**4.32 多扭轉柱**

var Debug = Core.Debug;

var Mesh3D = Core.Mesh3D;

var Path2D = Core.Path2D;

var Plugin = Core.Plugin;

var Tess = Core.Tess;

var Solid = Core.Solid;

params = [

{ "id": "turn\_angle","displayName": "繞z軸公旋旋轉度數","type": "float","rangeMin": 0,"rangeMax": 360,"default": 180},

{ "id": "twist\_angle","displayName": "自旋度數","type": "float","rangeMin": 0,"rangeMax": 360,"default": 180 },

{ "id": "turn\_times","displayName": "繞z軸公旋放大倍數","type": "float","rangeMin": 0,"rangeMax": 10,"default": 1},

{ "id": "twist\_times","displayName": "自旋放大倍數","type": "float","rangeMin": 0,"rangeMax": 10,"default": 1 },

{ "id": "dd","displayName": "方柱繞z軸分身個數", "type": "int", "rangeMin": 1, "rangeMax": 8, "default": 5 },

{ "id": "sideLength","displayName": "柱邊長","type": "float","rangeMin": 0,"rangeMax": 10,"default": 4},

{ "id": "x1\_pos","displayName": "起始層柱中心x座標","type": "float","rangeMin": 0,"rangeMax": 20,"default":10},

{ "id": "y1\_pos","displayName": "起始層柱中心y座標","type": "float","rangeMin": 0,"rangeMax": 20,"default":10},

{ "id": "height", "displayName": "高度", "type": "length", "rangeMin": 1, "rangeMax": 200, "default": 50 },

{ "id": "layers","displayName": "層數", "type": "int", "rangeMin": 3, "rangeMax": 360, "default": 360 },

{ "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 },

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

];

function process(params) {

var turn\_angle=params.turn\_angle;

var twist\_angle=params.twist\_angle;

var turn\_times=params.turn\_times;

var twist\_times=params.twist\_times;

var dd=params.dd;

var sideLength=params.sideLength;

var x1\_pos = params.x1\_pos;

var y1\_pos = params.y1\_pos;

var height = params.height;

var layers = params.layers;

var x\_pos = params.x\_pos;

var y\_pos = params.y\_pos;

var z\_pos = params.z\_pos;

var mesh = new Mesh3D();

//宣告各層各點座標

var p1=[];

var p2=[];

var p3=[];

var p4=[];

for (var j = 0 ; j <dd ;j++) {

p1[j]=[];

p2[j]=[];

p3[j]=[];

p4[j]=[]; }

//座標暫存

var tz; //每層z座標

var tx1 ; //自轉離柱中心點座標

var ty1 ;

var tx2 ;

var ty2;

var tx3 ;

var ty3 ;

var tx4 ;

var ty4 ;

var bx1 ; //柱中心點公轉與分身xy座標

var by1 ;

var small\_angle=twist\_angle/layers\*2\*Math.PI/360;   
 //自旋每層角度增加量徑度量

var big\_angle=turn\_angle/layers\*2\*Math.PI/360;   
 //公轉每層角度增加量徑度量

var height\_inc=height/layers; //自旋每層角度增加量徑度量

var angle45 = 45/360\*2\*Math.PI ; //中心點距各點角度

var centerAngle=Math.atan(y1\_pos/x1\_pos); //中心點角度

var anglen=2\*Math.PI/dd; //分身每次旋轉角度

var small\_radius=Math.sqrt(2)\*sideLength; //自轉半徑

var small\_radius\_inc=small\_radius\*(twist\_times-1)/layers;   
 //自轉半徑每層增量

var big\_radius=Math.sqrt(x1\_pos\*x1\_pos+y1\_pos\*y1\_pos);   
 //公轉半徑

var big\_radius\_inc=big\_radius\*(turn\_times-1)/layers;   
 //公轉半徑每層增量

for (var n=0; n< dd; n++) //對所有柱子

{

for (var i =0; i < layers; i++) //設定各點座標

{

tz=i\*height\_inc; //z軸每層升起

//以下8點自轉離柱中心點座標

tx1 = (small\_radius+i\*small\_radius\_inc)\*Math.cos(angle45\*5+i\* small\_angle);

ty1 =(small\_radius+i\*small\_radius\_inc)\*Math.sin(angle45\*5+i\* small\_angle);

tx2 =(small\_radius+i\*small\_radius\_inc)\*Math.cos(angle45\*7+i\* small\_angle);

ty2 =(small\_radius+i\*small\_radius\_inc)\*Math.sin(angle45\*7+i\* small\_angle);

tx3 =(small\_radius+i\*small\_radius\_inc)\*Math.cos(angle45+i\* small\_angle);

ty3 = (small\_radius+i\*small\_radius\_inc)\*Math.sin(angle45+i\* small\_angle);

tx4 =(small\_radius+i\*small\_radius\_inc)\*Math.cos(angle45\*3+i\* small\_angle);

ty4 = (small\_radius+i\*small\_radius\_inc)\*Math.sin(angle45\*3+i\* small\_angle);

//以下8點柱中心點公轉與分身xy座標

bx1 = (big\_radius+i\*big\_radius\_inc)\*Math.cos(centerAngle+i\*big\_angle+n\*anglen);

by1 =(big\_radius+i\*big\_radius\_inc)\*Math.sin(centerAngle+i\*big\_angle+n\*anglen);

p1[n][i]=([x\_pos+tx1+bx1, y\_pos+ty1+by1, z\_pos+tz]);

p2[n][i]=([x\_pos+tx2+bx1, y\_pos+ty2+by1, z\_pos+tz]);

p3[n][i]=([x\_pos+tx3+bx1, y\_pos+ty3+by1, z\_pos+tz]);

p4[n][i]=([x\_pos+tx4+bx1, y\_pos+ty4+by1, z\_pos+tz]);

}

//用四邊形掃邊(不包含最後封口)

for ( i = 0; i < layers-1; i++)

{

mesh.quad(p1[n][i], p2[n][i],p2[n][i+1],p1[n][i+1]);

mesh.quad(p2[n][i], p3[n][i],p3[n][i+1],p2[n][i+1]);

mesh.quad(p3[n][i], p4[n][i],p4[n][i+1],p3[n][i+1]);

mesh.quad(p4[n][i], p1[n][i],p1[n][i+1],p4[n][i+1]);

}

mesh.quad(p1[n][0], p4[n][0],p3[n][0],p2[n][0]);   
 //起始四邊形封底

mesh.quad(p1[n][layers-1], p2[n][layers-1],p3[n][layers-1],p4[n][layers-1]);   
 //最後四邊形封頂

}

return Solid.make(mesh);

}