```
import toxi.geom.*;
import toxi.physics2d.*;
import toxi.physics2d.behaviors.*;

//params panel for a cell
int number = 8; //segments
float strength = 0.1; //strength of strings
float radius = 2; //radius of nodes
int numcentr = 3;
boolean gravity = true;
float bigrad ;//= 100; temporary lock
float smallrad ;//= 50;
int chainnum ;//= 10;

//packing parametres
ArrayList<Cell> cells;
int count = 55;
int attempts = 100000;
float a = 160;
float b = 80;
int c = 17;

//other params

VerletPhysics2D physics;
Cell cell; //call Class


void setup()
{
 size(600, 600);
 smooth();

 physics=new VerletPhysics2D(); //start physics

 //gravity behaviour turn on/turn off
 if (gravity)
 {
  physics.addBehavior(new GravityBehavior(new Vec2D(0,0.05))); //gravity behavior
 }
 physics.setDrag (0.01); //set drag for smoother and less volatile
 physics.setWorldBounds(new Rect(10, 10, width-20, height-20)); //set World constraints

 cells = new ArrayList<Cell>(); //launch an array to store created cells

 //-----------------loop for creating cells-----------------
 while (cells.size()<count)
 {
  if (a<30 || c<3)
  {
```

```
      break;
    }
    Cell newC = newCell();
    if (newC != null)
    {
      cells.add(newC);
    }
    if (attempts<10)
    {
      attempts += 100000;
      a -= 10;
      b -= 5;
      c -= 1;
    }
    attempts--;
  }
  //---------------end of loop-------------------------------
}

void draw()
{
  physics.update(); //update physics
  background(255);
  for (Cell c:cells)
  {
    noFill();
    stroke(0);
    strokeWeight(0.5);
    c.display(radius);
    //c.showLines();
  }
}


Cell newCell()
{
  float newBigRad = a;
  float newSmallRad = b;
  int newChainNum = c;
  float x= random(0, width);
  float y= random(0, height);

  boolean valid = true;
  for (Cell c:cells)
  {
    float d= dist(x, y, c.x, c.y);
    if (d<c.bigrad + newBigRad/3)
    {
      valid = false;
      break;
    }
```

```
  }
  if (valid)
  {
    return new Cell(x, y, int(random(6,15)), strength, int(random(3,7)), newBigRad, newSmallRad,
newChainNum);
  }
  else
  {
    return null;
  }
}
```

```
class Cell
{
  ArrayList<Particle> myparticles = new ArrayList<Particle>();
  ArrayList<Particle> centres = new ArrayList<Particle>();
  ArrayList<Chain> chainss = new ArrayList<Chain>();
  Chain chain;
  float angle;
  float angle2;
  float x; //used in packing
  float y; //used in packing
  float bigrad; //used in packing
  float smallrad; //used in packing

  //class constructor
  Cell(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));
      //physics.addBehavior(new AttractionBehavior(particle, 10, -strength*5));
      physics.addParticle(particle); //apply physics to particle
      myparticles.add(particle); //add to array
      //particle.lock();

      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));
    physics.addBehavior(new AttractionBehavior(centre, smallrad, -strength*50));
    //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();
   }

   //connecting centres
   // for (int j=0; j< numcentr; j++)
   // {
   //   for (int k=j+1;k<numcentr;k++)
   //   {
   //     float distance = sqrt( pow(centres.get(k).x-centres.get(j).x,2) + pow(centres.get(k).y-
centres.get(j).y,2) );
   //     VerletSpring2D spring= new VerletSpring2D(centres.get(j),centres.get(k),distance,1);
   //     physics.addSpring(spring);
   //   }
   // }

   // // //connecting all particles
   // for (int i=0; i< number; i++)
   // {
   //   for (int j=i+2; j<number; j++)
   //   {
   //     float distance = sqrt( pow(myparticles.get(j).x-myparticles.get(i).x,2) +
pow(myparticles.get(j).y-myparticles.get(i).y,2) );
   //     VerletSpring2D spring= new
VerletSpring2D(myparticles.get(i),myparticles.get(j),distance,0.1);
   //     physics.addSpring(spring);
   //   }
   // }

   // //connecting centres with particles
   // for (int i=0; i< numcentr; i++)
   // {
```

```
  //   for (int j=0; j< number; j++)
  //   {
  //     float distance = sqrt( pow(centres.get(i).x-myparticles.get(j).x,2) + pow(centres.get(i).y-
myparticles.get(j).y,2) );
  //     VerletSpring2D spring= new VerletSpring2D(centres.get(i),myparticles.get(j),distance,0.1);
  //     physics.addSpring(spring);
  //   }
  // }
 }

 void showLines()
 {
  stroke(0,150);
  strokeWeight(2);
  for (int i=1; i<number; i++)
  {
   Particle pthis = myparticles.get(i);
   Particle plast = myparticles.get(i-1);
   line (pthis.x, pthis.y, plast.x, plast.y);
   if (i == number-1)
   {
    Particle first= myparticles.get(0);
    Particle lasty= myparticles.get(number-1);
    line (first.x, first.y, lasty.x, lasty.y);
   }
  }
  //loop for connecting centres to all the rest
  for (int j=0; j<numcentr; j++)
  {
   for (int i=0; i<number; i++)
   {
    line (centres.get(j).x, centres.get(j).y, myparticles.get(i).x, myparticles.get(i).y);
   }
  }
  //loop for connecting centres together
  for (int i=1; i <numcentr; i++)
  {
   line (centres.get(i-1).x, centres.get(i-1).y, centres.get(i).x, centres.get(i).y);
   if (i == numcentr-1)
   {
    line (centres.get(0).x, centres.get(0).y, centres.get(i).x, centres.get(i).y);
   }
  }
 }

 void display(float radius)
 {
  // for (Particle p:myparticles)
  // {
  //   p.display(radius);
  // }
```

```
   // for (Particle c:centres)
   // {
   //   c.display(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);
     physics.addBehavior(new AttractionBehavior(chain, distance/(chainnum), -str*5)); // ?????
     physics.addBehavior(new AttractionBehavior(chain, bigrad, str/50)); // ?????
     physics.addParticle(chain);
     chains.add(chain);

     if (i==1)
     {
       VerletSpring2D spring = new VerletSpring2D(start, chains.get(0), distance/(chainnum), str);
       physics.addSpring(spring);
     }
     if (i>1 && i<chainnum-1)
     {
```

```
      VerletSpring2D spring = new VerletSpring2D(chains.get(i-1), chains.get(i-2),
distance/(chainnum), str);
      physics.addSpring(spring);
    }
    if (i==chainnum-1)
    {
      VerletSpring2D spring = new VerletSpring2D(chains.get(i-1), chains.get(i-2),
distance/(chainnum), str);
      physics.addSpring(spring);
      VerletSpring2D spring2 = new VerletSpring2D(chains.get(i-1), end, distance/(chainnum), str);
      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 VerletParticle2D
{
  Particle(float x, float y)
  {
   super(x,y);
  }

  // All we're doing really is adding a display() function to a VerletParticle
```

```
  void display(float rad)
  {
    fill(0,150);
    stroke(0);
    strokeWeight(2);
    ellipse(x,y,rad,rad);
  }
}
```