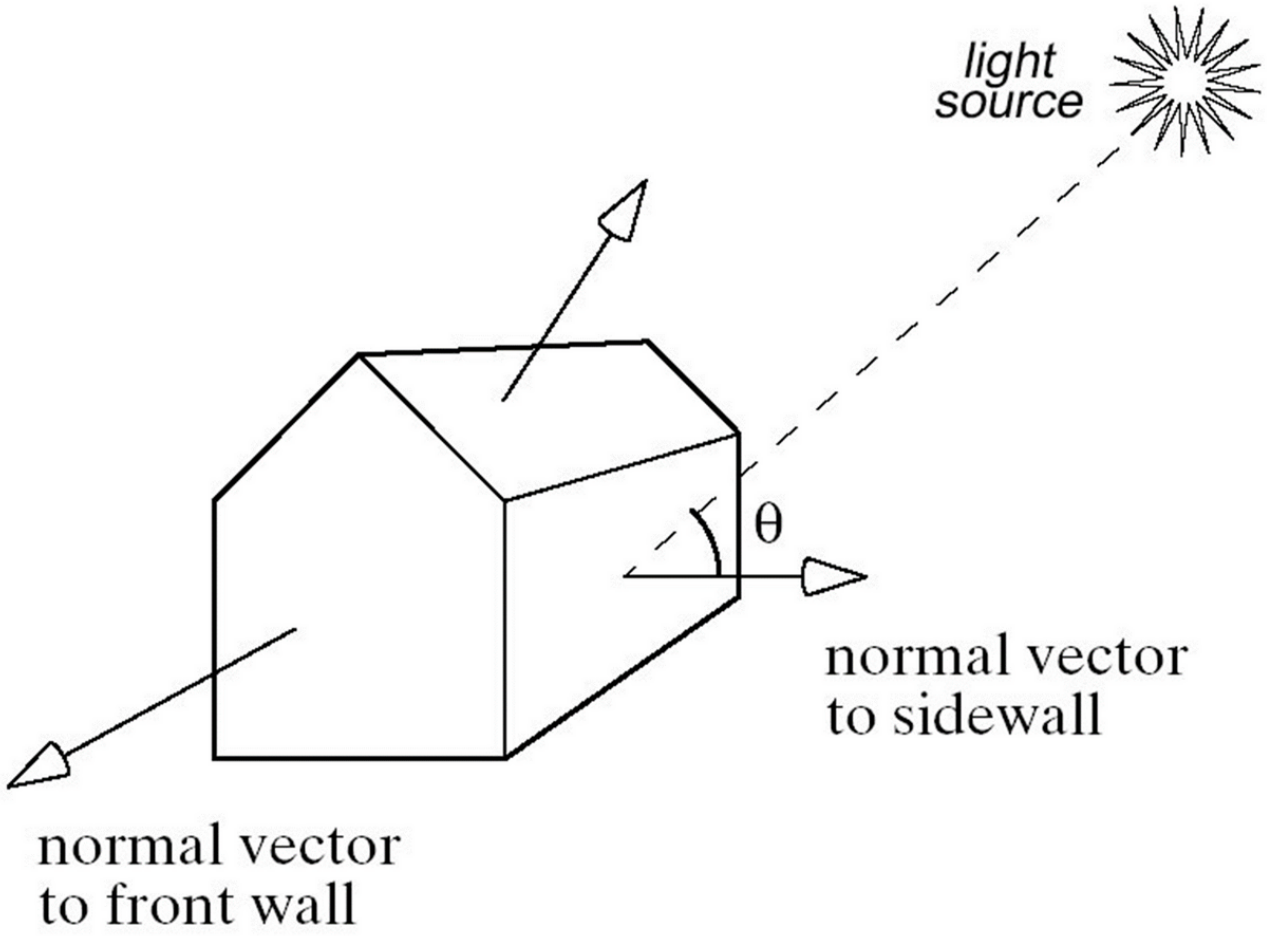
Winding Order

- When we index a set of triangle vertices, we list them in a certain **winding order**.
- The "front face" of each triangle is the one where the vertices wind in counterclockwise order.
- Use the right hand rule – your thumb should point "up" out of the front face of each triangle.

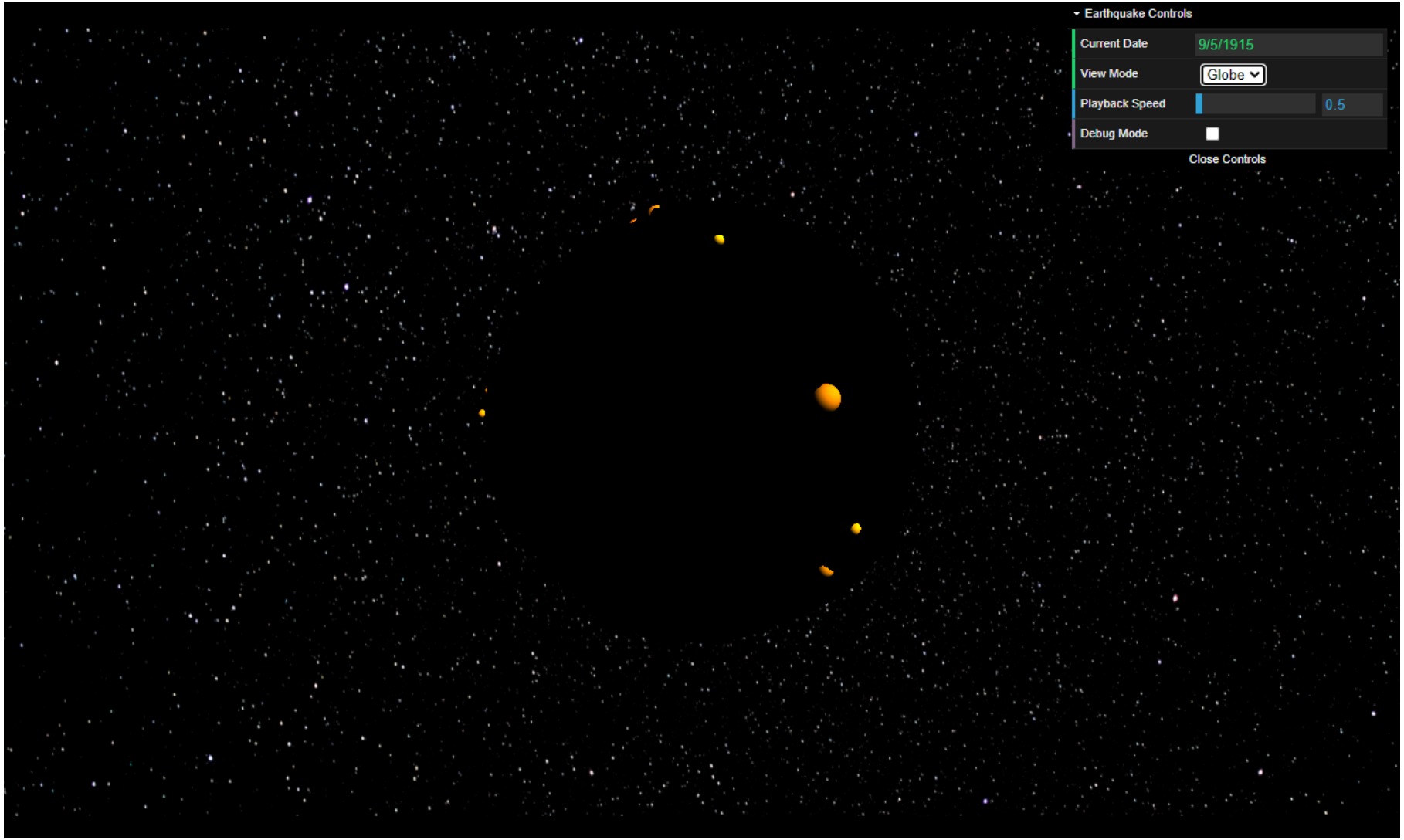




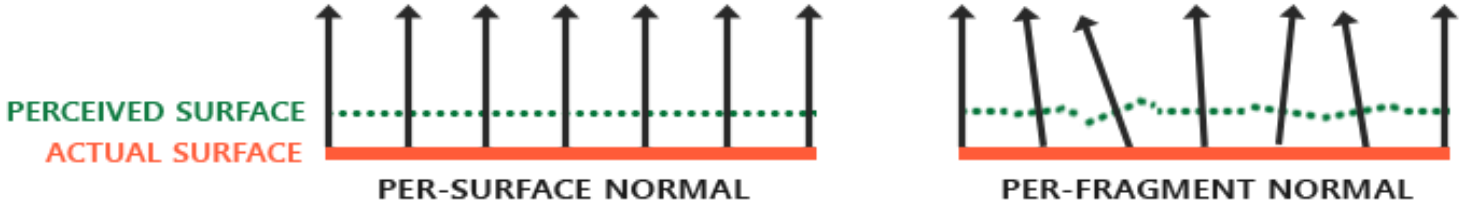
Normals

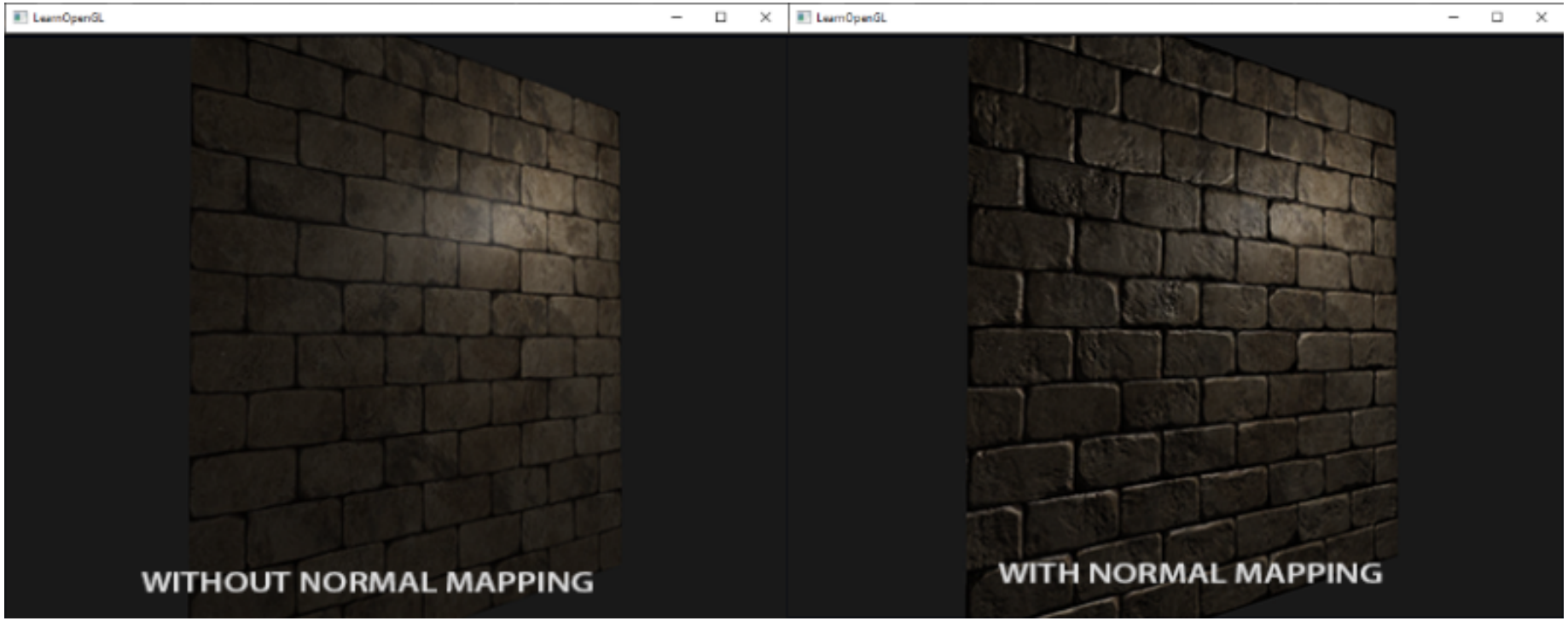

light
source
θ
normal vector
to sidewall
normal vector
to front wall

Earthquake Visualization without Normals



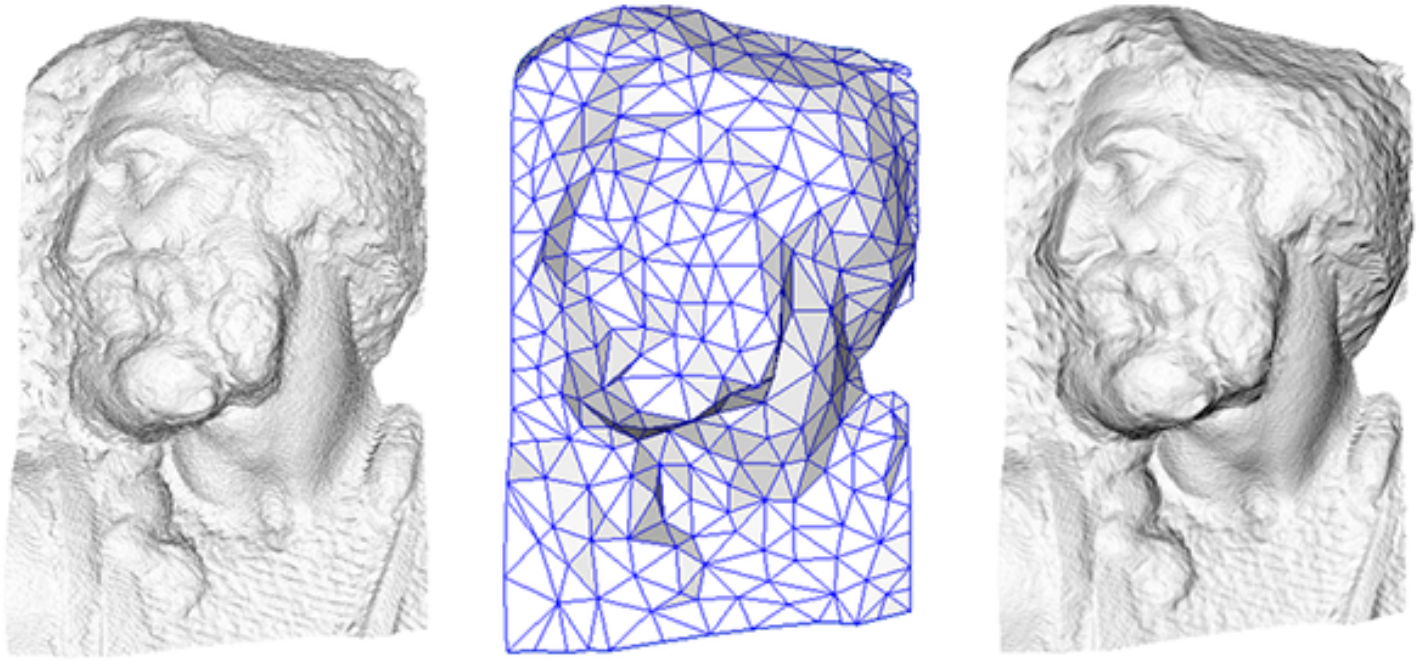
Normal Mapping





https://learnopengl.com/Advanced-Lighting/Normal-Mapping

Normal Mapping



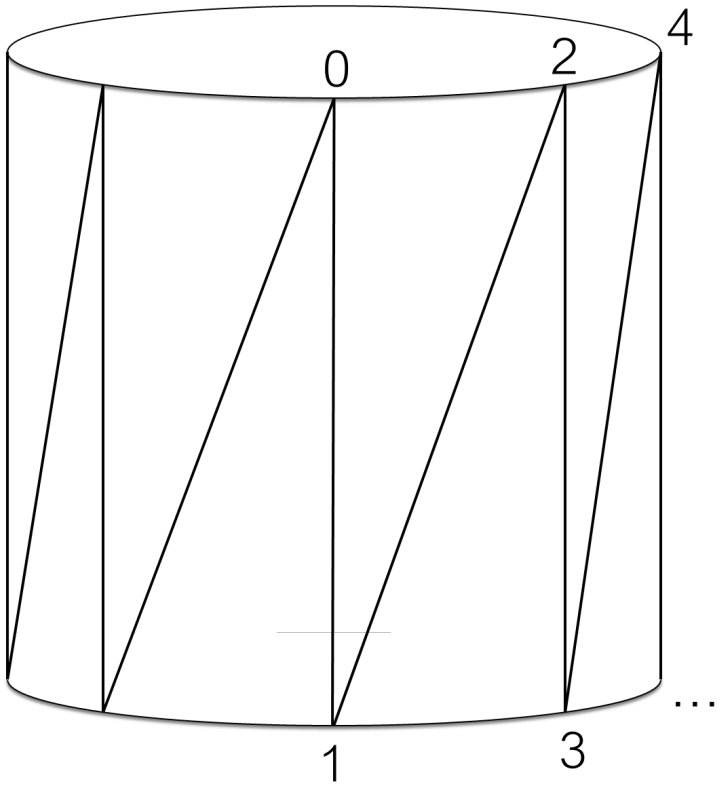
original mesh
4M triangles

simplified mesh
500 triangles

simplified mesh
and normal mapping
500 triangles

https://learnopengl.com/Advanced-Lighting/Normal-Mapping

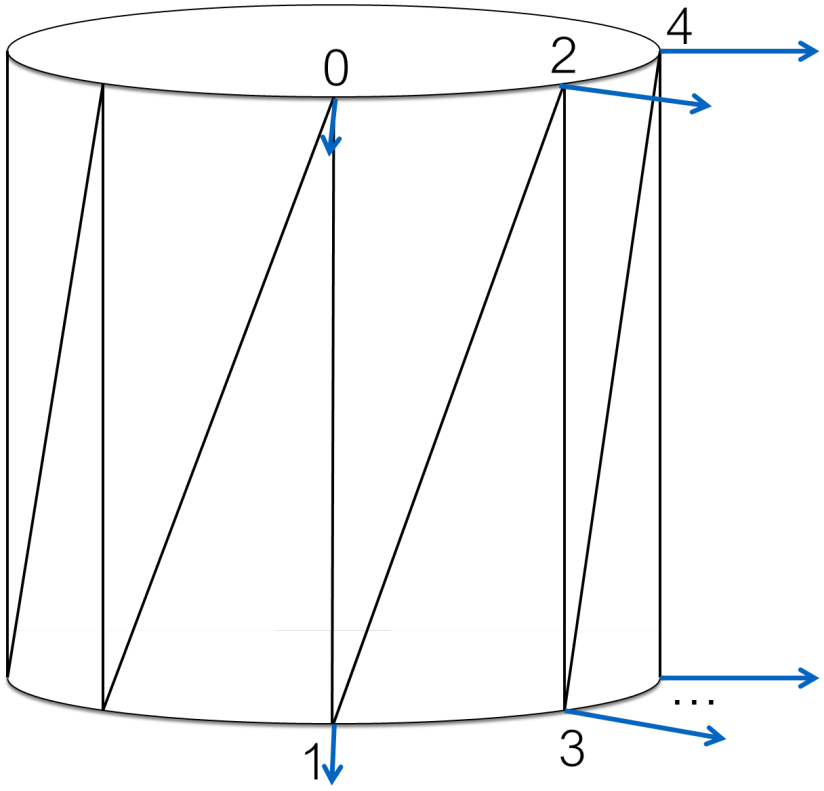
Cylinder Example



	vertices
0	cos(0), 1, sin(0)
1	cos(0), 0, sin(0)
2	cos(45), 1, sin(45)
3	cos(45), 0, sin(45)
4	cos(90), 1, sin(90)
	…
n	

	indices
0	0
1	1
2	2
3	1
4	3
5	2
6	2
7	3
8	4
9	3
10	5
11	4
	…
m	

Cylinder Example with Normals



To define a normal for each vertex,
we need to add a **normal buffer**.

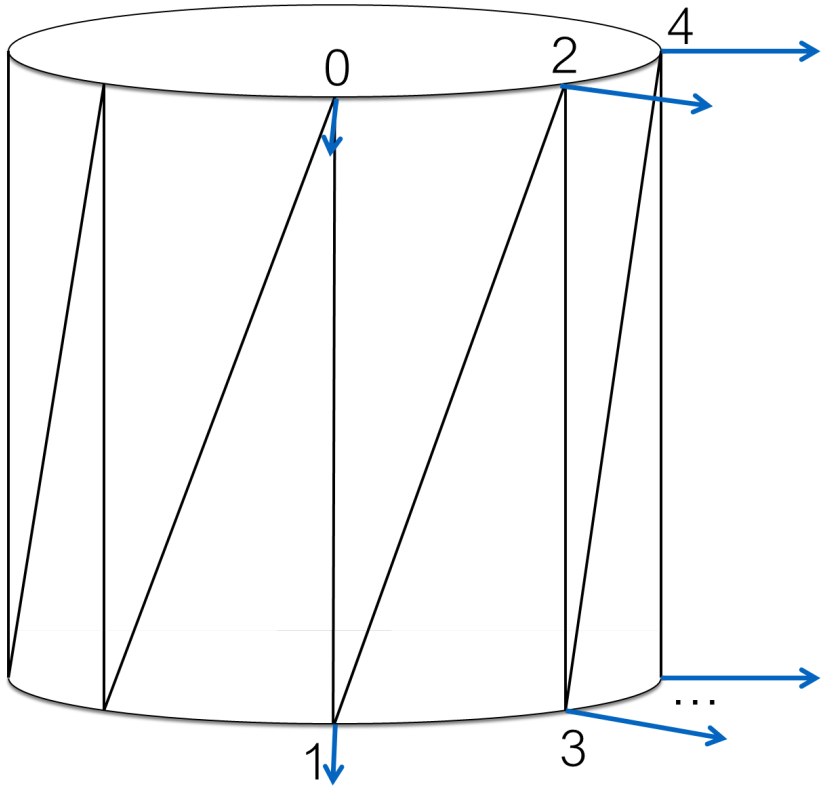
	vertices
0	cos(0), 1, sin(0)
1	cos(0), 0, sin(0)
2	cos(45), 1, sin(45)
3	cos(45), 0, sin(45)
4	cos(90), 1, sin(90)
	…
n	

	normals
0	cos(0), 0, sin(0)
1	cos(0), 0, sin(0)
2	cos(45), 0, sin(45)
3	cos(45), 0, sin(45)
4	cos(90), 0, sin(90)
	…
n	

	indices
0	0
1	1
2	2
3	1
4	3
5	2
6	2
7	3
8	4
9	3
10	5
11	4
	…
m	

This is the same size as the vertex buffer
and uses the same index buffer!

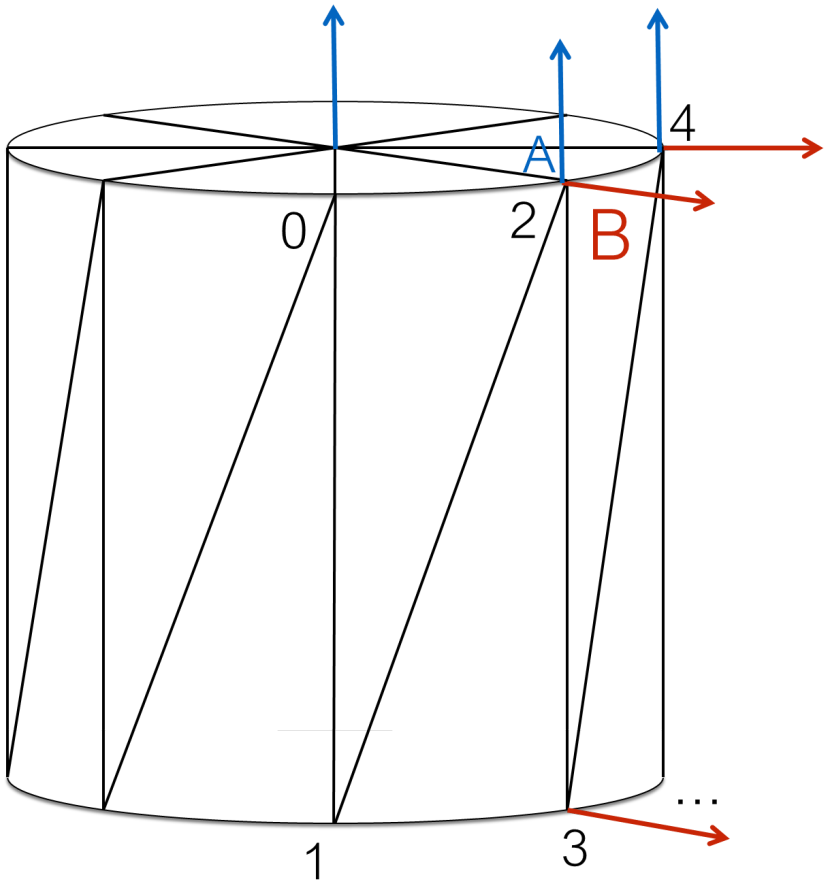
Cylinder Example with Normals



For the unit cylinder (radius of 1), the normal values are easy to calculate. They are the same x,y,z values as the vertices!

Hint: you should come back to this slide when you are trying to calculate the globe normals. :)

What about the top?



Triangles A and B both contain vertices 2 and 4, so we can save some space by just including these in the vertex table once, right?

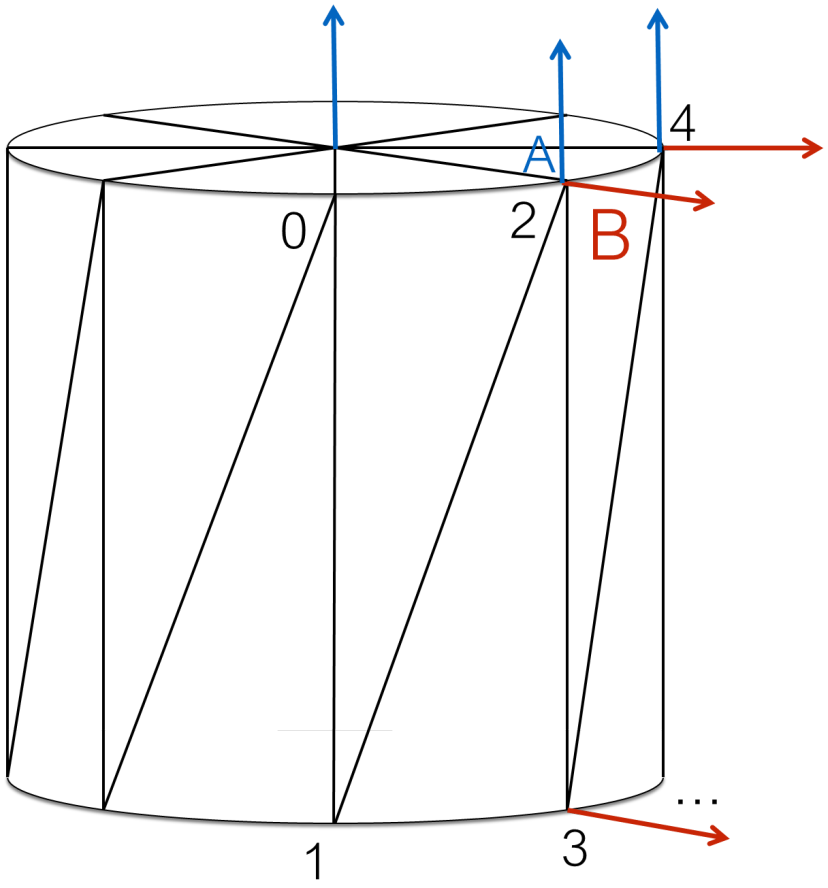
	vertices
0	cos(0), 1, sin(0)
1	cos(0), 0, sin(0)
2	cos(45), 1, sin(45)
3	cos(45), 0, sin(45)
4	cos(90), 1, sin(90)
	…
n	

	normals
0	cos(0), 0, sin(0)
1	cos(0), 0, sin(0)
2	cos(45), 0, sin(45)
3	cos(45), 0, sin(45)
4	cos(90), 0, sin(90)
	…
n	

	indices
0	0
1	1
2	2
3	1
4	3
5	2
6	2
7	3
8	4
9	3
10	5
11	4
	…
m	

No! The surface at these vertices is not continuous. There is a sharp edge, so the normal is actually different depending which triangle we are rendering!

What about the top?



Solution: Just make new vertices. They will have the same x,y,z position but different normals.

(You won't need to do this on Assignment 3.)

	vertices
0	cos(0), 1, sin(0)
1	cos(0), 0, sin(0)
2	cos(45), 1, sin(45)
3	cos(45), 0, sin(45)
4	cos(90), 1, sin(90)
	…
n	

	normals
0	cos(0), 0, sin(0)
1	cos(0), 0, sin(0)
2	cos(45), 0, sin(45)
3	cos(45), 0, sin(45)
4	cos(90), 0, sin(90)
	…
n	
n+1	0, 1, 0
n+2	0,-1, 0
n+3	0, 1, 0

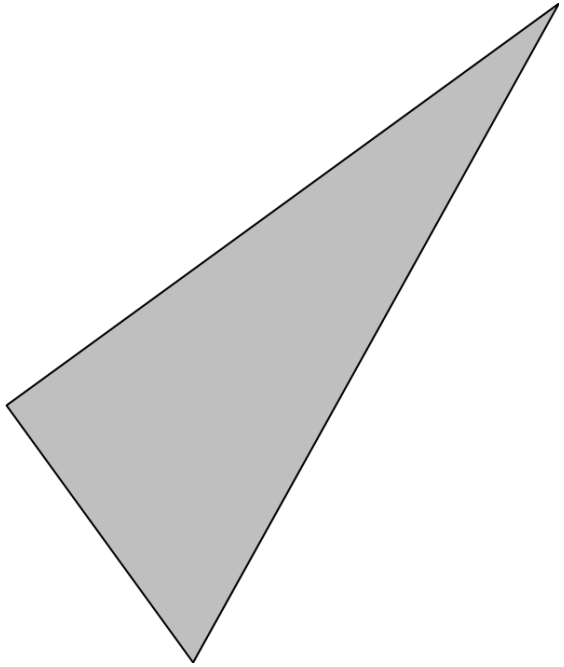
	indices
0	0
1	1
2	2
3	1
4	3
5	2
6	2
7	3
8	4
9	3
10	5
11	4
	…
m	
m+1	…
m+2	
m+3	

Calculating Triangle Normals

For a sphere or cylinder and other shapes where you know the equation for the geometry, you can usually also write an equation for the normals.

Sometimes, you are given a model with vertices but no normals (e.g., 3D scanning).

Given the position of the vertices, can you find the normal of a triangle?

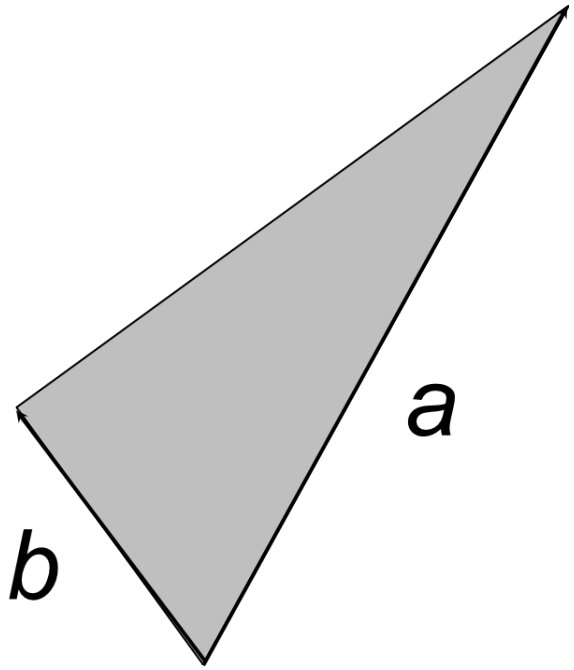


Calculating Triangle Normals

For a sphere or cylinder and other shapes where you know the equation for the geometry, you can usually also write an equation for the normals.

Sometimes, you are given a model with vertices but no normals (e.g., 3D scanning).

Given the position of the vertices, can you find the normal of a triangle?



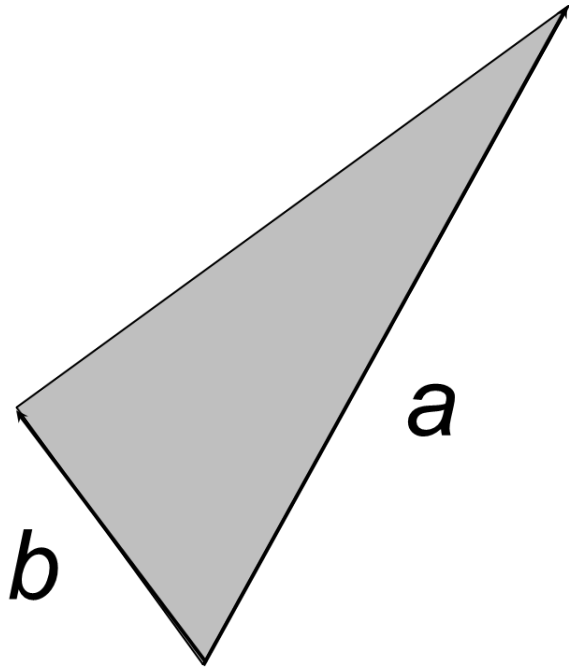
How do you calculate **n**?

Calculating Triangle Normals

For a sphere or cylinder and other shapes where you know the equation for the geometry, you can usually also write an equation for the normals.

Sometimes, you are given a model with vertices but no normals (e.g., 3D scanning).

Given the position of the vertices, can you find the normal of a triangle?



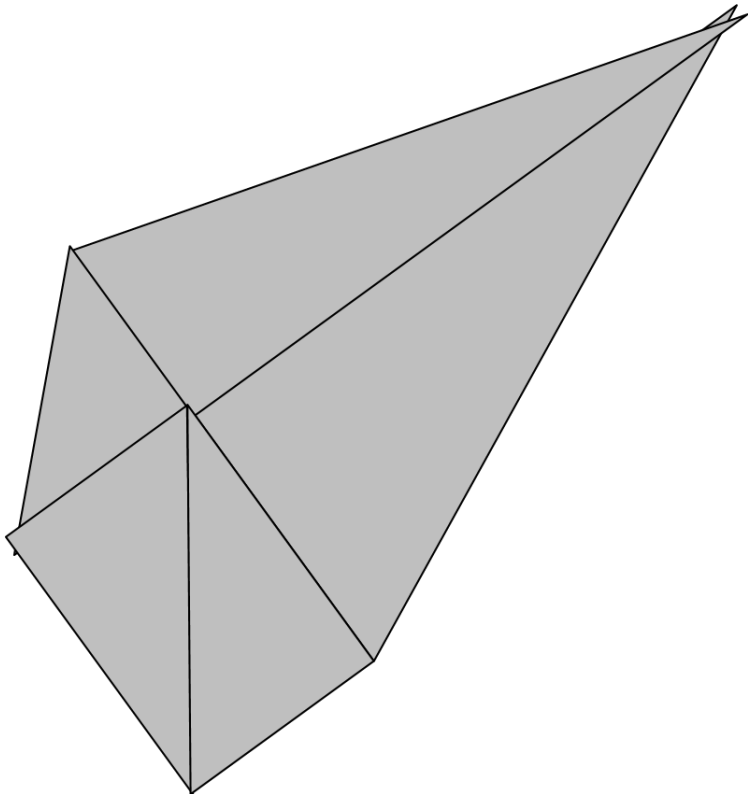
How do you calculate **n**?

n = normalize (**a** x **b**)

Calculating Vertex Normals

We just calculated the normal for the whole triangle, which is OK for "flat" surfaces.

What if you want a per-vertex normal, like we use for smooth surfaces?

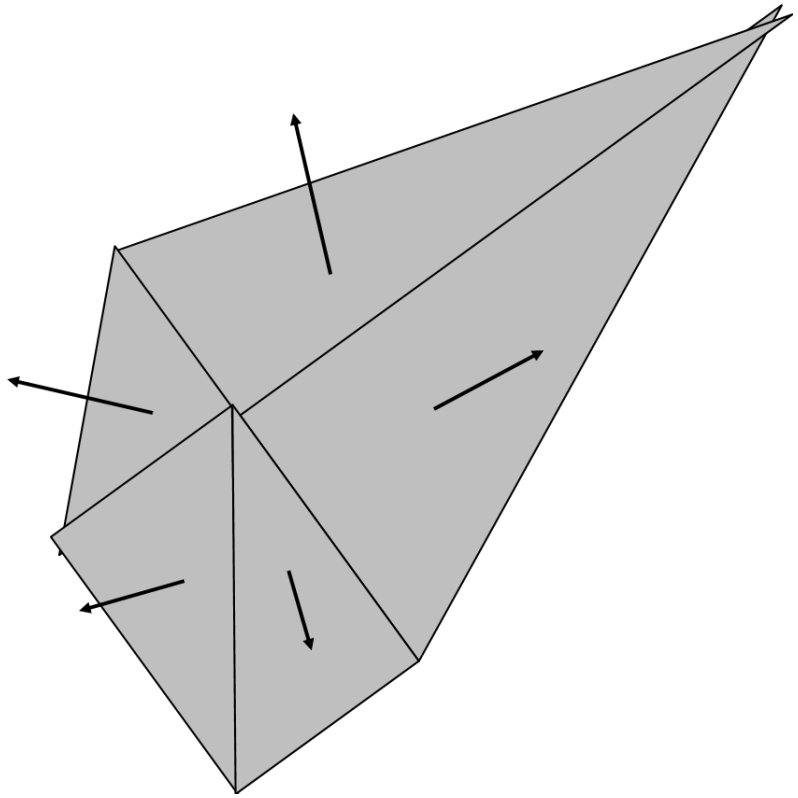


Calculating Vertex Normals

We just calculated the normal for the whole triangle, which is OK for "flat" surfaces.

What if you want a per-vertex normal, like we use for smooth surfaces?

First, find the normal for the neighboring triangles.



Calculating Vertex Normals

We just calculated the normal for the whole triangle, which is OK for "flat" surfaces.

What if you want a per-vertex normal, like we use for smooth surfaces?

First, find the normal for the neighboring triangles.

Then, average them to get the normal at the vertex. (Often a weighted average is used based on the area of each triangle.)