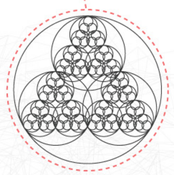
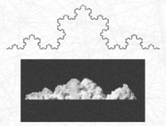
```
import processing.opengl.*;
import igeo.*;

void setup() {
  size(800, 600, IG.GL);
  IG.updateRate(0.01);
  IG.persp();
  IRand.init(50); // randomness sequence variable
  //new SpiralAgent(IG.v(0, 0, 0), IG.v(100, 0, 0), IG.v(0, 0, 0.25), 1, null);
  new SpiralGenerator();
}

class SpiralGenerator extends IAgent{
  void update(){
    if(time()==0){
      new SpiralAgent(IG.v(0, 0, 0), IG.v(100, 0, 0), IG.v(-0.02, 0, 0.25), 1, null);
    }
    else if(time()==100){
      new SpiralAgent(IG.v(400, 0, 0), IG.v(-100, 0, 0), IG.v(0.05, 0, 0.25), 1, null);
    }
    else if(time()==150){
      new SpiralAgent(IG.v(200, -300, 0), IG.v(80, 0, 0), IG.v(0, -0.05, 0.25), 1, null);
    }
     else if(time()==200){
      new SpiralAgent(IG.v(400, -400, 0), IG.v(120, 0, 0), IG.v(0, -0.05, 0.25), 1, null);
    }
  }
}


class SpiralAgent extends IAgent {
  IVec center, radius, axis, root;
  double initRadius;

  int generation;
  boolean isColliding=false;
  SpiralAgent parent;
  //ICurve curve;

  int angleDivision=60;

  ArrayList<SpiralAgent> childAgents ;

  ISurface cone;

  double scaleFactor = 1.002;
  double slopeAngle;

  IVec prevChildPos;


  SpiralAgent(IVec cnt, IVec rad, IVec axs, int gen, SpiralAgent par) {
    center = cnt;
    radius = rad;
    axis = axs;
    generation = gen;
    parent = par;
    root = center.cp();
    initRadius = radius.len();
    childAgents = new ArrayList<SpiralAgent>();

    slopeAngle = atan((float)(axis.len()/(scaleFactor-1.0)/radius.len()));

    //cone = IG.cone(root, root.cp(axis.cp().len(tan((float)slopeAngle)*radius.len()*scaleFactor)),
radius.len()*scaleFactor);
  }

  void update() {
    if(!isColliding){
      //if(curve!=null){ curve.clr(1.0, 0, 0); }
      del();
    }
    else{

      IVec pos = center.cp().add(radius);

      if (time()==0) {
        //curve = new ICurve(pos).clr(1)();
      }
      else {
        //curve.addCP(pos);
      }

      IVec axis2 = radius.cp().rot(radius.cross(axis), PI/2-slopeAngle);
      IVec radius2 = axis.cp().rot(radius.cross(axis), PI/2-slopeAngle);

      if (generation==1) {
        if (IG.time()%7 == 0 ) {
          axis2.len(axis.len()*0.2);
          radius2.len(radius.len()*0.125);
          SpiralAgent child = new SpiralAgent(pos.cp(), radius2, axis2, generation+1, this);
          child.clr(255, 140, 0);
          childAgents.add(child);
        }
      }
      else if (generation==2) {
        if (IG.time()%14 == 0 ) {
          axis2.len(axis.len()*0.2);
          radius2.len(radius.len()*0.15);

          SpiralAgent child = new SpiralAgent(pos.cp(), radius2, axis2, generation+1, this);
          child.clr(34, 139, 34);
          childAgents.add(child);

          //new ICurve(pos.cp(), axis2.cp().len(10).add(pos) ).clr(1.,1.,0);
          //new ICurve(pos.cp(), radius2.cp().len(10).add(pos) ).clr(1.,1.,0);
          if(prevChildPos != null){
            //new ICurve(pos.cp(), prevChildPos);
          }

          prevChildPos = pos.cp();
          //new IPoint(pos.cp());
          new Anchor(pos.cp(),this);
        }
      }

      else if (generation==3) {
        if (IG.time()%19 == 0) {
        axis2.len(axis.len()*0.1);
        radius2.len(radius.len()*0.1);
        //SpiralAgent child = new SpiralAgent(pos.cp(), radius2, axis2, generation+1, this);
        //child.clr(191, 171, 255);
        //childAgents.add(child);
          if(prevChildPos != null){
            //new ICurve(pos.cp(), prevChildPos);
          }

          prevChildPos = pos.cp();
          //new IPoint(pos.cp());
          new Anchor(pos.cp(),this);
        }
      }


      if(!isColliding) scale(root, scaleFactor);

      center.add(axis);
      radius.rot(axis, 2*PI/angleDivision);
    }
  }

  void scale(IVec cnt, double factor) {
    root.scale(cnt, factor);
    center.scale(cnt, factor);
    //if (curve!=null) {
    // curve.scale(cnt, factor);
    //}
    for (int i=0; i<childAgents.size(); i++) {
      childAgents.get(i).scale(cnt, factor);
    }
    initRadius *= factor;
    //cone.scale(cnt, factor);
    //cone.updateGraphics();
  }

  void interact(ArrayList<IDynamics> agents) {
    isColliding=false;
    for (int i=0; i<agents.size() && !isColliding; i++) {
      if (agents.get(i) instanceof SpiralAgent) {
        SpiralAgent a = (SpiralAgent)agents.get(i);
        if (a != this && a.generation==generation) {
          if (a.root.dist(root) < initRadius + a.initRadius) {
            isColliding = true;
            IG.p("colliding");

            new IPoint(a.root.mid(root)).clr(1.,0,1);
          }
        }
      }
    }
  }
}

class Anchor extends IAgent{
  IVec pos;
  IPoint point;
  SpiralAgent root;
  Anchor(IVec p, SpiralAgent s){
    pos = p;
    //point = new IPoint(pos);
    root = s;
    while(root.parent!=null){ root = root.parent; }
  }

  void interact(ArrayList<IDynamics> agents) {
    if(time()==0){
    for (int i=0; i<agents.size() ; i++) {
      if (agents.get(i) instanceof Anchor) {
        Anchor a = (Anchor)agents.get(i);
        if (a != this ) {
          if(a.root == root){
            if (a.pos.dist(pos) < 25 && a.pos.dist(pos) > 10 && IRand.pct(20)) {
              //new ICurve(pos, a.pos).clr(1.,0,1.0.2);
            }
          }
          else{
            if (a.pos.dist(pos) < 70 && a.pos.dist(pos) > 40 && IRand.pct(1)) {
              //new ICurve(pos, a.pos).clr(1.,0,1.0.2);
            }
          }

          if (IRand.pct(0.005)) {
            if(a.root == root){ // in a same broccori
            // new ICurve(pos, a.pos).clr(1..0,1,0.2);
            }
          }
        }
      }
    }
  }

  void update(){
    if(time()==60){// delete later to run faster
      del();
      //point.del();
    }
  }
}
```



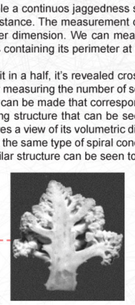
Definition and organization of the "**fractal broccoli's form**" (special organization of sub peaks - red circles): spirally arranged segments which are identical copies of the whole base unit (flower).

The Broccoli Romanesco exhibits **fractal structure** in its perimeter, planar and volumetric dimensions. If we look at the perimeter of the Broccoli Romanesco we can see an outline that bares a 'rough' resemblance to the Koch curve.

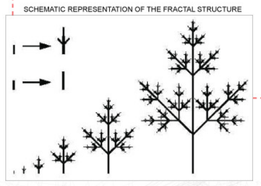


It's visible a continuos jaggedness seen at different levels of magnification even at a close distance. The measurement of the jaggedness of this irregular boundary is its perimeter dimension. We can measure its dimension by comparing the number of squares containing its perimeter at different grid sizes.

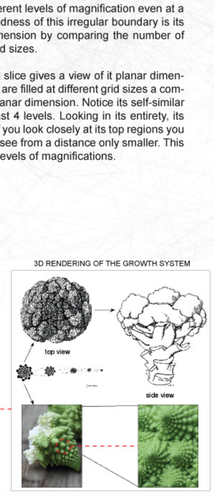
Cutting it in a half, it's revealed cross sectional slice gives a view of it planar dimension. By measuring the number of squares that are filled at different grid sizes a comparison can be made that corresponds to its planar dimension. Notice its self-similar branching structure that can be seen to at least 4 levels. Looking in its entirety, its form gives a view of its volumetric dimension. If you look closely at its top regions you can see the same type of spiral cones that you see from a distance only smaller. This self similar structure can be seen to at least 3 levels of magnifications.



SCHEMATIC REPRESENTATION OF THE FRACTAL STRUCTURE

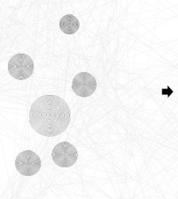


3D RENDERING OF THE GROWTH SYSTEM



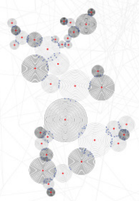
from **SINGULARITY** to **MULTIPLICITY**

1 - Horizontal development of the structure examined (Broccoli Romanesco) with the identification of the original cores for the **development of the system of growth** (spiral line black): Fibonacci sequence repetition.



2 - Definition and development of add-ons starting from primary forms (**decentralized cores** for the formation of new ramifications).

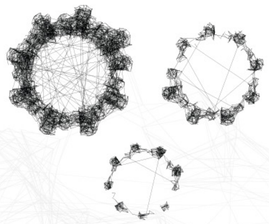
3 - Recognition and stabilization of focal center points (red dots) and the resulting from the collision rules (blue dots): **CONNECTIVITY DIAGRAM** and **GENERATIVE PATH SYSTEM**.

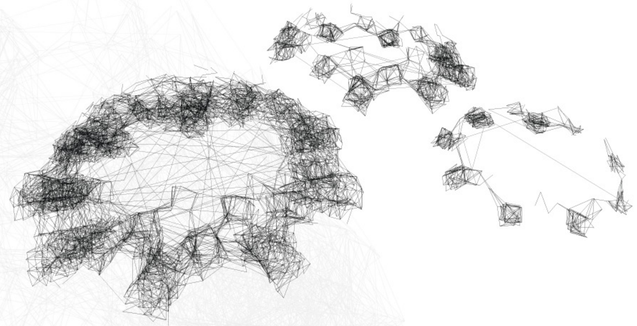


4 - Final **horizontal performing**, branching agents with a collision detection algorithm using intersection of line segments, with the creation and propagation of aggregative nucleus as a function of the collisions.

ADVANCED CODING FORM
Spring 2013

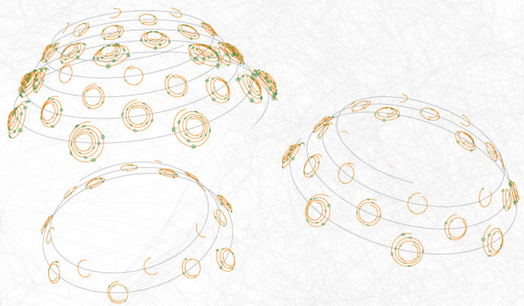
Satoru Sugihara
Federico Pessani

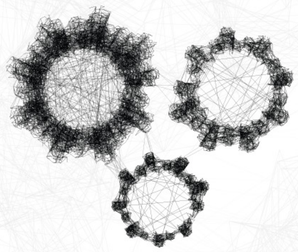


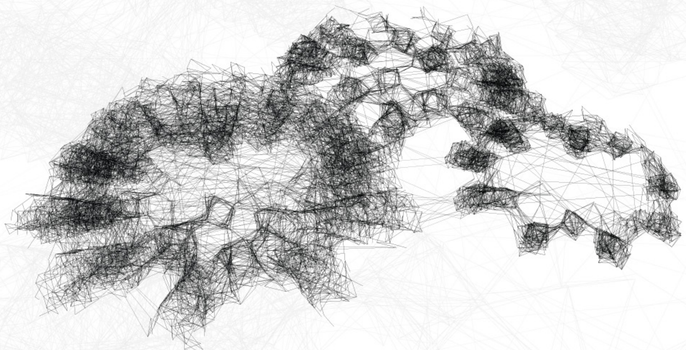


PHASE **1**

- definition of the spirals growth's origin (grey lines)
- beginning with the first generation (orange lines)
- formation of the boundary geometry related to the first generation

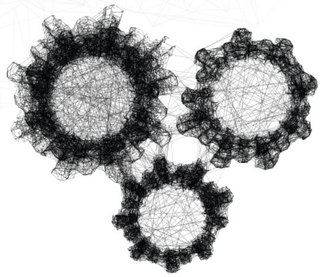


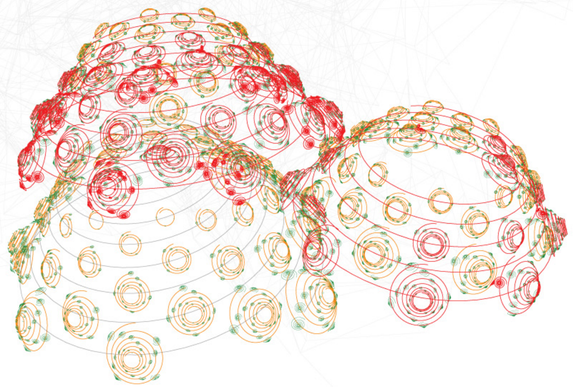




PHASE **2**

- origin and development of the second generation of growth from the first one (green lines)
- formation of peripheral cores
- development internal geometry resulting in the perimeter





PHASE **3**

- origin and definition of the various ratios of collision with consequent arrest of growth (red lines)
- ultimate definition of the geometry with peripheral bonds within the limits of the predefined distances