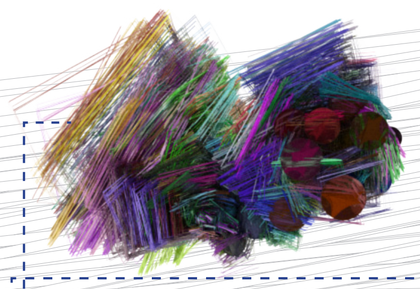
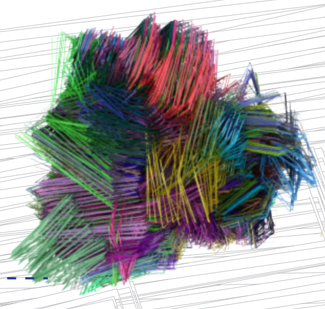
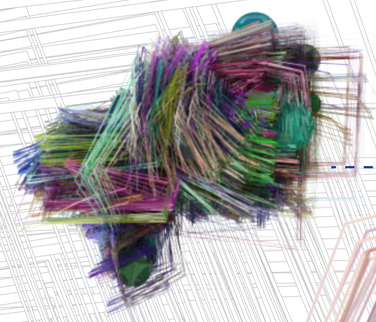


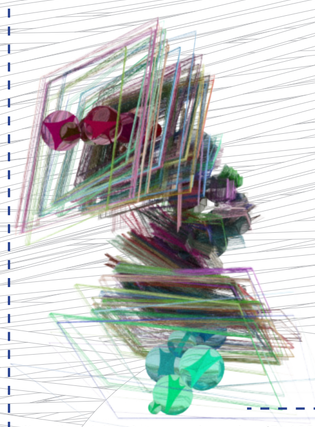
iteration 1 // # attractors = 10



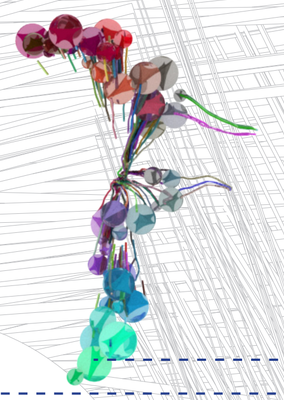
iteration 2 // additional MySphereAgent



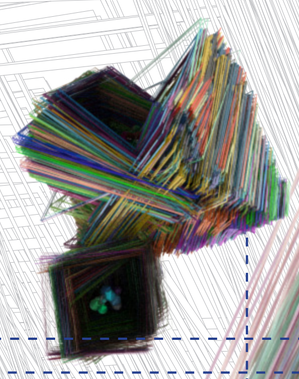
iteration 3 // time==2



iteration 4 // sphere random size {5, 100}



iteration 5 // frame set size {2}



iteration 6 // duration = 300

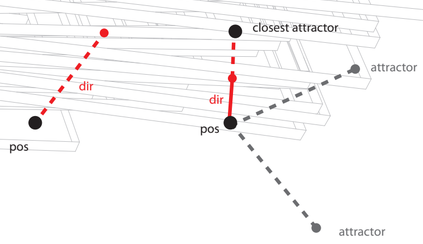
VS CODING FORM

MATTHEW TAM // FALL 2011

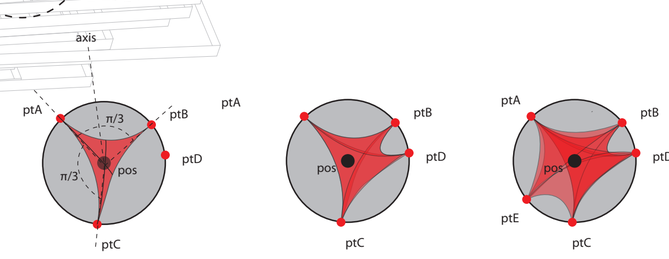
```
import processing.opengl.*;
import igeo.*;

void setup() {
  size(800, 600, IG.GL);
  IG.duration(100);
  for (int i=0; i < 5; i++) {
    new MyAttractor(IRandom.pt(-100, 100));
  }
  new MySphereAgent(IRandom.pt(-10, 10), IRandom.get(5, 20)).clr(0.5);
  new MySphereAgent(IRandom.pt(-10, 10), IRandom.get(5, 20)).clr(0.5);
  IG.transparent();
}

static class MyAttractor extends IAgent {
  static ILayer layer1 = IG.layer("attractor");
  IVec pos;
  IPoint point;

  MyAttractor(IVec p) {
    pos = p;
    point = new IPoint(pos).clr(1.0, 0, 0).layer(layer1);
  }
  void update() {
    pos.add(IRandom.pt(-5, 5)); // random walk
  }
}

static class MySphereAgent extends IAgent {
  static ILayer layer2 = IG.layer("spheres");
  static ILayer layer3 = IG.layer("surfaces");
  IVec pos, dir;
  double radius;
  ISphere sphere;
  ISurface surfa, surfb, surfc, surfd, surfe, surff, surfg, surfh, surfi, surfj;
  ISurface surfA1, surfA2, surfA3;
  boolean changed=true;
  IVec axis, axis2, axis3;
  IVec randompt1, randompt2, randompt3, randompt4, randompt5, randompt6;
  IVec ptA, ptB, ptC, ptD, ptE;

  MySphereAgent(IVec p, double rad) {
    pos = p;
    radius = rad;
    double angle = PI/3;
    axis = IRandom.pt(-1, 1);
    axis2 = IRandom.pt(-1, 1);
    randompt1 = IRandom.pt(0, 1).len(radius);
    randompt2 = randompt1.dup().rot(axis, angle);
    randompt3 = randompt1.dup().rot(axis, -angle);
    randompt4 = IRandom.pt(-1, 0).len(radius);
    randompt5 = randompt4.dup().rot(axis2, angle);
    randompt6 = randompt5.dup().rot(axis, -angle*3);
  }

  void interact(ArrayList< IDynamics > agents) {
    super.interact(agents);
    for (int i=0; i < agents.size(); i++) {
      if (agents.get(i) instanceof MySphereAgent) {
        MySphereAgent sa = (MySphereAgent)agents.get(i);
        if (sa != this) {
          double dist = sa.pos.dist(pos);
          if (dist < radius+sa.radius) {
            IVec diff = pos.diff(sa.pos);
            //overlap is this radius plus other radius minus distance between centers
            diff.len(radius+sa.radius-dist);
            pos.add(diff); //only this agent is moved, not others
            changed=true; //state variable is updated
          }
        }
      }
    }
  }

  void update() {
    super.update();

    if (changed) {
      //update sphere
      if (sphere!=null) sphere.del(); //sphere is null first
      sphere = new ISphere(pos, radius).clr(this.clr()).layer(layer2);
      synchronized(IG.lock) { // this prevents some flickering of display
        if (surfa!=null) surfa.del();
        if (surfb!=null) surfb.del();
        .....
        if (surfA3!=null) surfA3.del();
      }
      createSurfaces();
      changed=false;
    }

    if (time==3) { //delayed to create the next agent til time==3
      dir = IRandom.pt(-4, 4);  // next agent's direction
      double nextRadius = IRandom.get(5, 20);
      dir.len(radius+nextRadius);  // amount of move is the current radius + the next one
      int r = clr().getRed() + IRandom.getInt(-50, 50);
      int g = clr().getGreen() + IRandom.getInt(-50, 50);
      int b = clr().getBlue() + IRandom.getInt(-50, 50);
      new MySphereAgent(pos.cp(dir), nextRadius).clr(r, g, b);
      ptA = pos.dup().add(randompt1);
      ptB = pos.dup().sub(randompt2);
      ptC = pos.dup().sub(randompt3);
      double lineLength = 1.0;
      new MyLineAgent(ptA, randompt1.dup().len(lineLength)).clr(IRand.clr());
      new MyLineAgent(ptC, randompt2.dup().len(lineLength)).clr(IRand.clr());
      new MyLineAgent(ptD, randompt3.dup().len(lineLength)).clr(IRand.clr());
    }
  }

  void createSurfaces() {
    ptA = pos.dup().add(randompt1);
    ptB = pos.dup().sub(randompt2);
    ptC = pos.dup().sub(randompt3);
    ptD = pos.dup().sub(randompt5);
    ptE = pos.dup().sub(randompt6);
    IVec ptA1a = ptA.dup().sub(pos);
    IVec ptB1a = ptB.dup().sub(pos);
    IVec ptC1a = ptC.dup().sub(pos);
    IVec ptA1 = pos.cp(ptA1a.cross(ptB1a).len(radius/4));
    IVec ptB1 = pos.cp(ptB1a.cross(ptC1a).len(radius/4));
    IVec ptC1 = pos.cp(ptC1a.cross(ptA1a).len(radius/4));
    IVec ptA2 = pos.cp(ptA1a.cross(ptB1a).len(radius/2));
    IVec ptB2 = pos.cp(ptB1a.cross(ptC1a).len(radius/2));
    IVec ptC2 = pos.cp(ptC1a.cross(ptA1a).len(radius/2));

    // 4 STAR VOLUME - using ptA, ptB, ptC, ptF
    IVec[][] surfcptsA = new IVec [3][3];
    surfcptsA[0][0] = ptA;
    surfcptsA[0][1] = ptA;
    surfcptsA[0][2] = ptA;
    surfcptsA[1][0] = pos;
    surfcptsA[1][1] = pos;
    surfcptsA[1][2] = pos;
    surfcptsA[2][0] = ptB;
    surfcptsA[2][1] = pos;
    surfcptsA[2][2] = ptC;
    surfa = new ISurface(surfcptsA, 2, 2).clr(clr()).layer(layer3);
    .....
    IVec[][] surfcptsD = new IVec [3][3];
    surfcptsD[0][0] = ptB;
    surfcptsD[0][1] = ptB;
    surfcptsD[0][2] = ptB;
    surfcptsD[1][0] = pos;
    surfcptsD[1][1] = pos;
    surfcptsD[1][2] = pos;
    surfcptsD[2][0] = ptC;
    surfcptsD[2][1] = pos;
    surfcptsD[2][2] = ptE;
    surfd = new ISurface(surfcptsD, 2, 2).clr(clr()).layer(layer3);

    if (radius <10) {
      IVec[][] vol_cpt1 = new IVec[4][3];
      vol_cpt1[0][0] = ptA;
      vol_cpt1[0][1] = pos;
      vol_cpt1[0][2] = ptB;
      vol_cpt1[1][0] = (ptA.mid(ptA1)).mid(ptB.mid(ptB1));
      vol_cpt1[1][1] = ptB.mid(ptB1);
      vol_cpt1[1][2] = (ptB.mid(ptB1)).mid(ptC.mid(ptC1));
      vol_cpt1[2][0] = ptA1.mid(ptB1);
      vol_cpt1[2][1] = ptB1;
      vol_cpt1[2][2] = ptB1.mid(ptC1);
      vol_cpt1[3][0] = ptA2.mid(ptB2);
      vol_cpt1[3][1] = ptB2;
      vol_cpt1[3][2] = ptB2.mid(ptC2);
      surfA1 = new ISurface(vol_cpt1, 2, 2).clr(clr()).layer(layer3);
      .....
      IVec[][] vol_cpt3 = new IVec[4][3];
      vol_cpt3[0][0] = ptC;
      vol_cpt3[0][1] = pos;
      vol_cpt3[0][2] = ptA;
      vol_cpt3[1][0] = (ptC.mid(ptC1)).mid(ptA.mid(ptA1));
      vol_cpt3[1][1] = ptA.mid(ptA1);
      vol_cpt3[1][2] = (ptA.mid(ptA1)).mid(ptB.mid(ptB1));
      vol_cpt3[2][0] = ptC1.mid(ptA1);
      vol_cpt3[2][1] = ptA1;
      vol_cpt3[2][2] = ptA1.mid(ptB1);
      vol_cpt3[3][0] = ptC2.mid(ptA2);
      vol_cpt3[3][1] = ptA2;
      vol_cpt3[3][2] = ptA2.mid(ptB2);
      surfA3 = new ISurface(vol_cpt3, 2, 2).clr(clr()).layer(layer3);

      surfa.del();
    }

    else {
      //5 STAR VOLUME - using ptA, ptB, ptC, ptD, ptF
      IVec[][] surfcptsE = new IVec [3][3];
      surfcptsE[0][0] = ptA;
      surfcptsE[0][1] = ptA;
      surfcptsE[0][2] = ptA;
      surfcptsE[1][0] = pos;
      surfcptsE[1][1] = pos;
      surfcptsE[1][2] = pos;
      surfcptsE[2][0] = ptB;
      surfcptsE[2][1] = pos;
      surfcptsE[2][2] = ptD;
      surfe = new ISurface(surfcptsE, 2, 2).clr(clr()).layer(layer3);
      .....
      IVec[][] surfcptsJ = new IVec [3][3];
      surfcptsJ[0][0] = ptC;
      surfcptsJ[0][1] = ptC;
      surfcptsJ[0][2] = ptC;
      surfcptsJ[1][0] = pos;
      surfcptsJ[1][1] = pos;
      surfcptsJ[1][2] = pos;
      surfcptsJ[2][0] = ptD;
      surfcptsJ[2][1] = pos;
      surfcptsJ[2][2] = ptE;
      surfj = new ISurface(surfcptsJ, 2, 2).clr(clr()).layer(layer3);
    }
  }
}

static class MyLineAgent extends IAgent {
  static ILayer layer4 = IG.layer("squarepipes");
  IVec pos, dir;
  double piperad;

  MyLineAgent(IVec p, IVec d) {
    pos = p;
    dir = d;
  }

  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);
      piperad = diff.len();
      diff.len(dir.len());
      dir = diff;
    }
  }

  void update() {
    int r2 = clr().getRed() + time;
    int g2 = clr().getGreen() + time;
    int b2 = clr().getBlue() + time;
    IG.squarePipe(pos.dup(), pos.dup().add(dir), piperad).clr(r2, g2, b2).layer(layer4);
    pos.add(dir);
  }
}
```


MySphereAgent {definition, update method, interact method}

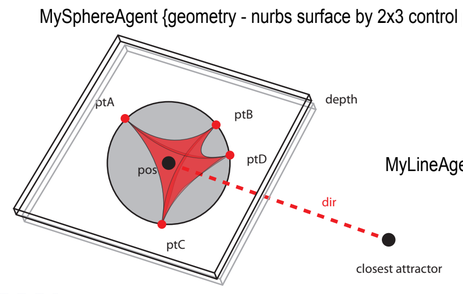
MyAttractor (definition, update method)



MyLineAgent (definition, update method, interact method)



MySphereAgent {geometry - nurbs surface by 2x3 control points}



MyLineAgent {geometry}