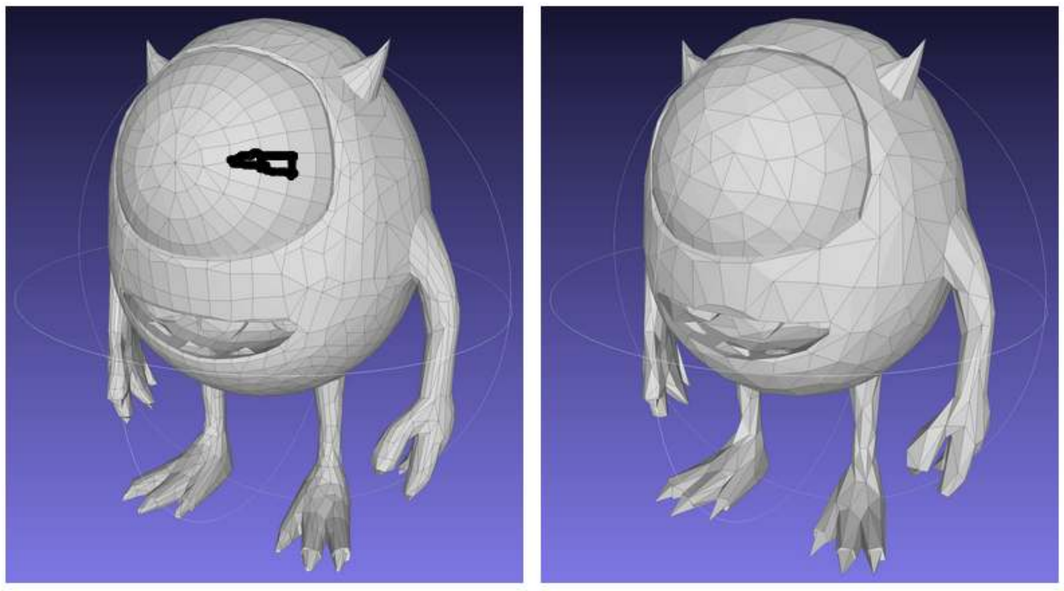
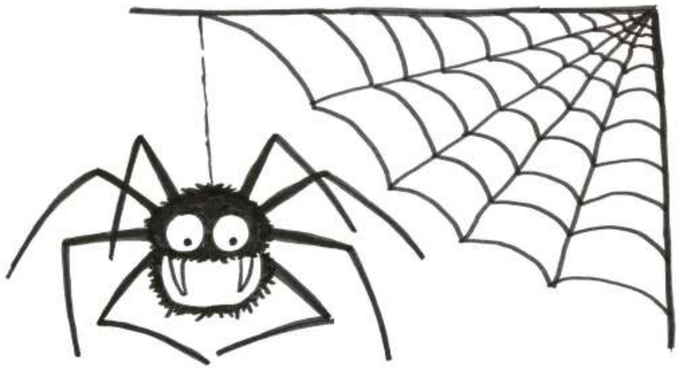
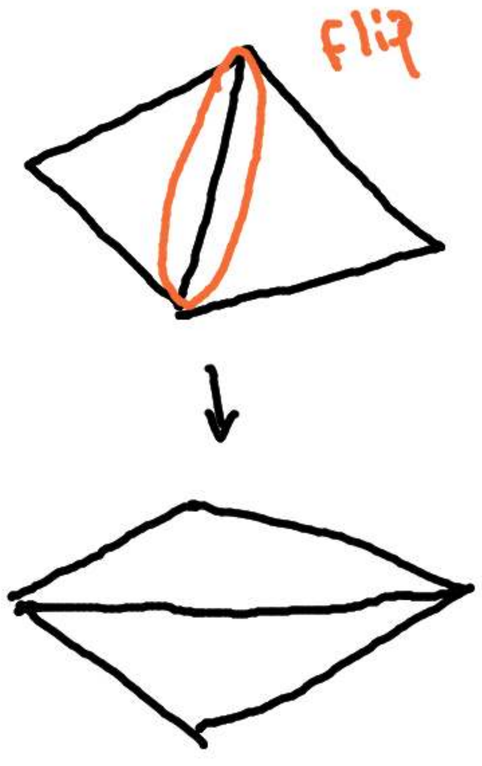
What if we want to modify our models?



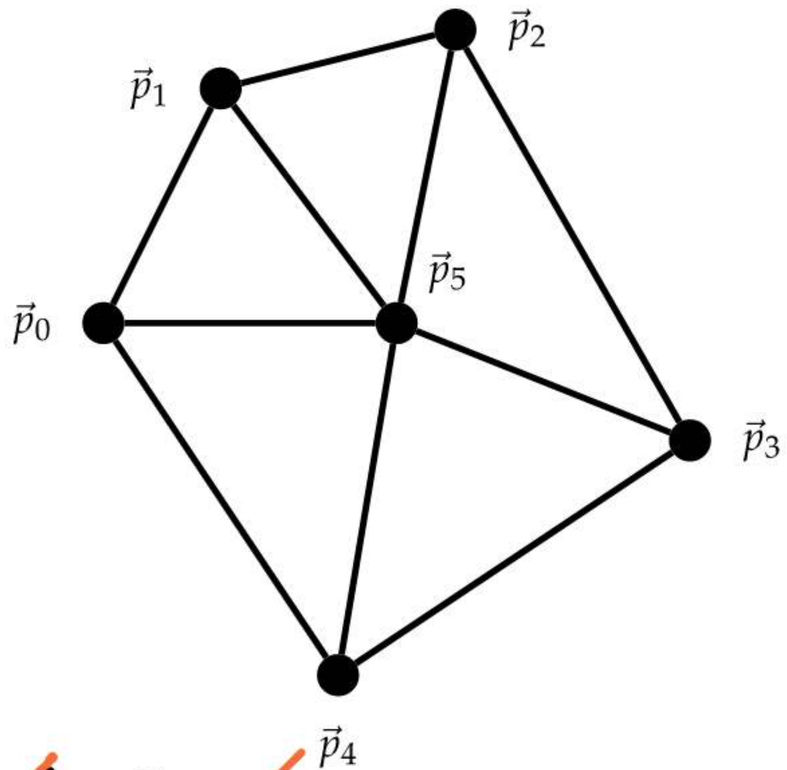




We need a concept of "connectedness."

What about memory consumption?

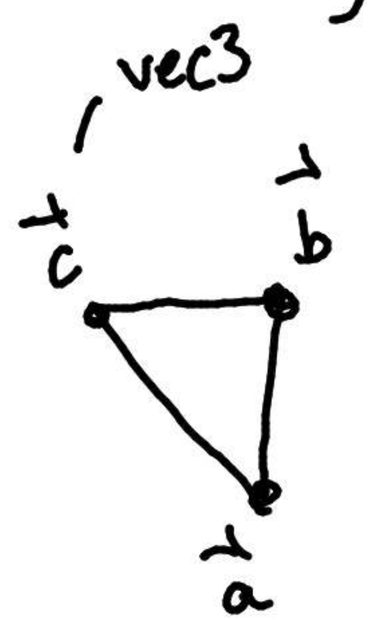
Assume floating-point values (each component of a `vec3`) are stored as `Float32` (4 bytes). How much memory is used if every triangle directly stores 3 `vec3`s?



How much memory is consumed by these 5 triangles using our previous representation? 24

- 180 bytes — 67%
- 60 bytes — 25%
- 45 bytes — 8%
- 80 bytes — 0%
- 128 bytes — 0%

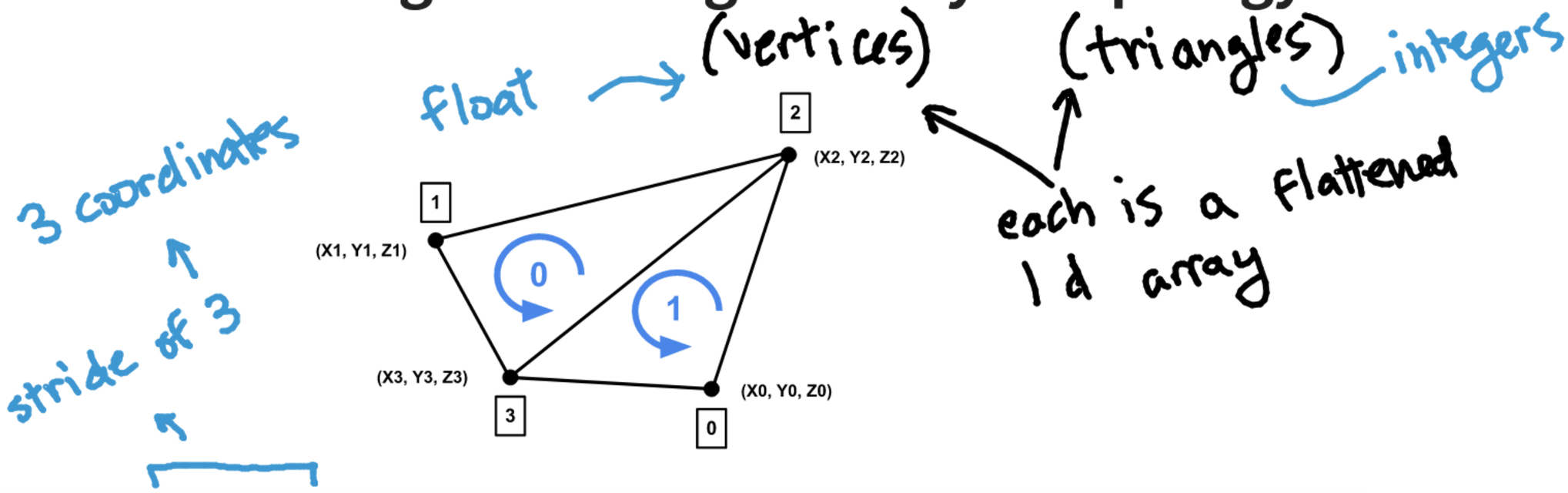
Edit response



$5\ \text{tri} \times 3\ \frac{\text{vec3}}{\text{tri}} \times 3\ \frac{\text{float}}{\text{vec3}} \times 4\ \frac{\text{bytes}}{\text{float}} = 180\ \text{bytes}$

$3 \times 5 \times 3 \times 4 = 180$

Meshes have two ingredients: geometry & topology.




```
1  let vertices = [x0, y0, z0, x1, y1, z1, x2, y2, z2, x3, y3, z2];
2
3  let v2 = 2;
4  let dim = 1;
5  let y2 = vertices[3 * v2 + dim];
6
7  let triangles = [3, 2, 1, 3, 0, 2];
8
9  // access third vertex reference in triangle 1
10 let v3 = triangles[3 * 1 + 2];
```

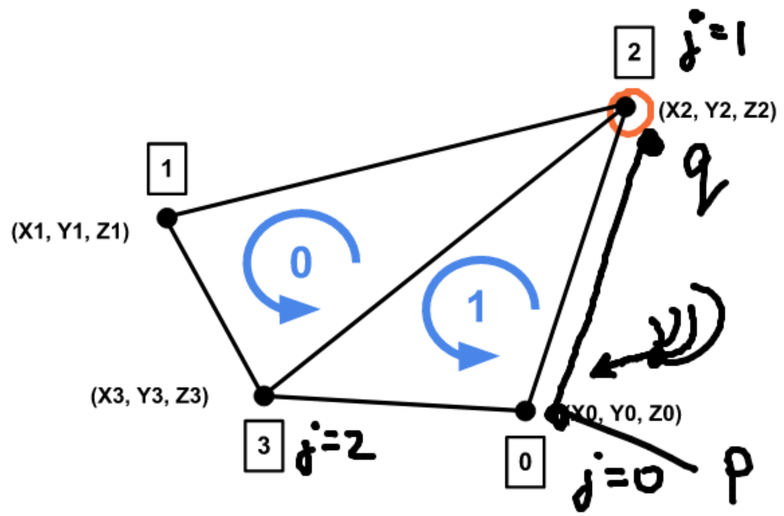
stride of 3

why?? how we will pass stuff to the GPU also for efficiency.

Extracting the edges of a triangle mesh.

0 - 2 - 3

0-2
2-3
3-0



edges:

1-2
2-6
1-3
3-0
3-2

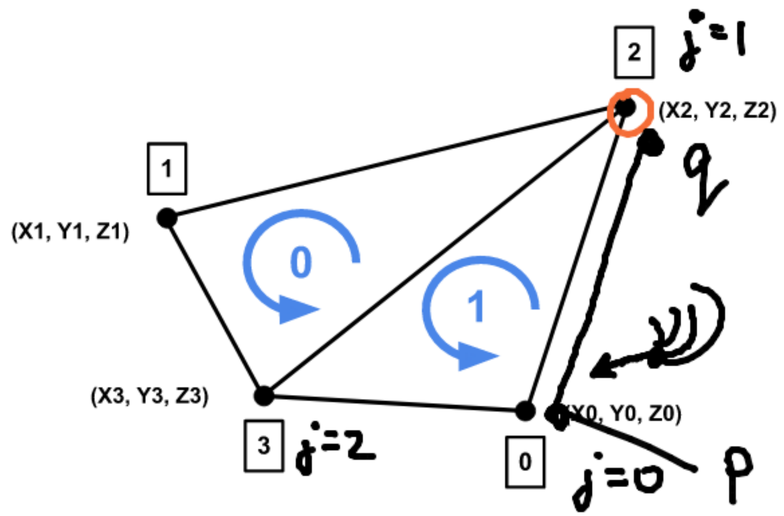
```
for (let i = 0; i < nTriangles; i++) {
    for (let j = 0; j < 3; j++) {
        let p = triangles[3 * i + j];
        let q = triangles[3 * i + (j + 1) % 3];
        let edge = [Math.min(p, q), Math.max(p, q);
        let edgeKey = JSON.stringify(edge);

        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

0-2
2-3
3-0



edges:

1-2
2-6
1-3
3-0
3-2

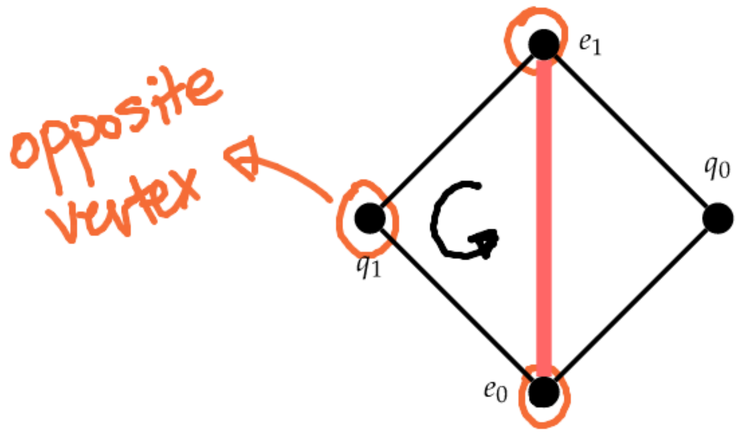
```
for (let i = 0; i < nTriangles; i++) {
    for (let j = 0; j < 3; j++) {
        let p = triangles[3 * i + j];
        let q = triangles[3 * i + (j + 1) % 3];
        let edge = [Math.min(p, q), Math.max(p, q);
        let edgeKey = JSON.stringify(edge);

        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 currently added

edges = [3, 2, 2, 1, 1, 3, 0, 2, 3, 0]

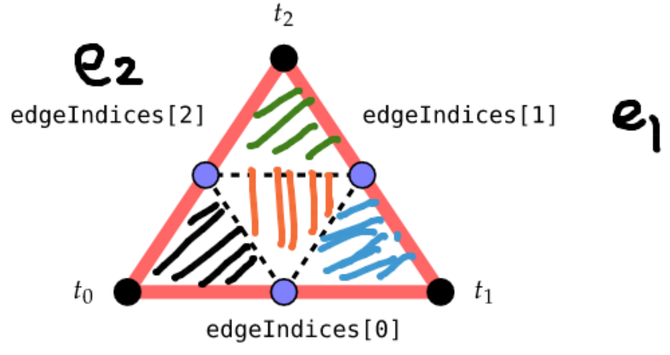
Getting the "opposite" vertex of an edge.




```
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?



what are the indices of 4 new triangles?

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