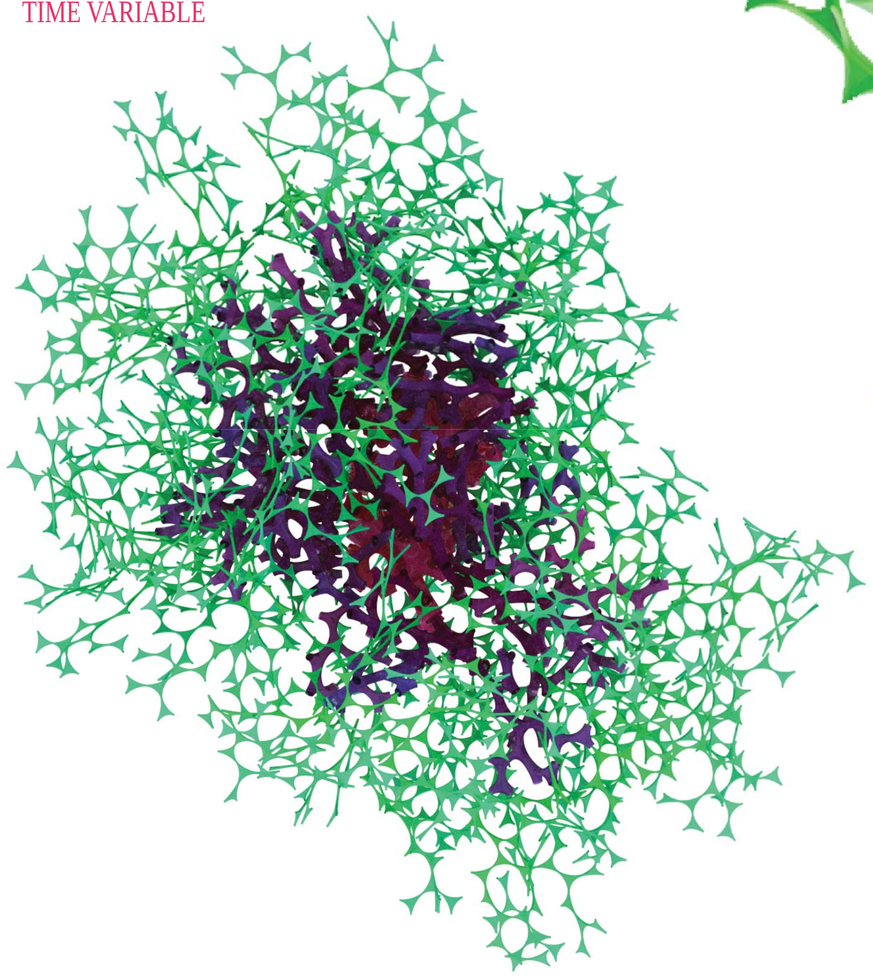
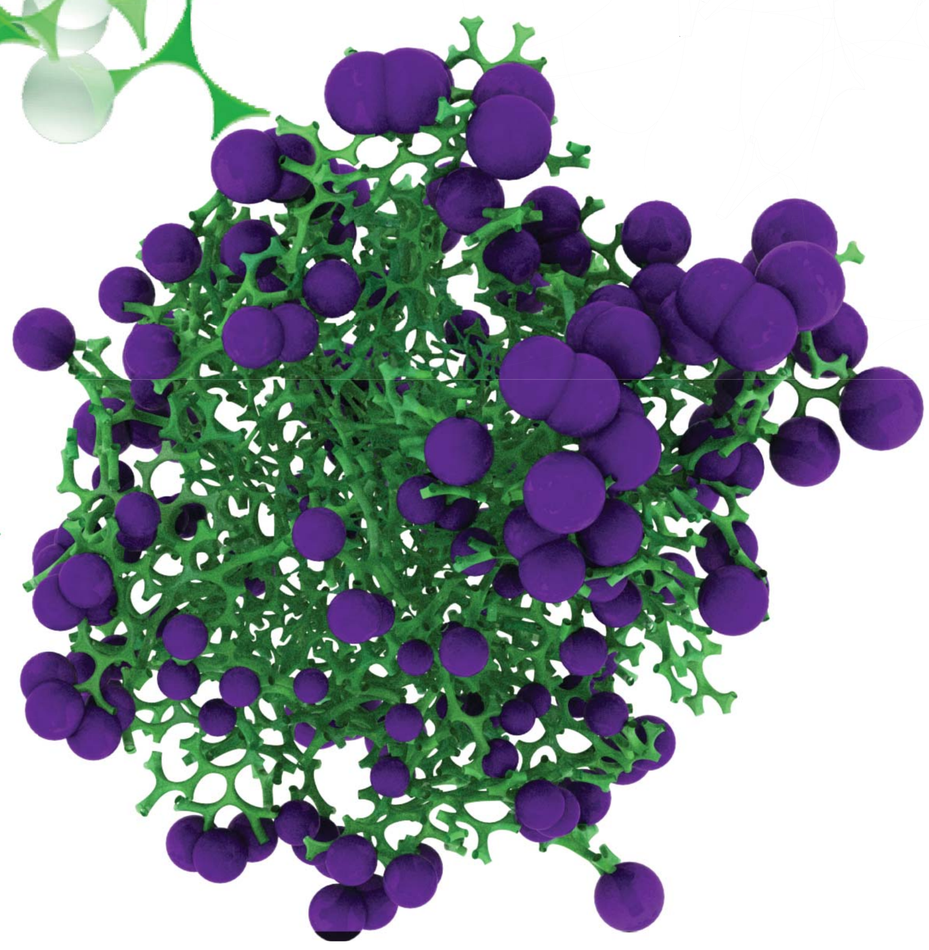
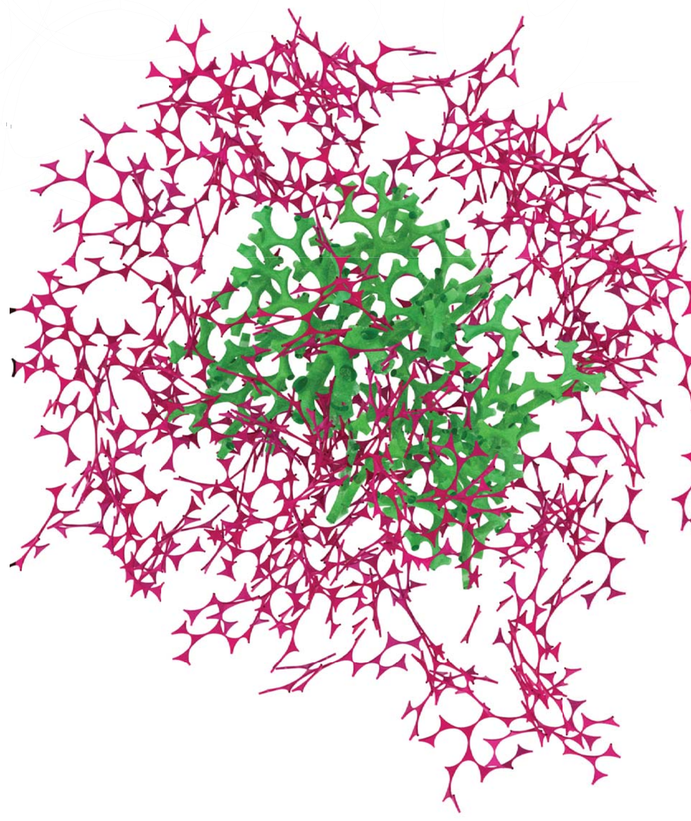
VS CODING FORM

OMAR G. MUNOZ

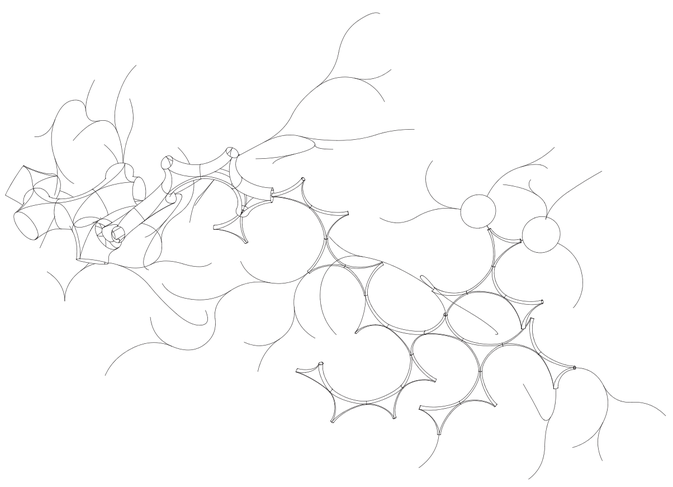
TIME VARIABLE



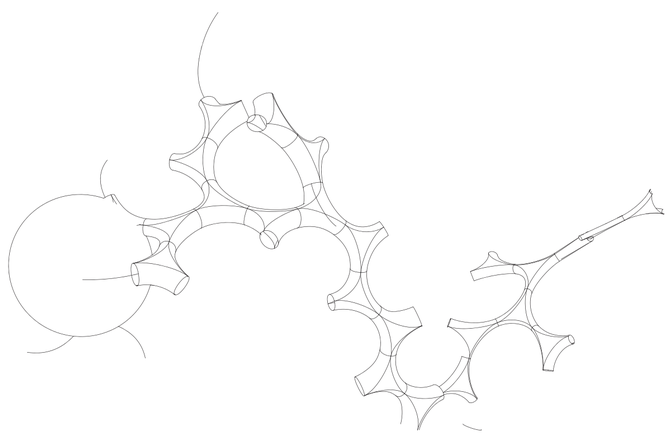




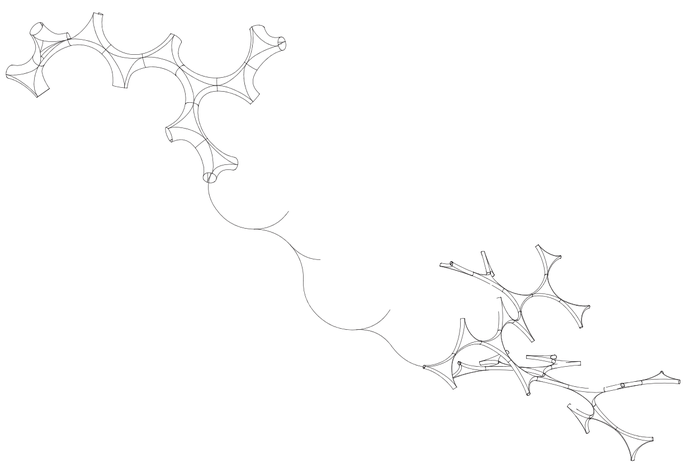
OUTPUT 1



OUTPUT 2

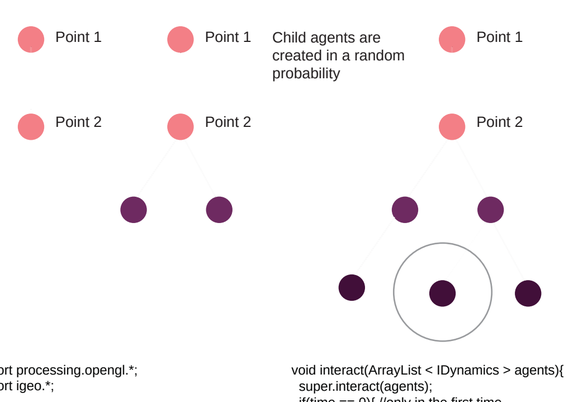


OUTPUT 3



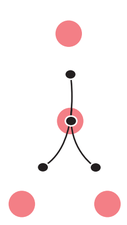
AGENT ALGORITHM DIAGRAM

LINE AGENT



AGENT GEOMETRY DIAGRAM

SURFACE AROUND LINEAGENT

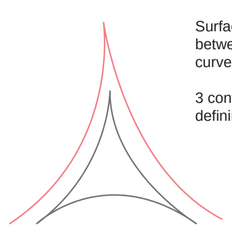


A two degree curve is created between 3 points.

1. Middle point of start and end points.

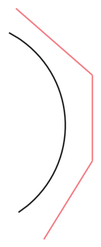
2. Point 2

3. Midpoint of Point 2 and the point of the

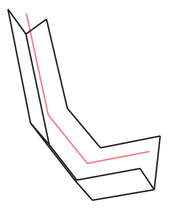


Surface created between two tangent curves.

3 control point are defining the surface.



Offset point are created perpendicular to the axis of the agent and with the direction of the branch (cross vector operation).



L shaped suface created around the line.

The degree can be changed in order to create a curved surface.

```
import processing.opengl.*;
import igeo.*;

void setup(){
  size(980, 860, IG.GL);
  IRand.init(3);
  IG.duration(33);
  //second and third vector needs to be perpendicular
  new LineAgent(new IVec(0,0,0), new IVec(1,0,0),
              new IVec(0,0,1)).clr(1.0,0,0.5);
  IG.fill();
}

static class LineAgent extends IAgent{
  static ILayer layer1 = IG.layer("Layer1");
  static ILayer layer2 = IG.layer("Layer2");
  static ILayer layer3 = IG.layer("Layer3");
  static ILayer layer4 = IG.layer("Layer4");
  static ILayer layer5 = IG.layer("Layer5");
  static ILayer layer6 = IG.layer("Layer6");
  static ILayer layer7 = IG.layer("Layer7");
  static ILayer layer8 = IG.layer("Layer8");

  static double length = 2;
  static double clearance = 1.99; //less than length
  IVec pt1, pt2, axis;
  boolean isColliding=false;

  LineAgent(IVec pt, IVec dir, IVec ax){
    pt1 = pt;
    pt2 = pt.dup().add(dir.dup().len(length));
    axis = ax;
  }
```

```
void interact(ArrayList < IDynamics > agents){
  super.interact(agents);
  if(time == 0){ //only in the first time
    for(int i=0; i < agents.size() && !isColliding; i++){
      if(agents.get(i) instanceof LineAgent){
        LineAgent lineAgent =
          (LineAgent)agents.get(i);
        if(lineAgent != this){ //agents include "this"
          //checking clearance of end point
          if(lineAgent.pt2.dist(pt2) < clearance){
            isColliding=true;
          }
        }
      }
    }
  }
}

void update(){
  super.update();

  if(isColliding){
  del(); }


  else if(time == 0){ //if not colliding

    //new ICurve(pt1,pt2); // center line

    IVec dir = pt2.diff(pt1);
    // child dir & points
    IVec nextDir1 = dir.dup().rot(axis, IRand.get(PI/4,PI/3));
```

```
IVec nextDir2 = dir.dup().rot(axis,-IRand.get(PI/4,PI/3));
IVec nextPt1 = pt2.cp(nextDir1);
IVec nextPt2 = pt2.cp(nextDir2);
//midpoints
IVec mid1 = pt1.mid(pt2);
IVec mid2 = pt2.mid(nextPt1);
IVec mid3 = pt2.mid(nextPt2);
//mid of midpoints
IVec quarter1 = pt2.mid(mid1);
IVec quarter2 = pt2.mid(mid2);
IVec quarter3 = pt2.mid(mid3);

//axis of child agents
IVec nextAxis1 =
  axis.dup().rot(nextDir1, IRand.get(-PI/4,PI/4));
IVec nextAxis2 =
  axis.dup().rot(nextDir2, IRand.get(-PI/4,PI/4));



double offsetWidth = 0;
double depth = 0;

if(IG.time()<15){
  offsetWidth = -.8;
  depth = .8;
}
else if(IG.time()<20){
  offsetWidth = -.4;
  depth = .4;
}
else if(IG.time()<30){
  offsetWidth = -.1;
  depth = .1;
}
```

```
}


if(IG.time()==15){ clr(0.5,0,0.5); }
else if(IG.time()==25){ clr(0.5,1,.5); }
else if(IG.time()==30){ clr(.5,1,.5); }


IVec offset1 = dir.cross(axis).len(offsetWidth);
IVec offset2 = nextDir1.cross(nextAxis1).len(offsetWidth);
IVec offset3 = nextDir2.cross(nextAxis2).len(offsetWidth);

IVec depthVec1 = axis.dup().len(depth);
IVec depthVec2 = nextAxis1.dup().len(depth);
IVec depthVec3 = nextAxis2.dup().len(depth);

if(depth > 0){
createEdgeSurface(mid1,quarter1,quarter2,mid2,
                  offset1, offset2,
                  depthVec1, depthVec2).clr(clr()).layer(layer1);
createEdgeSurface(mid2,quarter2,quarter3,mid3,
                  offset2.dup().flip(), offset3,
                  depthVec2, depthVec3).clr(clr()).layer(layer2);
createEdgeSurface(mid3,quarter3,quarter1,mid1,
                  offset3.dup().flip(), offset1.dup().flip(),
                  depthVec3, depthVec1).clr(clr()).layer(layer3);

createCapSurface(mid1,quarter1,mid2,quarter2,mid3,quarter3,
                 depthVec1,depthVec2,depthVec3).clr(clr()).layer(layer4);
createCapSurface(mid1,quarter1,mid3,quarter3,mid2,quarter2,
                 depthVec1.flip(),depthVec3.flip(),depthVec2.flip()).clr(clr())
                     .layer(layer5);
}
```

```
//child agents color
    int r = clr().getRed() + IRand.getInt(-10,10);
   int g = clr().getGreen()+ IRand.getInt(10,8);
    int b = clr().getBlue() + IRand.getInt(10,8);

  boolean anyChild=false;

    if(IRand.percent(100)){ //bend

    dir.dup().rot(axis,IRandom.get(PI/3,PI/3*2));
      //degree 2 curve at midpoint of pt1&pt2, pt2,
            and midpoint of pt2 and next agent's point
      new ICurve(new IVec[]{ pt1.mid(pt2),
                        pt2,
                        pt2.mid(pt2.cp(nextDir1))},
                2).clr(1.).layer(layer6);



      new LineAgent(pt2, nextDir1, nextAxis1).clr(r,g,b);
      anyChild=true;

    }
    if(IRand.percent(50)){ //bend the other way


    dir.dup().rot(axis,-IRandom.get(PI/3,PI/3*2));
      new ICurve(new IVec[]{ pt1.mid(pt2),
                        pt2,
                        pt2.mid(pt2.cp(nextDir2))},
                2).clr(1.).layer(layer7);
    new LineAgent(pt2, nextDir2, nextAxis2).clr(clr());

      anyChild=true;
```

```
if(!anyChild || IG.time()==30){
    double dist = pt2.dist(new IVec(0,0,0));
    double radius = dist*0.03;
    new ISphere(pt2,radius ).clr(1.).layer(layer8);
  }


  }
}

ISurface createEdgeSurface(IVec pt1, IVec pt2,
                           IVec pt3, IVec pt4,
                                IVec offsetDir1,
                                IVec offsetDir2,
                                IVec extrudeDir1,
                           IVec extrudeDir2){
  IVec[][] cpts = new IVec[4][4];
  cpts[0][0] = pt1.dup().add(extrudeDir1);
  cpts[1][0] = pt2.dup().add(extrudeDir1);
  cpts[2][0] = pt3.dup().add(extrudeDir2);
  cpts[3][0] = pt4.dup().add(extrudeDir2);
  cpts[0][1] = pt1.dup().add(offsetDir1).add(extrudeDir1);
  cpts[1][1] = pt2.dup().add(offsetDir1).add(extrudeDir1);
  cpts[2][1] = pt3.dup().add(offsetDir2).add(extrudeDir2);
  cpts[3][1] = pt4.dup().add(offsetDir2).add(extrudeDir2);
  cpts[0][2] = pt1.dup().add(offsetDir1).sub(extrudeDir1);
  cpts[1][2] = pt2.dup().add(offsetDir1).sub(extrudeDir1);
  cpts[2][2] = pt3.dup().add(offsetDir2).sub(extrudeDir2);
  cpts[3][2] = pt4.dup().add(offsetDir2).sub(extrudeDir2);
  cpts[0][3] = pt1.dup().sub(extrudeDir1);
  cpts[1][3] = pt2.dup().sub(extrudeDir1);
  cpts[2][3] = pt3.dup().sub(extrudeDir2);
  cpts[3][3] = pt4.dup().sub(extrudeDir2);
```

```
  return new ISurface(cpts, 3, 3);
}

ISurface createCapSurface(IVec pt1, IVec pt2,
                          IVec pt3, IVec pt4,
                          IVec pt5, IVec pt6,
                               IVec shiftDir1,
                               IVec shiftDir2,
                               IVec shiftDir3){
  IVec[][] cpts = new IVec[4][4];
  cpts[0][0] = pt1.dup().add(shiftDir1);
  cpts[1][0] = pt1.dup().add(shiftDir1);
  cpts[2][0] = pt1.dup().add(shiftDir1);
  cpts[3][0] = pt1.dup().add(shiftDir1);
  cpts[0][1] = pt2.dup().add(shiftDir1);
  cpts[1][1] = pt2.dup().add(shiftDir1);
  cpts[2][1] = pt2.dup().add(shiftDir1);
  cpts[3][1] = pt2.dup().add(shiftDir1);
  cpts[0][2] = pt4.dup().add(shiftDir2);
  cpts[1][2] = pt4.mid(pt6).add(shiftDir2.mid(shiftDir3));
  cpts[2][2] = pt4.mid(pt6).add(shiftDir2.mid(shiftDir3));
  cpts[3][2] = pt6.dup().add(shiftDir3);
  cpts[0][3] = pt3.dup().add(shiftDir2);
  cpts[1][3] = pt4.dup().add(shiftDir2);
  cpts[2][3] = pt6.dup().add(shiftDir3);
  cpts[3][3] = pt5.dup().add(shiftDir3);
  return new ISurface(cpts, 3, 3);
 }
```