

OPENJSCAD (a web based variant of openScad)

<https://en.wikipedia.org/wiki/OpenSCAD>

<http://openjscad.org/>

<http://joostn.github.io/OpenJsCad/>

https://en.wikibooks.org/wiki/OpenJSCAD_User_Guide

<https://github.com/Spiritdude/OpenJSCAD.org>

web jsCad

notes

details

misc

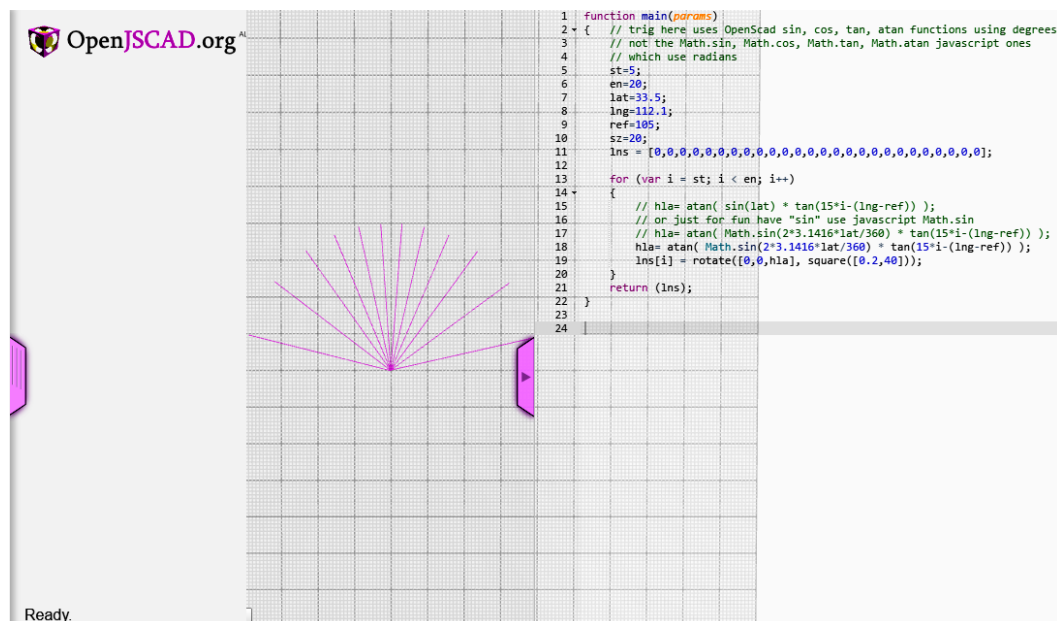
2D shapes ~ Two dimensional shapes can be defined through the CAG class. CAG stands for 'Constructive Area Geometry' which is the 2D variant of CSG.

shift+enter to redisplay

VERY BASIC H-DIAL (using: <http://openjscad.org/>)

```
function main(params)
{
  // trig here uses OpenScad sin, cos, tan, atan functions using degrees
  // not the Math.sin, Math.cos, Math.tan, Math.atan javascript ones
  // which use radians
  st=5;
  en=20;
  lat=33.5;
  lng=112.1;
  ref=105;
  sz=20;
  lns = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];

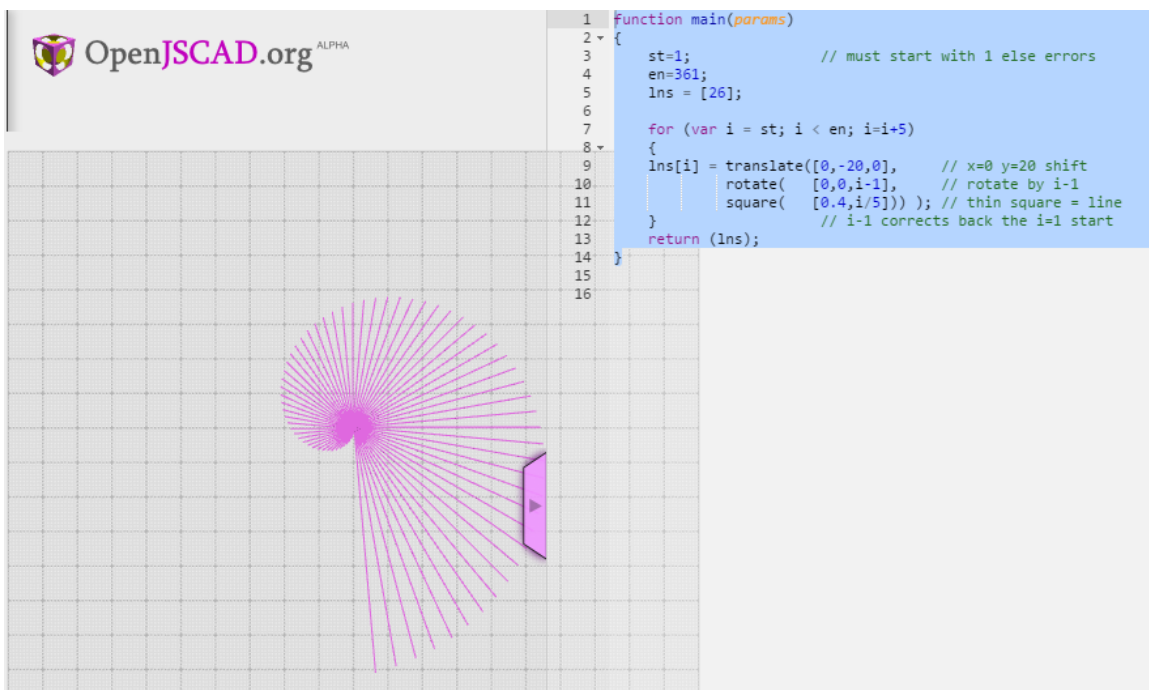
  for (var i = st; i < en; i++)
  {
    // hla= atan( sin(lat) * tan(15*i-(lng-ref)) );
    // or just for fun have "sin" use javascript Math.sin
    // hla= atan( Math.sin(2*3.1416*lat/360) * tan(15*i-(lng-ref)) );
    hla= atan( Math.sin(2*3.1416*lat/360) * tan(15*i-(lng-ref)) );
    lns[i] = rotate([0,0,hla], square([0.2,40]));
  }
  return (lns);
}
```



A NAUTILAS (using: <http://openjscad.org/>)

```
function main(params)
{
  st=1; // must start with 1 else errors
  en=361;
  lns = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];

  for (var i = st; i < en; i=i+5)
  {
    lns[i] = translate([0,-20,0], // x=0 y=20 shift
      rotate( [0,0,i-1], // rotate by i-1
        square( [0.4,i/5])) ); // thin square = line
  }
  return (lns);
}
```



A NAUTILAS AND PARAMETERS

(using: <http://openjscad.org/>)

```
function getParameterDefinitions() {
  return [{ name: 'stp', type: 'float', initial: 1, caption: "Start:" },
          { name: 'enp', type: 'float', initial: 361, caption: "End:" } ];
}
// above code enables parameter box on screen

function main(params)
{
  st = params.stp;
  en = params.enp;
  lns = []; // no elements predefined, so array could have voids in it

  for (var i = st; i <= en; i=i+5)
  {
    lns[i] = translate([0,-20,0], // x=0 y=20 shift
                      rotate( [0,0,i-1], // rotate by i-1
                              square( [0.4,i/5])) ); // thin square = line
  }
  // i-1 corrects back the i=1 start
  return (lns);
}
```

The screenshot displays the OpenJSCAD.org web interface. On the left, a 3D model of a nautilus shell is rendered in a light purple color on a grid background. On the right, a code editor shows the JavaScript code for the model. Below the grid, a parameter control panel is visible, featuring two input fields: 'Start: 90' and 'End: 271'. Below these fields are an 'Update' button and a checked checkbox labeled 'Instant Update'. The code editor on the right shows the same code as the text block above, with line numbers 1 through 24 visible.

SLIGHTLY BETTER H-DIAL (using: <http://openjscad.org/>)

```
function getParameterDefinitions() {
  return [{name:'lat', type:'float', initial: 33.5, caption: "Latitude:" },
    {name:'lng', type:'float', initial: 108.1, caption: "Longitude:"},
    {name:'ref', type:'float', initial: 105, caption: "Ref.Long:" },
    {name:'st', type:'float', initial: 6, caption: "Start HR:" },
    {name:'en', type:'float', initial: 18, caption: "End HR:" }
  ];
}
// above code enables parameter box on screen

function main(params)
{
  // trig here uses OpenScad sin, cos, tan, atan functions using degrees
  // not the Math.sin, Math.cos, Math.tan, Math.atan javascript ones
  // which use radians
  st = params.st;
  en = params.en;
  lat = params.lat;
  lng = params.lng;
  ref = params.ref;
  sz=20;
  lns = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
  // for the array of lines to be returned for display

  for (var i = st; i <= en; i++)
  {
    // hla= atan( sin(lat) * tan(15*i-(lng-ref)) );
    // or just for fun have "sin" use javascript Math.sin
    // hla= atan( Math.sin(2*3.1416*lat/360) * tan(15*i-(lng-ref)) );
    // sin(degrees) and Math.sin(radians)
    hla= atan( Math.sin(2*3.1416*lat/360) * tan(15*i-(lng-ref)) );

    // put this line into the array
    lns[i] = rotate([0,0,hla], square([0.2,40]));
  }
  return (lns);
}
```

