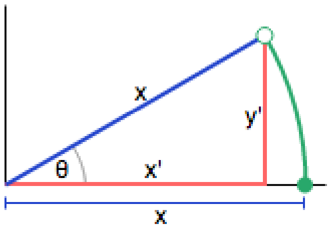
Rotating 3D Shapes

Rotating things in three dimensions sounds complicated and it can be, but there are some simple rotations. For example, if we imagine rotating our cube around the z-axis (which points out of the screen), we are actually just rotating a square in two dimensions:

A reason to learn trigonometry



We can simplify things further, by just looking at a single node at position (x, 0). Using simple trigonometry we can find that the position of the point after rotating it by θ around the origin is:

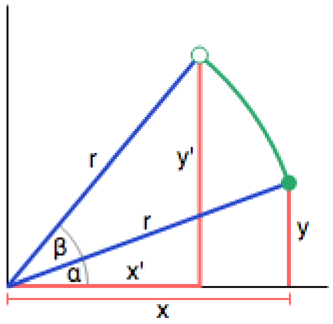
https://www.khanacademy.org/math/trigonometry/less-basic-trigonometry/angle-addition-formula-proofs/v/proof-angle-addition-sine

`x′=x×cos(θ)x' = x \times cos(\theta)x′=x×cos(θ)`

`y′=x×sin(θ)y' = x \times sin(\theta)y′=x×sin(θ)`

If you don't understand where these equations came from, this video might help.

Rotating a point about the origin



The example above allows us to rotate a point that starts on the x-axis about the origin, but what if it isn't on the x-axis? This requires some slightly more advanced trigonometry. If we call the distance between the point (x, y) and the origin `rrr`, and the angle between the line to (x, y) and x-axis α, then:

$x=r\times cos(\alpha)y=r\times sin(\alpha)$

If we rotate by β to point (x', y'), then:

$x'=r\times cos(\alpha+\beta)y'=r\times sin(\alpha+\beta)$

Using some trigonometric identities, we get:

$x'=r\times cos(\alpha)cos(\beta)-r\times sin(\alpha)sin(\beta)y'=r\times sin(\alpha)cos(\beta)+r\times cos(\alpha)sin(\beta)$

Substituting in the values for x and y above, we get an equation for the new coordinates as a function of the old coordinates and the angle of rotation:

$x'=x\times cos(\beta)-y\times sin(\beta)y'=y\times cos(\beta)+x\times sin(\beta)$

Writing a rotate function

Now we know the mathematics, we can write a function to rotate a node, or even better, our array of nodes, around the z-axis. This function will loop through all the nodes in the node array, find its current x and y coordinates and then update them. We store `sin(theta)` and `cos(theta)` outside the loop so we only need to calculate them once:

```
var rotateZ3D = function(theta) {
   var sinTheta = sin(theta);
   var cosTheta = cos(theta);
   for (var n = 0; n < nodes.length; n++) {
      var node = nodes[n];
      var x = node[0];
      var y = node[1];
      node[0] = x * cosTheta - y * sinTheta;
      node[1] = y * cosTheta + x * sinTheta;
   }
};
```

To rotate the cube by 30 degrees, we'll call the function like this:

```
rotateZ3D(30);
```

You can see the rotated cube below - it's slightly more interesting than before, but not by much:

```
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];


// Rotate shape around the z-axis

var rotateZ3D = function(theta) {

   var sinTheta = sin(theta);

   var cosTheta = cos(theta);


   for (var n=0; n<nodes.length; n++) {

      var node = nodes[n];

      var x = node[0];

      var y = node[1];

      node[0] = x * cosTheta - y * sinTheta;

      node[1] = y * cosTheta + x * sinTheta;

   }

};


rotateZ3D(30);


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);
```

Rotating in three dimensions

We can now rotate our cube in two dimensions, but it still looks like a square. What if we want to rotate our cube around the y-axis (vertical axis)? If we imagine looking down on our cube as we rotate it around the y-axis, what we would see is a rotating square, just like we do when we rotate about the z-axis.

We can take our trigonometry and function from before, and just re-label the axis so that the z-axis becomes the y-axis. In this case, the y-coordinates of the node do not change, only the x and the z:

```
var rotateY3D = function(theta) {
   var sinTheta = sin(theta);
   var cosTheta = cos(theta);
   for (var n = 0; n < nodes.length; n++) {
      var node = nodes[n];
      var x = node[0];
      var z = node[2];
      node[0] = x * cosTheta - z * sinTheta;
      node[2] = z * cosTheta + x * sinTheta;
   }
};
```

And we can use the same argument to create a function that rotates our cube around the x-axis:

```
var rotateX3D = function(theta) {
   var sinTheta = sin(theta);
   var cosTheta = cos(theta);
   for (var n = 0; n < nodes.length; n++) {
      var node = nodes[n];
      var y = node[1];
      var z = node[2];
      node[1] = y * cosTheta - z * sinTheta;
      node[2] = z * cosTheta + y * sinTheta;
   }
};
```

Now that we have those functions defined, we can rotate 30 degrees by the two other axis:

```
rotateX3D(30);
rotateY3D(30);
```

You can see the complete code below. Try using the number scrubber to change the values in the function calls.

```
 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];

// Rotate shape around the z-axis
var rotateZ3D = function(theta) {
    var sinTheta = sin(theta);
    var cosTheta = cos(theta);

    for (var n=0; n<nodes.length; n++) {
        var node = nodes[n];
        var x = node[0];
        var y = node[1];
        node[0] = x * cosTheta - y * sinTheta;
        node[1] = y * cosTheta + x * sinTheta;
    }
};

var rotateY3D = function(theta) {
    var sinTheta = sin(theta);
    var cosTheta = cos(theta);
    for (var n=0; n<nodes.length; n++) {
        var node = nodes[n];
        var x = node[0];
        var z = node[2];
        node[0] = x * cosTheta - z * sinTheta;
        node[2] = z * cosTheta + x * sinTheta;
    }
};

var rotateX3D = function(theta) {
    var sinTheta = sin(theta);
    var cosTheta = cos(theta);

    for (var n=0; n<nodes.length; n++) {
        var node = nodes[n];
        var y = node[1];
        var z = node[2];
        node[1] = y * cosTheta - z * sinTheta;
        node[2] = z * cosTheta + y * sinTheta;
    }
};

rotateZ3D(30);
rotateY3D(30);
rotateX3D(30);

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);
```

User interaction

We can rotate the cube by adding function calls, but it's a lot more useful (and satisfying) if we can enable the viewer to rotate the cube using their mouse. For this we need to create a `mouseDragged()` function. This function is automatically called whenever the mouse is dragged.

```
mouseDragged = function() {
   rotateY3D(mouseX - pmouseX);
   rotateX3D(mouseY - pmouseY);
};
```

`mouseX` and `mouseY` are built-in variables that contain the current position of the mouse. `pmouseX` and `pmouseY` are built-in variables that contain the position of the mouse in the previous frame. So if the x-coordinate has increased (we move the mouse right), we send a postive value to `rotateY3D()` and rotate the cube counter-clockwise around the y-axis.

You can see for yourself below.

```
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];

// Rotate shape around the z-axis
```

```
var rotateZ3D = function(theta) {
   var sinTheta = sin(theta);
   var cosTheta = cos(theta);

   for (var n=0; n<nodes.length; n++) {
      var node = nodes[n];
      var x = node[0];
      var y = node[1];
      node[0] = x * cosTheta - y * sinTheta;
      node[1] = y * cosTheta + x * sinTheta;
   }
};

var rotateY3D = function(theta) {
   var sinTheta = sin(theta);
   var cosTheta = cos(theta);

   for (var n=0; n<nodes.length; n++) {
      var node = nodes[n];
      var x = node[0];
      var z = node[2];
      node[0] = x * cosTheta - z * sinTheta;
      node[2] = z * cosTheta + x * sinTheta;
   }
};

var rotateX3D = function(theta) {
   var sinTheta = sin(theta);
   var cosTheta = cos(theta);
```

```
    for (var n=0; n<nodes.length; n++) {
        var node = nodes[n];
        var y = node[1];
        var z = node[2];
        node[1] = y * cosTheta - z * sinTheta;
        node[2] = z * cosTheta + y * sinTheta;
    }
};
rotateZ3D(30);
rotateY3D(30);
rotateX3D(30);

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);
    }


};


mouseDragged = function() {
    rotateY3D(mouseX - pmouseX);
    rotateX3D(mouseY - pmouseY);
};


translate(200, 200);
```