Drawing 3D Shapes

- So now that we have a representation of our cube, we need to find a way to draw it.
- To draw a 3D shape on a 2D surface, we need to "project" it. Imagine shining a light behind the cube and projecting its shadow onto a screen. The simplest form of projection is an orthogonal projection, which is what you would get if the beams of light were parallel to one another.
- All this talk of projections might sound complicated, but it is very easy to implement: we just ignore the z-coordinates when drawing.

Setting things up

- I like to create variables at the top of my programs to control how things are displayed, so I can easily change them later. Here are some variables we will want shortly; feel free to change the values.

```
var backgroundColor = color(255, 255, 255);
var nodeColor = color(40, 168, 107);
var edgeColor = color(34, 68, 204);
var nodeSize = 8;
```

- Now we add a basic draw function:

```
var draw = function() {
    background(backgroundColor);
};
```

- We will also need to add the following to our code:

```
translate(200, 200);
```

- This shifts our canvas 200 pixels right and 200 pixels down, so the pixel at position (0, 0), is now in the centre of our canvas. This means that our cube will be drawn in the center of the screen. The reason for doing things this way will become clear when we start to rotate our objects.

Drawing nodes

- Inside the draw function we loop through all the nodes and draw an ellipse at the (x, y) coordinate of that node:

```
fill(nodeColor);
noStroke();
for (var n = 0; n < nodes.length; n++) {
   var node = nodes[n];
   ellipse(node[0], node[1], nodeSize, nodeSize);
}
```

Drawing edges

- Also inside the draw function we add the code for drawing edges. I would add it before the code for drawing nodes, so the nodes are drawn on top of the edges.

```
stroke(edgeColor);
for (var e = 0; e < edges.length; e++) {
   var n0 = edges[e][0];
   var n1 = edges[e][1];
   var node0 = nodes[n0];
   var node1 = nodes[n1];
```

-     line(node0[0], node0[1], node1[0], node1[1]);
- }
- This code loops through the array of edges. It gets the two numbers defined by an edge and looks up the corresponding node in the nodes array. Then it draws a line from the (x, y) coordinate of the first node to the (x, y) coordinate of the second node.

Is that it?

- We set out to draw a cube, yet all we've done is draw a square and four circles:

```
var backgroundColour = color(255, 255, 255);
var nodeColour = color(40, 168, 107);
var edgeColour = color(34, 68, 204);
var nodeSize = 8;

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];
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];

var draw= function() {
    background(backgroundColour);

    // Draw edges
    stroke(edgeColour);
    for (var e = 0; e < edges.length; e++) {
        var n0 = edges[e][0];
        var n1 = edges[e][1];
        var node0 = nodes[n0];
        var node1 = nodes[n1];
        line(node0[0], node0[1], node1[0], node1[1]);
    }

    // Draw nodes
    fill(nodeColour);
    noStroke();
    for (var n = 0; n < nodes.length; n++) {
        var node = nodes[n];
        ellipse(node[0], node[1], nodeSize, nodeSize);
    }
};
```

```
translate(200, 200);
```

We could have drawn that with a lot less effort. However, this really is our cube - it's just that we're viewing it end on. If we can work out how to rotate our cube so it's no longer end on to the screen, we'll see that it's not just a square and four circles.