```
import toxi.geom.*;
import toxi.physics.*;
import toxi.physics.behaviors.*;
import peasy.*;
import processing.dxf.*;

ArrayList<Particle> history = new ArrayList<Particle>();
Particle pastParticle;

boolean saveDXF = false;

//params panel
int number = 9; //segments
float strength = 0.05; //strength of strings
float radius = 5; //radius of nodes
int numcentr = 3;
boolean gravity = true;
float bigrad = 205;
float smallrad = 100;
int chainnum = 15;
float x;
float y;
int increment = 0;

VerletPhysics physics;
myShape myshape; //call Class
PeasyCam cam;

void setup()
{
  size(900, 900, P3D);
  cam = new PeasyCam(this, 500);


  physics=new VerletPhysics(); //start physics
  if (gravity)
  {
    physics.addBehavior(new GravityBehavior(new Vec3D(0,0.05,0))); //gravity behavior
  }
  physics.setDrag (0.01);
  //physics.setWorldBounds(new Rect(10, 10, width-20, height-20)); //set World constraints

  myshape= new myShape(0, 0, number,strength, numcentr, bigrad, smallrad, chainnum); //call for class object
creation

}

void draw()
{
  if (keyPressed)
  {
      if (key == 's' || key == 'S')
      {
        delay(500);
        saveDXF = true;
      }
    }
```

```
   if ( saveDXF == true )
   {
     beginRaw( DXF, "files/SuperS.dxf" );
   }

  physics.update(); //update physics
  background(255);
  myshape.display(radius);

  if (increment < 900)
  {
   for(int i=0; i< physics.particles.size()-numcentr ; i++)
   {
     if((i+number*chainnum)%chainnum*number>number)
     {
       Particle pastParticle = new Particle (physics.particles.get(i).x, physics.particles.get(i).y,
physics.particles.get(i).z);
       history.add(pastParticle);
       pastParticle.z += increment;
     }
   }

  }
  increment += 5;

  for(int i=chainnum*number-number; i< history.size()-1 ; i++)
  {
   //history.get(i).display(radius);
   stroke(0);
   strokeWeight(1);
   beginShape(TRIANGLE_STRIP);
   vertex(history.get(i).x, history.get(i).y, history.get(i).z);
   vertex(history.get(i+1).x, history.get(i+1).y, history.get(i+1).z);
   vertex(history.get(i-(chainnum-2)*number+1).x, history.get(i-(chainnum-2)*number+1).y, history.get(i-
(chainnum-2)*number+1).z);


   endShape();
   beginShape(TRIANGLE_STRIP);
   vertex(history.get(i).x, history.get(i).y, history.get(i).z);
   vertex(history.get(i-(chainnum-2)*number+1).x, history.get(i-(chainnum-2)*number+1).y, history.get(i-
(chainnum-2)*number+1).z);
   vertex(history.get(i-(chainnum-2)*number).x, history.get(i-(chainnum-2)*number).y, history.get(i-
(chainnum-2)*number).z);
   endShape();
   //line (history.get(i).x, history.get(i).y, history.get(i).z, history.get(i-1).x, history.get(i-1).y, history.get(i-1).z);
  }


  if ( saveDXF == true )
  {
     endRaw();
     saveDXF = false;
   }

}
```

```
class myShape
{
  ArrayList<Particle> myparticles = new ArrayList<Particle>();
  ArrayList<Particle> centres = new ArrayList<Particle>();
  ArrayList<Chain> chainss = new ArrayList<Chain>();
  Chain chain;
  float angle;
  float angle2;

  //class constructor
  myShape(float x_temp, float y_temp, int num_temp, float str_temp, int cen_temp, float big_temp, float
small_temp, int chainnum_temp)
  {
   number=num_temp;
   strength=str_temp;
   numcentr=cen_temp;
   bigrad=big_temp;
   smallrad= small_temp;
   chainnum= chainnum_temp;
   x=x_temp;
   y=y_temp;

   //loop for connections on the boundary
    for (int i=0; i< number; i++)
    {
      angle=TWO_PI/number;
      Particle particle= new Particle(x+ bigrad*cos(angle*i), y+ bigrad*sin(angle*i),0);
      //physics.addBehavior(new AttractionBehavior(particle, smallrad, -strength*5)); // ?????
      physics.addParticle(particle); //apply physics to particle
      myparticles.add(particle); //add to array

      if(i > 0) //connect all particle except first and last
      {
        chain = new Chain (myparticles.get(i-1), particle, chainnum, strength); //creating chains
        chainss.add(chain);
      }
     if (i == number-1) //connect first and last particle
      {
        chain = new Chain (myparticles.get(i), myparticles.get(0), chainnum, strength); //creating chains
        chainss.add(chain);
      }
    }


    //loop for creating centres
    for (int j=0; j< numcentr; j++)
    {
      angle2= TWO_PI/numcentr;
      Particle centre= new Particle(x+ smallrad*cos(angle2*j), y+ smallrad*sin(angle2*j),0);
      physics.addBehavior(new AttractionBehavior(centre, smallrad, -strength*5)); // ?????
      //physics.addBehavior(new AttractionBehavior(centre, smallrad/2, -strength*2)); // ?????
      physics.addParticle(centre); //apply physics to particle
      centres.add(centre); //add to array of centres
      centre.lock();
    }

  }
  void display(float radius)
  {
    for (Chain h:chainss)
    {
      h.lines();
    }
  }
}
```

---

```
  class Chain
{
    ArrayList<Particle> chains = new ArrayList<Particle>();
    ArrayList<Particle> starts = new ArrayList<Particle>();
    ArrayList<Particle> ends = new ArrayList<Particle>();
    Particle chain;
    int chainnum;
    float str;

    Chain(Particle start_temp, Particle end_temp, int chainnum_temp, float str_temp)
  {
    chainnum= chainnum_temp;
    str= str_temp;
    Particle start = start_temp;
    Particle end= end_temp;
    starts.add(start);
    ends.add(end);

    //distance between two points on a grid
    float distance = sqrt( pow(end.x-start.x,2) + pow(end.y-start.y,2) );
    float changex = end.x - start.x;
    float changey = end.y - start.y;

    for (int i=1;i<chainnum;i++)
    {
      Particle chain = new Particle(start.x+(changex/chainnum)*i, start.y+(changey/chainnum)*i,0);
      physics.addBehavior(new AttractionBehavior(chain, distance/(chainnum), -str)); // ?????
      physics.addParticle(chain);
      chains.add(chain);

      if (i==1)
      {
        VerletSpring spring = new VerletSpring(start, chains.get(0), distance/(chainnum), 1);
        physics.addSpring(spring);
      }
      if (i>1 && i<chainnum-1)
      {
        VerletSpring spring = new VerletSpring(chains.get(i-1), chains.get(i-2), distance/(chainnum), 1);
        physics.addSpring(spring);
      }
      if (i==chainnum-1)
      {
        VerletSpring spring = new VerletSpring(chains.get(i-1), chains.get(i-2), distance/(chainnum), 1);
        physics.addSpring(spring);
        VerletSpring spring2 = new VerletSpring(chains.get(i-1), end, distance/(chainnum), 1);
        physics.addSpring(spring2);
      }
    }
  }

  void display()
  {
    for (Particle c:chains)
    {
      c.display(5);
    }
  }

  void lines()
  {
    for (int i=1; i<chainnum-1; i++)
    {
      line (chains.get(i-1).x, chains.get(i-1).y, chains.get(i).x, chains.get(i).y);
      if (i==1)
      {
        line (starts.get(0).x, starts.get(0).y, chains.get(i-1).x, chains.get(i-1).y);
      }
      if (i==chainnum-2)
      {
        line (ends.get(0).x, ends.get(0).y, chains.get(i).x, chains.get(i).y);
      }
    }
  }
}

// create one new Verlet Paricle and name it

class Particle extends VerletParticle
{
  Particle(float x, float y, float z)
  {
    super(x,y,z);
  }

  // All we're doing really is adding a display() function to a VerletParticle
  void display(float rad)
  {
    fill(0,150);
    stroke(0);
    strokeWeight(5);
    pushMatrix();
    translate(x,y,z);
    sphere(rad);
    popMatrix();
  }
}
```