Creating 3D Shapes

A simple shape to start working with is a cube. Although a tetrahedron has fewer sides, its sides aren't orthogonal, which makes things trickier. Let's start by creating a 200×200×200 pixel cube, centred at the origin (0, 0, 0).

We will start not with actually drawing anything, but with coming up with arrays of numbers that describe our shapes in 3D shape - specifically, arrays that describe our **nodes** and our **edges**.

Nodes

We start by defining an array of nodes, where each node is an array of three digits, the x, y and z coordinates of that node:

```
var node0 = [-100, -100, -100];
var node1 = [-100, -100, 100];
var node2 = [-100, 100, -100];
var node3 = [-100, 100, 100];
var node4 = [ 100, -100, -100];
var node5 = [ 100, -100, 100];
var node6 = [ 100, 100, -100];
var node7 = [ 100, 100, 100];
var nodes = [node0, node1, node2, node3, node4, node5, node6, node7];
```

As you may have noticed, the nodes are all 8 ways of arranging three lots of positive or negative 100.

You can see the nodes of a 2x2x2 cube centered at the origin in the visualization below. Rotate using the mouse:

Edges

Next we define an array of edges, where each edge is an array of two numbers. For example, `edge0` defines an edge between `node0` and `node1`. We start counting at 0 because arrays are indexed starting at zero (To get the value of the first node we type `nodes[0]`).

```
var edge0 = [0, 1];
var edge1 = [1, 3];
var edge2 = [3, 2];
var edge3 = [2, 0];
var edge4 = [4, 5];
var edge5 = [5, 7];
var edge6 = [7, 6];
var edge7 = [6, 4];
var edge8 = [0, 4];
var edge9 = [1, 5];
var edge10 = [2, 6];
var edge11 = [3, 7];
var edges = [edge0, edge1, edge2, edge3, edge4, edge5, edge6, edge7, edge8,
edge9, edge10, edge11];
```

The tricky part is making sure you join the right edges together. Here's a visualization of the edges we're connecting for a cube: