**4.7 有倒角方筒**

var Conversions = Core.Conversions;

var Debug = Core.Debug;

var Path2D = Core.Path2D;

var Point2D = Core.Point2D;

var Point3D = Core.Point3D;

var Matrix2D = Core.Matrix2D;

var Matrix3D = Core.Matrix3D;

var Mesh3D = Core.Mesh3D;

var Plugin = Core.Plugin;

var Tess = Core.Tess;

var Sketch2D = Core.Sketch2D;

var Solid = Core.Solid;

var Vector2D = Core.Vector2D;

var Vector3D = Core.Vector3D;

var maxLen = 100;

// Round edged box shell from 2D Paths

// Template Code:

params = [

{ "id": "dimx", "displayName": "X Dim", "type": "length", "rangeMin": 1, "rangeMax": maxLen, "default": 20 },

{ "id": "dimy", "displayName": "Y Dim", "type": "length", "rangeMin": 1, "rangeMax": maxLen, "default": 20 },

{ "id": "dimz", "displayName": "Z Dim", "type": "length", "rangeMin": 1, "rangeMax": maxLen, "default": 20 },

{ "id": "wall", "displayName": "Wall", "type": "length", "rangeMin": 0.1, "rangeMax": 100, "default": 3 },

{ "id": "rad\_outer", "displayName": "Outer Radius", "type": "length", "rangeMin": 0, "rangeMax": 40, "default": 6 },

{ "id": "rad\_inner", "displayName": "inner Radius", "type": "length", "rangeMin": 0, "rangeMax": 40, "default": 6 },

{ "id": "x\_pos","displayName": "x position","type": "float","rangeMin": -100,"rangeMax": 100,"default": 0},

{ "id": "y\_pos","displayName": "y position","type": "float","rangeMin": -100,"rangeMax": 100,"default": 0 }

];

function process(params) {

var x\_pos = params.x\_pos;

var y\_pos = params.y\_pos;

var dimx = params["dimx"];

var dimy = params["dimy"];

var dimz = params["dimz"];

var wall = params["wall"];

var rad\_outer = params["rad\_outer"];

var rad\_inner = params["rad\_inner"];

var orig = [0,0,0];

dimx = Math.max(1 ,Math.min(dimx, maxLen));

dimy = Math.max(1 ,Math.min(dimy, maxLen));

dimz = Math.max(1 ,Math.min(dimz, maxLen));

var lmin = Math.min(dimx, dimy);

wall = Math.max(0.1 ,Math.min(wall, lmin/2));

rad\_outer = Math.max(0.1 ,Math.min(rad\_outer , lmin/2));

rad\_inner = Math.max(0.1 ,Math.min(rad\_inner , lmin/2));

var path = new Path2D();

// outer wall (cw)

path.moveTo(x\_pos+rad\_outer,y\_pos+0);

path.quadraticCurveTo(x\_pos+0,y\_pos+0,x\_pos+0,y\_pos+rad\_outer);

path.lineTo(x\_pos+0,y\_pos+dimy-rad\_outer);

path.quadraticCurveTo(x\_pos+0,y\_pos+dimy,x\_pos+rad\_outer,y\_pos+dimy);

path.lineTo(x\_pos+dimx-rad\_outer,y\_pos+dimy);

path.quadraticCurveTo(x\_pos+dimx,y\_pos+dimy,x\_pos+dimx,y\_pos+dimy-rad\_outer);

path.lineTo(x\_pos+dimx,y\_pos+rad\_outer);

path.quadraticCurveTo(x\_pos+dimx,y\_pos+0,x\_pos+dimx-rad\_outer,y\_pos+0);

path.lineTo(x\_pos+rad\_outer,y\_pos+0);

// inner wall (ccw)

if (wall < lmin/2) {

path.moveTo(x\_pos+rad\_inner,y\_pos+wall);

path.lineTo(x\_pos+dimx-rad\_inner,y\_pos+wall);

path.quadraticCurveTo(x\_pos+dimx-wall,y\_pos+wall,x\_pos+dimx-wall,y\_pos+rad\_inner);

path.lineTo(x\_pos+dimx-wall,y\_pos+dimy-rad\_inner);

path.quadraticCurveTo(x\_pos+dimx-wall,y\_pos+dimy-wall,x\_pos+dimx-rad\_inner,y\_pos+dimy-wall);

path.lineTo(x\_pos+rad\_inner,y\_pos+dimy-wall);

path.quadraticCurveTo(x\_pos+wall,y\_pos+dimy-wall,x\_pos+wall,y\_pos+dimy-rad\_inner);

path.lineTo(x\_pos+wall,y\_pos+rad\_inner);

path.quadraticCurveTo(x\_pos+wall,y\_pos+wall,x\_pos+rad\_inner,y\_pos+wall);

}

var outer = Solid.extrude([path], dimz);

return outer;

}