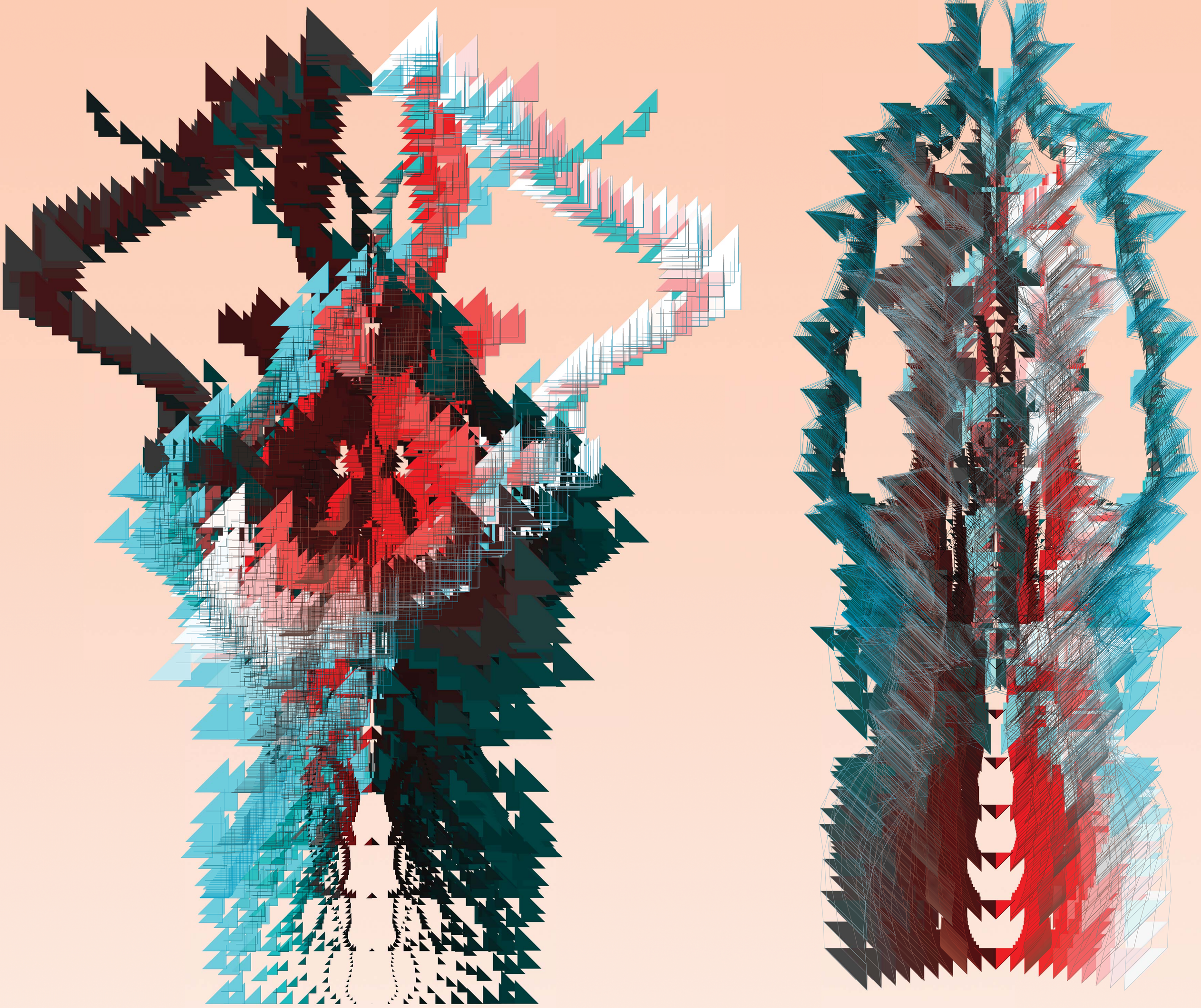
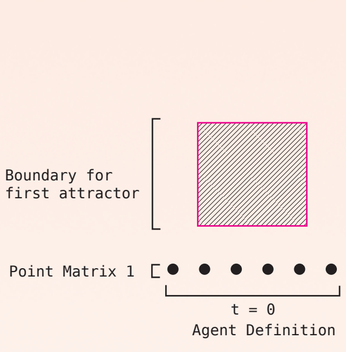
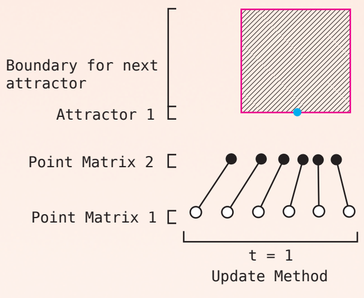
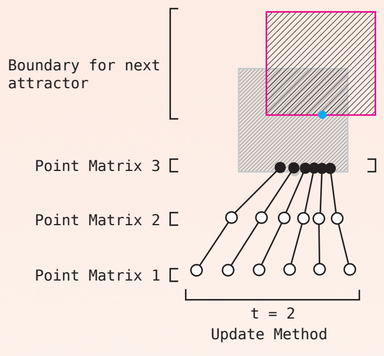
CODING FORM_FALL 2011_FINAL_DALE STRONG

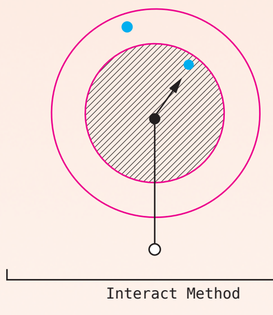


AGENT ALGORITHM



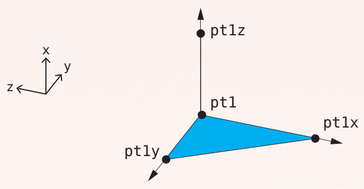




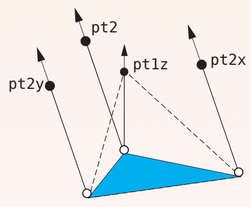


Agent checks for closest attractor and orients next vector in 3d space towards its direction.

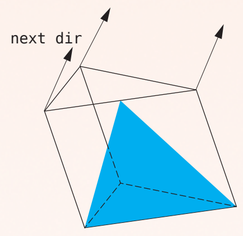
AGENT GEOMETRY



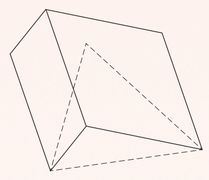
Points for Agent set-up



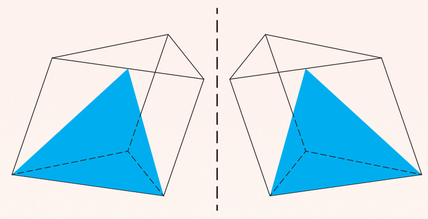
Line connects initial point to second point in growth sequence



Surface Geometry created between points



Additional Lines drawn between points



All Lines are reflected across Y axis

PROCESSING CODE

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

void setup(){
  size(640,640,IG.GL);
  IG.duration(30);

  for(int i=0; i < 3; i++){
    new MyAttractor(IRand.pt(10, 0, 20, 60, 30, 30));
  }
  //agents in a matrix
  for(int i=0; i < 10; i++){
    for(int j=0; j < 10; j++){
      new MyLineAgent(new IVec(j*10, i*3,10*tan(i+j)),
                      new IVec(0,0,i),(i+j)*.5).
                              clr(j*.08,i*.1,i*.1);
    }
  }
}
static class MyAttractor extends IAgent{
  IVec pos;
  IPoint point;

  MyAttractor(IVec p){
    pos = p;
    point = new IPoint(pos).clr(1.0,0,0);
  }
  void update(){
    // random walk
    pos.add(IRandom.pt(0,-15,-30,5,15,30));
  }
}
static class MyLineAgent extends IAgent{
  double size1;

  IVec pos, dir,att;

  MyLineAgent(IVec p, IVec d, double sz){ //add double sz
    pos = p;
    dir = d;
    size1 = sz;
    att = null;
  }

  void interact(ArrayList < IDynamics > agents){
    //searching the closest attractor
    MyAttractor closestAttractor=null;
    double minDist=-1;
    for(int i=0; i < agents.size(); i++){
      if(agents.get(i) instanceof MyAttractor){
        MyAttractor attractor = (MyAttractor)agents.get(i);
        double dist = attractor.pos.dist(pos);
        //first attractor to check
        if(minDist < 0){
          closestAttractor = attractor;
          minDist = dist;
        }
        //if less than minimum, it's new minimum
        else if(dist < minDist){
          closestAttractor = attractor;
          minDist = dist;
        }
      }
    }
    //in case no attractor found, if-condition is used
    if(closestAttractor!=null){
      IVec diff = closestAttractor.pos.diff(pos);
      diff.len(dir.len());
      dir = diff;
      att = closestAttractor.pos.dup();
    }
  }

  void update(){

    ILayer layer1 = IG.layer("line1");
    ILayer layer2 = IG.layer("line2");
    ILayer layer3 = IG.layer("line3");
    ILayer layer4 = IG.layer("line4");
    ILayer layer5 = IG.layer("line5");
    ILayer layer6 = IG.layer("line6");

    //set-up geometry
    IVec pt1 = pos.dup();
    IVec pt2 = pos.dup().add(dir);

    IVec pt2y = pt2.dup().add(0,size1,0);
    IVec pt2x = pt2.dup().add(size1,0,0);

    IVec pt1z = pos.dup().add(0,0,size1);
    IVec pt1y = pos.dup().add(0,size1,0);
    IVec pt1x = pos.dup().add(size1,0,0);

    //Set-up Curves
    IVec[] ptList1 = new IVec[2];
    ptList1[0] = pt2;
    ptList1[1] = pt1;

    IVec[] ptList2 = new IVec[2];
    ptList2[0] = pt1x;
    ptList2[1] = pt1;

    IVec[] ptList3 = new IVec[2];
      ptList3[0] = pt1y;
      ptList3[1] = pt1;

    IVec[] ptList4 = new IVec[2];
      ptList4[0] = pt1y;
      ptList4[1] = pt2y;

    IVec[] ptList5 = new IVec[2];
      ptList5[0] = pt1x;
      ptList5[1] = pt2x;

    //create curves
    ICurve crv1 = new ICurve(ptList1,1,false).clr(this.clr())
    ICurve crv2 = new ICurve(ptList2,1,false).clr(this.clr())
    ICurve crv3 = new ICurve(ptList3,1,false).clr(this.clr())
    ICurve crv4 = new ICurve(ptList4,1,false).clr(this.clr())
    ICurve crv5 = new ICurve(ptList5,1,false).clr(this.clr())

    //mirror curves
    crv1.cp().ref(new IVec(0,10,0));
    crv2.cp().ref(new IVec(0,10,0));
    crv3.cp().ref(new IVec(0,10,0));
    crv4.cp().ref(new IVec(0,10,0));
    crv5.cp().ref(new IVec(0,10,0));
    crv6.cp().ref(new IVec(0,10,0));

    //Create srf
    new ISurface(ptx,pty,ptz).clr(this.clr());
    pos.add(dir);
  }
}
```