What if we want to modify our models?

We need a concept of "connectedness."
What about memory consumption?

Assume floating-point values (each component of a vec3) are stored as Float32 (4 bytes). How much memory is used if every triangle directly stores 3 vec3s?

How much memory is consumed by these 5 triangles using our previous representation?

- 180 bytes
- 60 bytes
- 45 bytes
- 80 bytes
- 128 bytes

Total: 300 bytes

Total: 180 bytes

Total: 160 bytes
Meshes have two ingredients: geometry & topology.

- Geometry: 3 coordinates per vertex, stored in a 1D array with a stride of 3.
- Topology: Each triangle is a flattened 1D array of indices.

```javascript
let vertices = [x0, y0, z0, x1, y1, z1, x2, y2, z2, x3, y3, z3];

let v2 = 2;
let dim = 1;
let y2 = vertices[3 * v2 + dim];

let triangles = [3, 2, 1, 3, 0, 2];

// access third vertex reference in triangle 1
let v3 = triangles[3 * 1 + 2];
```

Why??

- How we will pass stuff to the GPU.
- Also for efficiency.

Integers
Extracting the edges of a triangle mesh.

```javascript
for (let i = 0; i < nTriangles; i++) {
    for (let j = 0; j < 3; j++) {
        let p = triangles[3 * i + j];
        let q = triangles[3 * i + (j + 1) % 3];
        let edge = [Math.min(p, q), Math.max(p, q)];
        let edgeKey = JSON.stringify(edge);

        if (edgeKey in edgeMap) {
            // this edge already exists, no need to add it to list of edges
        } else {
            // edge does not exist yet, so add to map and to list of edges
            edgeMap[edgeKey] = edges.length / 2;
            edges.push(p, q);
        }
    }
}
```
Extracting the edges of a triangle mesh.

0 - 2 - 3
2 - 3
3 - 0

edges:
1 - 2
2 - 6
1 - 3
3 - 0
3 - 2

```javascript
for (let i = 0; i < nTriangles; i++) {
  for (let j = 0; j < 3; j++) {
    let p = triangles[3 * i + j];
    let q = triangles[3 * i + (j + 1) % 3];
    let edge = [Math.min(p, q), Math.max(p, q)];
    let edgeKey = JSON.stringify(edge);
    if (edgeKey in edgeMap) {
      // this edge already exists, no need to add it to list of edges
    } else {
      // edge does not exist yet, so add to map and to list of edges
      edgeMap[edgeKey] = edges.length / 2;
      edges.push(p, q);
    }
  }
}
```

edges = [3, 2, 2, 1, 1, 3, 0, 2, 3, 0]
Getting the "opposite" vertex of an edge.

```javascript
for (let i = 0; i < nTriangles; i++) {
    for (let j = 0; j < 3; j++) {
        let p = triangles[3 * i + j];
        let q = triangles[3 * i + (j + 1) % 3];
        let opposite = triangles[3 * i + (j + 2) % 3];
        let edge = [Math.min(p, q), Math.max(p, q)];
        let edgeKey = JSON.stringify(edge);
        if (edgeKey in edgeMap) {
            // this edge already exists, no need to add it to list of edges
        } else {
            // edge does not exist yet, so add to map and to list of edges
            edgeMap[edgeKey] = edges.length / 2;
            edges.push(p, q);
        }
    }
}
```
What are the indices of the four new triangles resulting from the subdivision?

1. $t_0 - e_0 - e_2$
2. $t_1 - e_1 - e_0$
3. $t_2 - e_2 - e_1$
4. $e_0 - e_1 - e_2$