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

PeasyCam cam;
VerletPhysics physics;
myShape myshape;

boolean saveDXF = false;
//PVector [][] globe;
Particle [][] pcords;
Particle [][] ccords;

int number = 15; //segments
float strength = 0.05; //strength of strings
float radius = 5; //radius of nodes
int numcentr = 4;
boolean gravity = true;
float bigrad = 280;
float smallrad = 150;
int chainnum = 8;
boolean perimeter = false;
boolean chainz = true;
float x;
float y;
float z;

void setup()
{
  size(700, 700, P3D);
  cam = new PeasyCam(this, 1500);
  physics = new VerletPhysics();
  physics.setDrag (0.01);
  physics.addBehavior(new GravityBehavior(new Vec3D(0,0.05,0)));

  pcords= new Particle [number+1][number];
  ccords= new Particle [numcentr+1][numcentr+1];
  //storing the globe vertices as an array, total+1 for the sphere to close (end where is starts)
  myshape= new myShape(50, 50, 50, number,strength, numcentr, bigrad, smallrad, chainnum);
}

void draw()
{
        if (keyPressed)
        {
        if (key == 's' || key == 'S')
        {
                delay(500);
                saveDXF = true;
        }
    }
    if ( saveDXF == true )
    {
                beginRaw( DXF, "files/SuperS.dxf" );
    }
```

```
        physics.update();
        background(255);
        myshape.display(radius);
        if (perimeter == true)
        {
        myshape.lines();
        }

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

}

void doMesh()
{
                for (int i =0; i<number; i++)
        {
                beginShape(TRIANGLE_STRIP);
                for (int j=0; j<number;j++)
                {
                        Particle v1 = pcords[i][j];
                        Particle v2 = pcords[i+1][j];
                        stroke(100);
                        strokeWeight(2);
                        vertex(v1.x,v1.y,v1.z);
                        vertex(v2.x,v2.y,v2.z);
                        if (j == number-1)
                        {
                                Particle v3 = pcords[i][0];
                                Particle v4 = pcords[i+1][0];
                                vertex(v3.x,v3.y,v3.z);
                                vertex(v4.x,v4.y,v4.z);
                        }
                }
                endShape();
        }
}

-----------------------------------------------------------------------------------------------------------------------------

class myShape
{
 ArrayList<Particle> myparticles = new ArrayList<Particle>();
 ArrayList<Particle> centres = new ArrayList<Particle>();
 ArrayList<Chain> chainss = new ArrayList<Chain>();
 Chain chain;
 Chain chain2;
 float angle;
 float angle2;
 boolean perimeter;

 //class constructor
```

```
  myShape(float x_temp, float y_temp, float z_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;
   z=z_temp;



    //-------------------------------------------------------------------------------
    for (int i = 0; i < number+1; i++) //longitude
    {
      float lat = map(i, 0, number, 0, PI);
      for (int j = 0; j < number; j++) //lattitude
      {
        float lon = map(j, 0, number, 0, TWO_PI);
        float rx = bigrad * sin(lat) * cos(lon);
        float ry = bigrad * sin(lat) * sin(lon);
        float rz = bigrad * cos(lat);
        pcords[i][j] = new Particle (x+ rx, y+ ry, z+rz);
        physics.addParticle(pcords[i][j]); //apply physics to particle
        myparticles.add(pcords[i][j]); //add to array
      }
    }
    //--------------------------------------------------------the loops below can also be used for chain
    //loop for the connection with particle neighbours

    for (int i = 0; i < number; i++)
    {
      for (int j = 0; j < number-1; j++)
      {
        Particle p1 = pcords[i][j];
        Particle p2 = pcords[i][j+1];
        Particle p3 = pcords[i+1][j];
        float distance1 = dist(p1.x, p1.y, p1.z, p2.x, p2.y, p2.z);
        float distance2 = dist(p1.x, p1.y, p1.z, p3.x, p3.y, p3.z);
        //add chains

        chain = new Chain (p1, p2, chainnum, strength); //creating chains
        chainss.add(chain);
        chain2 = new Chain (p1, p3, chainnum, strength); //creating chains
        chainss.add(chain2);

        //add springs
        if (perimeter == true)
        {
        VerletSpring spring1 = new VerletSpring(p1, p2, distance1, 0.1);
        physics.addSpring(spring1);
        VerletSpring spring2 = new VerletSpring(p1, p3, distance2, 0.1);
        physics.addSpring(spring2);
        }
      }
    }
    //loop for connections at the last row of i (straight line going right)
    for (int j = 0; j < number -1; j++)
    {
        Particle p1 = pcords[number][j];
        Particle p2 = pcords[number][j+1];
        //add chains
        chain = new Chain (p1, p2, chainnum, strength); //creating chains
        chainss.add(chain);
        //add springs
        if (perimeter == true)
        {
         float distance1 = dist(p1.x, p1.y, p1.z, p2.x, p2.y, p2.z);
         VerletSpring spring1 = new VerletSpring(p1, p2, distance1, 0.1);
         physics.addSpring(spring1);
        }
    }
    //loop for connections at the last column of j (straight line going downwards)
    for (int i = 0; i < number; i++)
    {
        Particle p1 = pcords[i][number-1];
        Particle p2 = pcords[i+1][number-1];
        //add chains
        chain = new Chain (p1, p2, chainnum, strength); //creating chains
        chainss.add(chain);
        //add springs
        if (perimeter == true)
        {
         float distance1 = dist(p1.x, p1.y, p1.z, p2.x, p2.y, p2.z);
         VerletSpring spring1 = new VerletSpring(p1, p2, distance1, 0.1);
         physics.addSpring(spring1);
        }
    }
    //loop for connections for the last column to connect back to the first one
    for (int i = 0; i < number; i++)
    {
        Particle p1 = pcords[i][number-1];
        Particle p2 = pcords[i][0];
        //add chains
        chain = new Chain (p1, p2, chainnum, strength); //creating chains
        chainss.add(chain);
        //add springs
        if (perimeter == true)
        {
         float distance1 = dist(p1.x, p1.y, p1.z, p2.x, p2.y, p2.z);
         VerletSpring spring1 = new VerletSpring(p1, p2, distance1, 0.1);
         physics.addSpring(spring1);
        }
    }

    //loop for creating centres
    for (int i = 0; i < numcentr+1; i++) //longitude
    {
      float lat = map(i, 0, numcentr, 0, PI);
      for (int j = 0; j < numcentr; j++) //lattitude
      {
        float lon = map(j, 0, numcentr, 0, TWO_PI);
        float rx = smallrad * sin(lat) * cos(lon);
        float ry = smallrad * sin(lat) * sin(lon);
        float rz = smallrad * cos(lat);
        //ccords[i][j] = new Particle (random(0,smallrad*2), random(0,smallrad*2), random(0,smallrad*2));
        ccords[i][j] = new Particle (x+rx, y+ry, z+rz);
        physics.addBehavior(new AttractionBehavior(ccords[i][j], smallrad, -strength*15));
        physics.addParticle(ccords[i][j]); //apply physics to particle
        centres.add(ccords[i][j]); //add to array
        ccords[i][j].lock(); //lock the particle
      }
    }

  }

  void display(float radius)
  {
    if (chainz == true)
    {
      for (Chain h:chainss)
      {
        h.lines();
      }
    }
  }

  void lines()
  {
    strokeWeight(2);
    for (int i = 0; i < number; i++)
    {
      for (int j = 0; j < number-1; j++)
      {
        Particle p1 = pcords[i][j];
        Particle p2 = pcords[i][j+1];
        Particle p3 = pcords[i+1][j];
        line(p1.x, p1.y, p1.z, p2.x, p2.y, p2.z);
        line(p1.x, p1.y, p1.z, p3.x, p3.y, p3.z);
      }
    }
    //loop for connections at the last row of i (straight line going right)
    for (int j = 0; j < number -1; j++)
    {
        Particle p1 = pcords[number][j];
        Particle p2 = pcords[number][j+1];
        line(p1.x, p1.y, p1.z, p2.x, p2.y, p2.z);
    }
    //loop for connections at the last column of j (straight line going downwards)
    for (int i = 0; i < number; i++)
    {
        Particle p1 = pcords[i][number-1];
        Particle p2 = pcords[i+1][number-1];
        line(p1.x, p1.y, p1.z, p2.x, p2.y, p2.z);
    }
    //loop for connections for the last column to connect back to the first one
    for (int i = 0; i < number; i++)
    {
        Particle p1 = pcords[i][number-1];
        Particle p2 = pcords[i][0];
        line(p1.x, p1.y, p1.z, p2.x, p2.y, p2.z);
    }
  }
 }
```

```
 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) + pow(end.z-start.z,2) );
  float changex = end.x - start.x;
  float changey = end.y - start.y;
  float changez = end.z - start.z;

  for (int i=1;i<chainnum;i++)
  {
   Particle chain = new Particle(start.x+(changex/chainnum)*i, start.y+(changey/chainnum)*i,
start.z+(changez/chainnum)*i);
   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(2);
  }
 }

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

------------------------------------------------------------------------------------------------------------------------

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(100,150);
  stroke(100);
  strokeWeight(2);
  pushMatrix();
  translate(x, y, z);
  sphere(rad);
  popMatrix();
 }
}
```