**CADjs** ([www.CADjs.org](http://www.CADjs.org)); Version 2016.01  
cadjs.help@gmail.com

**3D Solids**

* **cube**(L,H,W);

g = cube(200,50,10);

g = cube(5);

g = cube();

* **cylinder**(R,H)

g = cylinder(50,200);

* **cone**(rBase,rTop,H)

g = cone(100,5,300);

* **sphere**(R)

g = sphere(100);

* **torus**(R,r)

g = torus(100,40);

* **textgeom**(string,H,D)

g = textgeom(‘John’)

**Booleans**

* **union(**geom**)**

g = cube(2).union(cylinder());

* **difference(**geom**)**

g = cube().difference(sphere(.6));

* **intersection(**geom**)**

g=cube().intersection(sphere(.6));

**Transformations,**

* **clone();** make a copy

g2 = g.clone();

* **translateX**(d), **translateY**(d), **translateZ**(d);

g = cube().translateX(20);

* **translate**(dx,dy,dz);

cube().translate(20,40,0);

* **rotateX**(deg), **rotateY**(deg), **rotateZ**(deg),

g = cube().rotateX(90);

* **scale**(sx,sy,sz);

g = cube().scale(0.5,1,1);

* **mirrorX**(), **mirrorY**(), **mirrorZ**()

g = cone().mirrorY();

**Display, Save, Info**

* **display** ()**, display**(color)

g = cube().display(‘blue’);

* **displayTransparent**(color)

cube().displayTransparent ();

For colors supported see [link](http://www.w3schools.com/html/html_colornames.asp):

**Info, Save**

* **info**(); //area, volume, etc

cube().info();

* **saveSTL**(); save geometry as STL

cube().saveSTL();

* **saveOFF**(); save geometry in OFF format

cube().saveOFF();

**Extrude, Revolve**

* **extrudeShape**(s,thickness);

pts = [[0,0],[1,0],[0,1]];

s = polygon(pts);

g = extrudeShape(s,0.5);

* **revolveShape**(s,radius);

pts = [[0,0],[1,0],[0,1]];

s = polygon(pts);

g = revolveShape (s,3.0);

* **revolveShapeBetween**(s,radius,startAngle,endAngle);

pts = [[0,0],[1,0],[0,1]];

s = polygon(pts);

g = revolveShapeBetween (s,3,0,90);

* **sweepShapeAlongPath**(s,path);

See examples

**2D Shapes**

* **polygon**(arrayOfPts)

pts = [[0,0],[1,0],[0,1]];

s = polygon(pts);

s.display();

* **circle**(radius)

s = circle(3.0);

s.display();

* **polyarc**(arrayOfPtsAndArcs)

arc = [1.0,0.1,0.9,0.1,false];

pts = [[0,0],[0.9,0],arc,[1,1],[0,1]];

s = polyarc(pts);

s.display();

* **Adding holes**

pts = [[0,0],[1,0],[1,1],[0,1]];

s = polygon(pts);

c = circle(0.1);

s.holes.push(c);

s.display();

**Functions**

* **function f**(h){}; see examples

**Splines, Bezier**

* **start2DGeom**(x,y)

s = start2DGeom(0,0);

* **addLine**(x,y);

s.addLine(1,1);

* **addArc(x,y,xc,yc,clockwise)**

s.addArc(1,1,0,1,false);

* **addSpline(<array of 4 pts>)**
* **addBezier(<array of 4 pts>)**

**If, Else**

* **if () {} else {};** see examples

**Math**

* **makeArray(start,end,step);**

t= makeArray(0,1,0.1);

* **math.poly(coeffArray,tArray);**

x= math.poly ([0,1,-1],t);

* **math.sin(tArray);**

x= math.sin ([0,1,-1],t);

* **b = scalarMultiply(array,a)**

x = scalarMultiply(x,2.0);

* **plot2d(xArray,yArray,clr);**
* **plot3d(xArray,yArray,zArray,clr);**

**Loops**

* **for** (i = 0; i < N;i++) {};

for (i = 0; i < 5; i= i+1){

beep();

}

* **while** () {}

See examples

**CADjs Examples**

|  |  |  |  |
| --- | --- | --- | --- |
| **Category** | **Figure** | **Example** | **Variations** |
| Basics |  | g = cube(100);  g.display(); | g = cube(200,60,20);  g.display(‘blue’); |
| Solids | http://www.sciartsoft.com/1_Cylinder.png | radius = 100;  height = 200;  g = cylinder(radius,height);  g.display(); | *g = torus(100,25);*  *g = cone(100,50)*  *g= sphere(100);* |
| Booleans |  | g1 = cube(1);  g2 = sphere(0.6);  g1 = g1.difference(g2);  g1.display();​ | *g1 = g1.union(g2);*  *g1=g1.intersection(g2);* |
| Axes |  | g = cube(1);  g.display();  axesoff(); | *g = cube(1);*  *g.display();*  *axeson();* |
| Text Geometry |  | g = textGeom("CADjs");  g.scale(2).rotateZ(45);  g.display(); |  |
| Move & Rotate |  | g = cube(1,0.25,0.25);  g1 = g.clone();// create copy  g1.rotateX(45);  g1.translateY(0.5);  g.display('red');  g1.display('blue'); | g1.translateY(0.5);  g1.rotateX(45);​  g1.translate(1,-1,0.2);  g1.displayWireFrame(); |
| Scaling |  | g1 = sphere(1,64);  g1.scale(1,0.75,0.75);  g1.display('purple');  g2 = cube(1.15);  g2.display('red'); |  |
| Concatenation of functions |  | g1 = cube(1,0.2,0.2).rotateX(90).translateY(0.5);  g = cube(0.2,1,0.5).union(g1);  g.display(); | |
| loops |  | g1 = cube(0.25,1,1);  g = g1.clone();  for (var i = 0;i < 9;i++) {    g1.rotateX(10);// 10 degrees each time    g = g.union(g1);  }  g.display(); | i = 0;  while (i < 10) {  ...  i++;  } |
| Geometry Info |  | g = cube(1);  g.info(); |  |
| functions |  | function insert(h) {  var g1, g2, g;  g1 = cylinder(0.2,h);  g2 = cylinder(0.1,h);  g = g1.difference(g2);  return g;  }  g1 = insert(0.2);  g1.display();  g2 = insert(0.1).translateY(0.5).display(); |  |
| extrude |  | s = polygon([ [0,0],[0,1],[1,0] ]);  s.display();  g = extrudeShape(s,0.25);//thickness  g.translateZ(1).display(); |  |
| extrude |  | arc =[0.4,0.5,0.5,0.5,true];  s = polyarc([ [0,0], [1,0],[1,0.4],[0.5,0.4],arc,[0.4,1],[0,1]]);  s.display();//display 2D shape  thickness = 0.1;  g = extrudeShape(s,thickness).translateZ(0.5);  g.display(); | |
| revolve |  | pts = [[0,0],[1,0],[1,1],[0,1]];  s = polygon(pts);  radius = 3;  g = revolveShape (s,radius);  g.display(); | |
| revolveShapeBetween |  | s = circle(1);  radius = 3;  g = revolveShapeBetween (s,radius,0,180);  g.display(); | |
| sweepShapeAlongPath |  | function helicalSpring( radius, rCrossSection,nTurns, pitch) {  var x,y,z,nSamples,theta,vec;  var path = [];  shape = circle(rCrossSection);// make a circular shape  nSamples = 10;//samples per turn  for (var i = 0; i <= nSamples\*nTurns; i++) {  theta = 2\*3.1415\*i/(nSamples);  x = radius\*Math.cos(theta);  y = radius\*Math.sin(theta);  z = pitch\*i/(nSamples);  vec = new THREE.Vector3( x, y, z );  path.push(vec);  }  var g = sweepShapeAlongPath(shape,path);  return g;  }  radius = 5; // radius of spring  rCrossSection = 0.5;  pitch = 4;// distance between consecutive turns  nTurns = 6;// Number of turns  spring = helicalSpring( radius, rCrossSection, nTurns, pitch);  spring.display(); | |
| sweepShapeAlongPath |  | function torsionalSpring( rStart, rEnd,rCrossSection, nTurns) {  var x,y,z,nSamples,theta,vec;  var path = [];  shape = circle(rCrossSection);// make a circular shape  nSamples = 20;//samples per turn  for (var i = 0; i <= nSamples\*nTurns; i++) {  theta = 2\*3.1415\*i/(nSamples);  radius = rStart + (rEnd-rStart)\*i/(nSamples\*nTurns);  x = radius\*Math.cos(theta);  y = radius\*Math.sin(theta);  vec = new THREE.Vector3( x, y, 0 );  path.push(vec);  }  var g = sweepShapeAlongPath(shape,path);  return g;  }  rStart = 5; //  rEnd = 1;//  rCrossSection = 0.25;  nTurns = 3;// Number of turns  spring = torsionalSpring( rStart, rEnd,rCrossSection, nTurns);  spring.display(); | |
| math | http://www.sciartsoft.com/Cosine.png | for (i = 0; i < 36;i++) {     height = Math.cos(i\*10\*3.14/180)\*50+60;     g = cylinder(5,height );     g.translateX(10\*i).translateY(height /2);     if (i % 2 == 0) // even       g.display('red');     else       g.display('blue');  } | |
| plot2d & math |  | x = makeArray(0,2\*3.1415,0.1);  y = Math.sin(x);  plot2d(x,y); | |
| plot2d & math |  | t = makeArray(-1,1,0.1);  x = Math.poly([0,1],t);  y = Math.poly([0,0,1,-1],t);  plot2D(x,y,'red'); | |
| spline |  | s = start2DGeom(0,0);  s.addLine(1,0);  s.addSpline([[1,1],[0.75,1.25],[0.25,0.75],[0,1.0]]);//4 points to create cubic spline  s.addLine(0,0);  s.display();  g = extrudeShape(s,1).translateZ (1);  g.display(); | |

**polygon example**

The shape below consists of 4 straight lines. To draw the sketch, we simply list the points in the anti-clockwise direction, starting from any point.

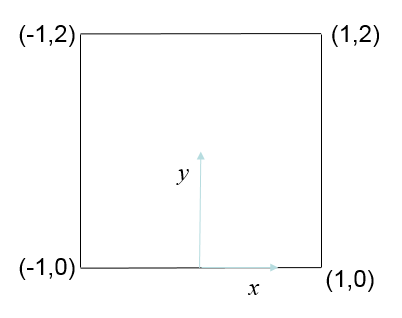
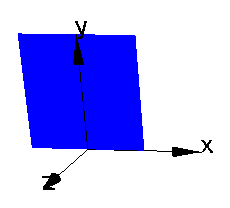
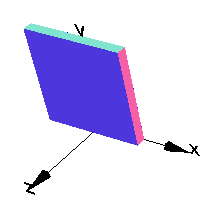
s = polygon([ [1,0], [1,2],[-1,2] ,[-1,0] ]);

s.display();

This will result in the shape shown in the middle. You can now extrude the sketch via:

g = extrudeShape(s,0.2).display();

This will result in the geometry shown at the right.

**polyarc example-1**

The shape shown on the left consists of 3 straight lines, and 1 arc. Observe that the arc goes from start point (1,2) to end point (-1,2) (anti-clockwise direction), with the center located at (0,2) with a radius of 1. To draw the arc, we don’t need to worry about the start point or the radius. Instead we specify the end point, center point and direction, and create the sketch as follows.

arc = [-1,2,0,2,false]; // first the end point, then the center point and finally the direction.

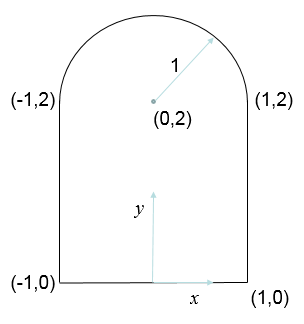
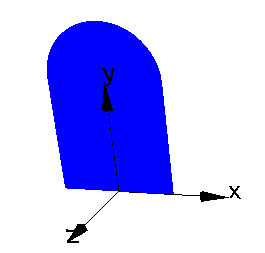
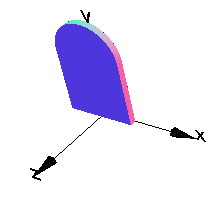
s = polyarc([ [1,0], [1,2],arc,[-1,0] ]);

s.display();

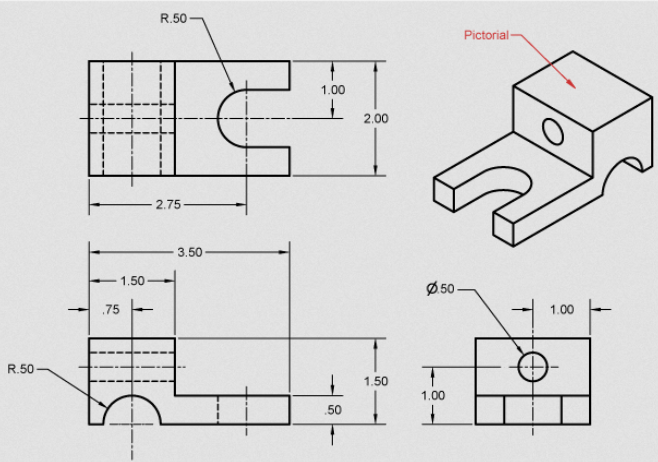
You can now extrude the sketch via:

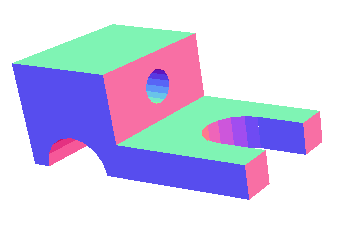
g = extrudeShape(s,0.2).display();

This will result in the geometry shown at the right.

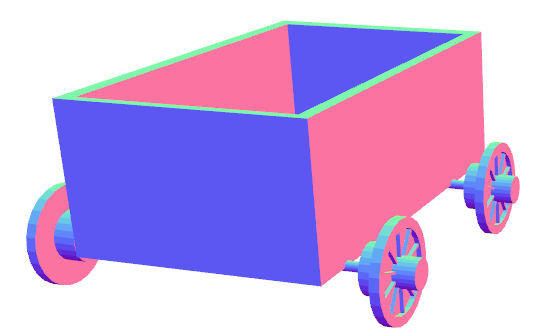
**Fixture**





|  |  |
| --- | --- |
| **Method 1 using Booleans of cubes and cylinders** | **Method 2 using Sketch** |
| base = cube(3.5,1.5,2);  g1 = cube(2,1,2);  hump = cylinder(0.5,2.5).rotateX(90).translateX(-1).translateY(-0.75);  cut1 = cylinder(0.5,2).translateX(1);  cut2 = cube(1,0.5,1).translateX(1.5).translateY(-0.5);  hole = cylinder(.25,2).rotateZ(90).translate(-1,0.25);  g1.translate(0.75,0.25,0);  base = base.difference(g1).difference(hump).difference(cut1);  base = base.difference(cut2).difference(hole);  base.display(); | arc = [1.25,0,0.75,0,true];  s=polyarc([[0,0],[0.25,0],arc,[3.5,0],[3.5,0.5],[1.5,0.5],[1.5,1.5],[0,1.5]]);  g = extrudeShape(s,2.0);  cut1 = cylinder(0.25,2).rotateZ(90).translate(0.75,1,1);  g = g.difference(cut1);  g.display(); |

**Wagon**



function wheelFunction()

{

//always declare variables, else they will be

//treated as global variables, messing up code

var g1, g2, g3,g4,g,gs,gsr;//always declare

g1 = cylinder(10,2);//main

g2 = cylinder(3,5).translateY(2);// outside hub

g3 = cylinder(5,5).translateY(-3); //inside hub

g4 = cylinder(8,2).translateY(1);//cutout

g = g1.difference(g4).union(g2).union(g3);

gs =cube(9,1.5,0.5).translate(4.5);// spoke

for (i = 0; i < 10;i++) {

gsr = gs.rotateY(36);

g = g.union(gsr);

}

g = g.rotateZ(-90);

return g;

}

var wheels = [];// create an empty array

wheels[0] = wheelFunction().translateX(35);

// use clone instead of calling function again

wheels[1] = wheels[0].clone().translateZ(50);

// mirroring same as rotate about Z

wheels[2] = wheels[0].clone().rotateZ(180);

wheels[3] = wheels[1].clone().rotateZ(180);

for (i = 0; i < 4; i++)

wheels[i].display();

axle1 = cylinder(1,70).rotateZ(90);

axle2 = axle1.clone().translateZ(50);

axle1.display();

axle2.display();

box1 = cube(50,30,80).translate(0,15,20);

box2 = cube(44,24,74).translate(0.25,19,21);

box = box1.difference(box2);

box.display();