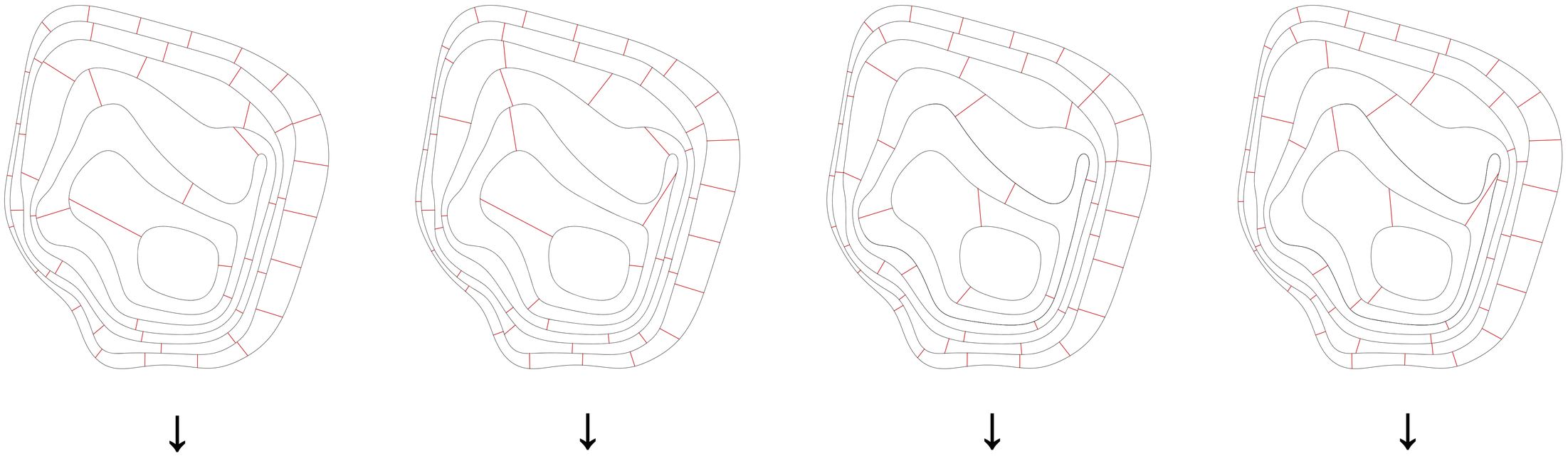


A B C D

C

*L-System Basic Rule

n=1; A → 1

n=2; AB → 2

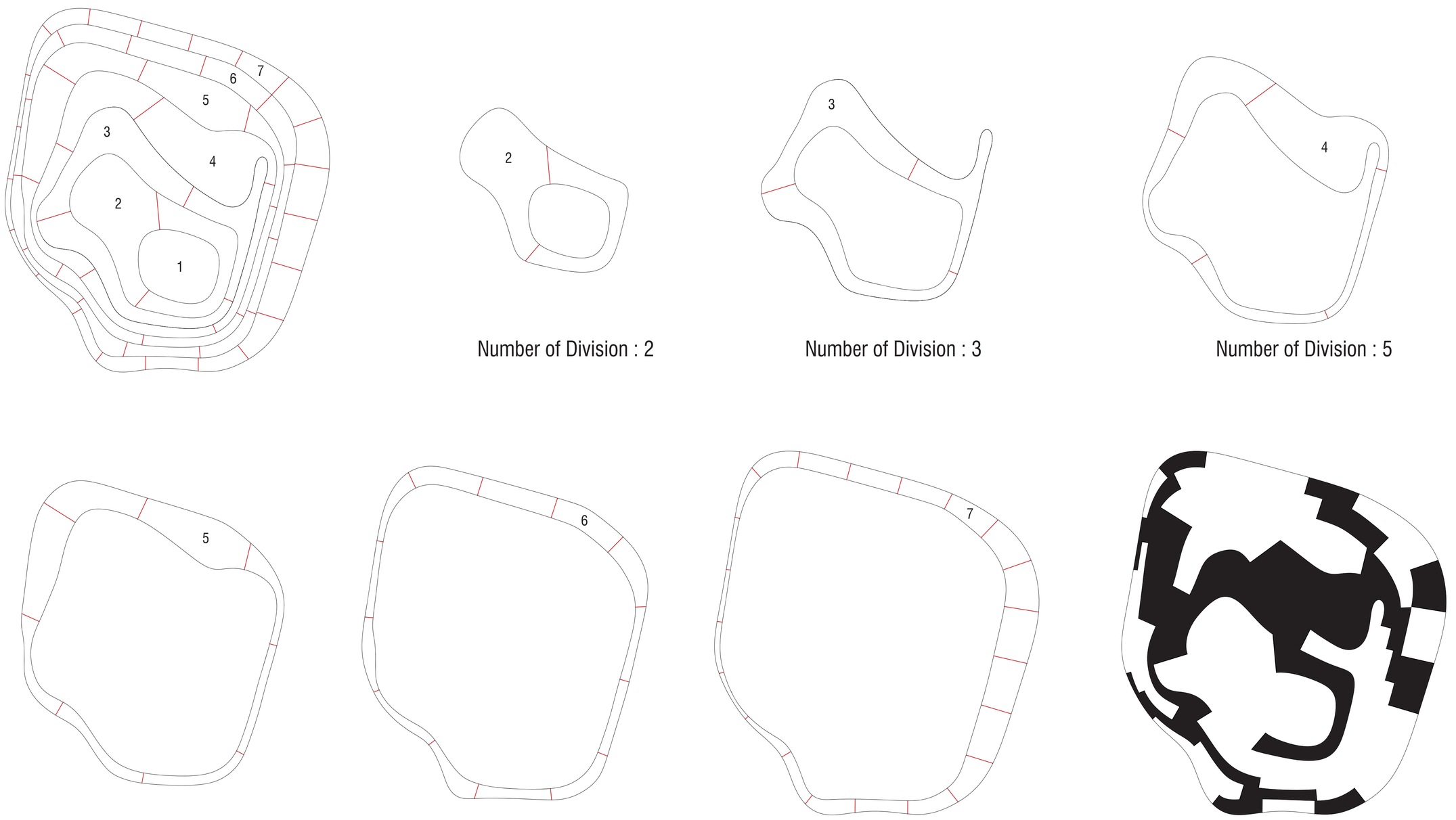
n=3; ABA → 3

n=4; ABAAB → 5

n=5; ABAABABA → 8

n=6; ABAABABAABAAB → 13

n=7; ABAABABAABAABABAABABA → 21



Number of Division : 2

Number of Division : 3

Number of Division : 5

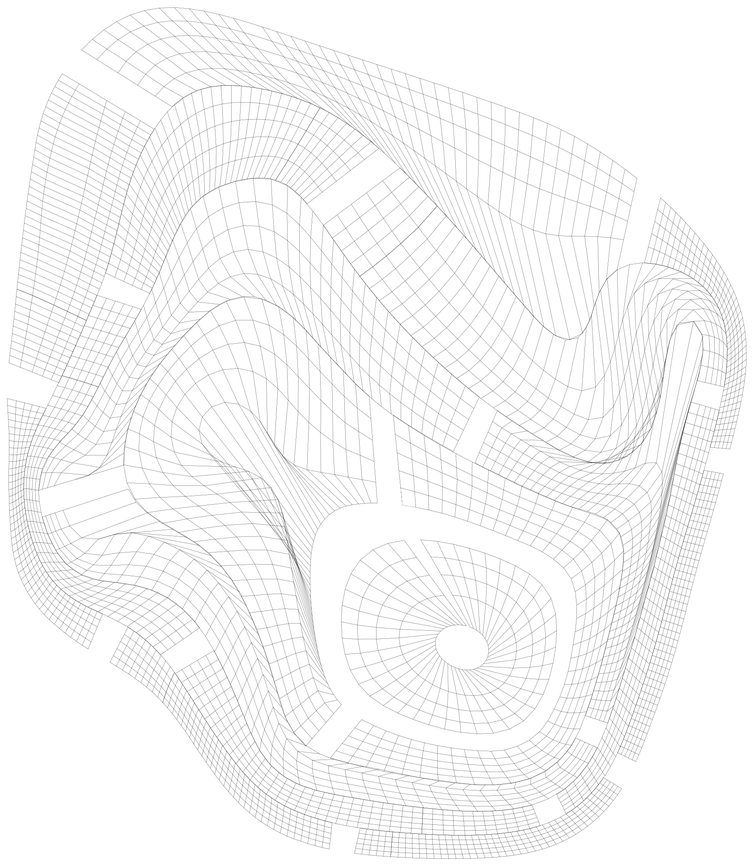
Number of Division : 8

Number of Division : 13

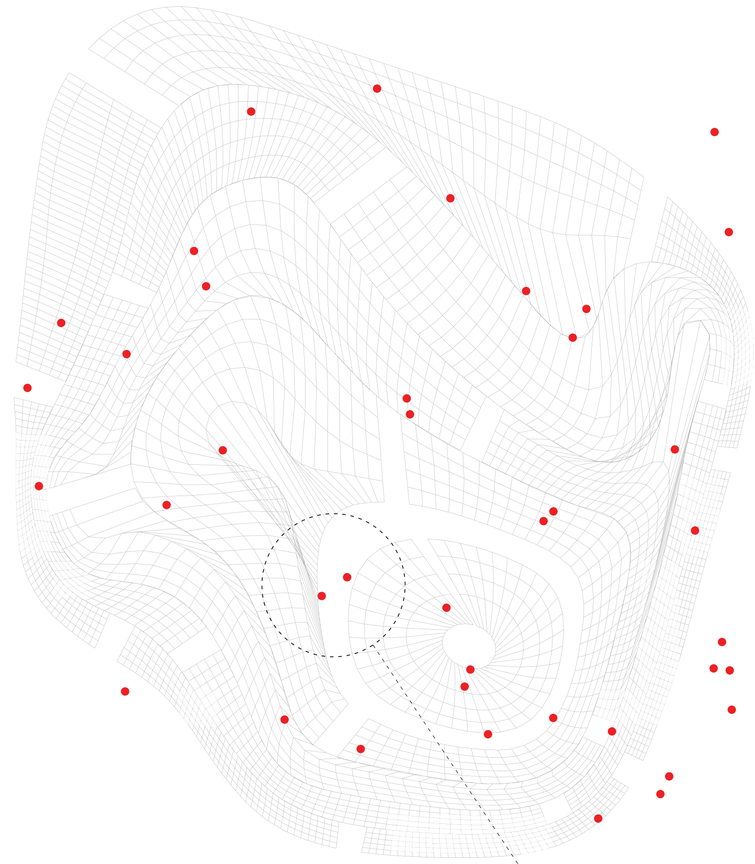
Number of Division : 21

Part2: IGEO Computation

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

void setup(){

  size(800,800,IG.GL);
  IG.open("Contour_01_032212.3dm");
  IG.duration(500);

  ICurve[] lineA = IG.layer("A").curves();
  IPoint[] pointA = IG.layer("A").points();


  ICurve[] lineB = IG.layer("B").curves();
  IPoint[] pointB = IG.layer("B").points();


  ITensileNet.particleClass(ParticleA.class);
  ITensileNet.tensionClass(MyTensionA.class);
  ITensileNet.create(lineA, pointA);

  ITensileNet.particleClass(ParticleB.class);
  ITensileNet.tensionClass(MyTensionB.class);
  ITensileNet.create(lineB, pointB);


  new MyGravityA(IG.v(0,0,0.4));

  new MyGravityB(IG.v(0,0,-0.4));

  for(int i=0; i < 40; i++){
  new RepulsionAgent(IRand.pt(-60,-60,-5,60,60,5)).clr(1.,0,0);
  }



}


class ParticleA extends IParticleAgent{
  double thresholdB = 0;
  double repulsionB = 0;

}



class ParticleB extends IParticleAgent{
  double thresholdA = 0;
  double repulsionA = 0;

}



class MyTensionA extends ITensionLine{

  MyTensionA(IParticleAgent p1, IParticleAgent p2){
    super(p1,p2);
  }

  void update(){
    if(IG.time()%10==0){
      IVec p1 = pos1().cp();
      IVec p2 = pos2().cp();
      new ICurve(p1, p2).clr(IG.time()*0.002);
    }
  }
}

class MyTensionB extends ITensionLine{

  MyTensionB(IParticleAgent p1, IParticleAgent p2){
    super(p1,p2);
  }

  void update(){
    if(IG.time()%10==0){
      IVec p1 = pos1().cp();
      IVec p2 = pos2().cp();
      new ICurve(p1, p2).clr(IG.time()*0.002);
    }
  }
}



class MyGravityA extends IAgent{
  IVec gravity;
  MyGravityA(IVec g){ gravity=g; }
  void interact(IDynamics agent){
    if(agent instanceof ParticleA){
      IParticleAgent particleA = (ParticleA)agent;
      particleA.push(gravity);
    }
  }
}


class MyGravityB extends IAgent{
  IVec gravity;
  MyGravityB(IVec g){ gravity=g; }
  void interact(IDynamics agent){
    if(agent instanceof ParticleB){
      IParticleAgent ParticleB = (ParticleB)agent;
      ParticleB.push(gravity);
    }
  }
}


  class RepulsionAgent extends IPointAgent{
  double threshold = 10.0;
  double minDist = 1.0;
  double repulsion = 3.0;

  RepulsionAgent(IVec v){ super(v); }

  void interact(IDynamics agent){
    if(agent instanceof IParticleAgent){
      IParticleAgent particle = (IParticleAgent)agent;
      IVec dif = particle.pos().dif(pos());
      double dist = dif.len();
      if(dist < threshold){
        if(dist < minDist){ dist = minDist; }
        double strength = repulsion/dist;
        IVec force = dif.len(strength);
        particle.push(force);
      }
    }
  }
}
```

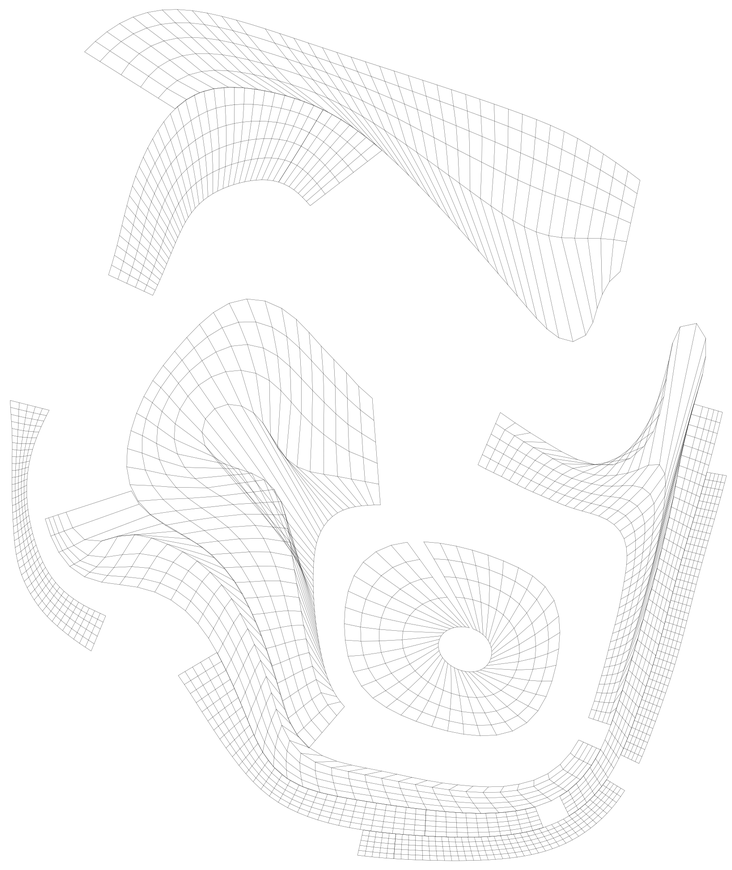


Initial Condition / A+B

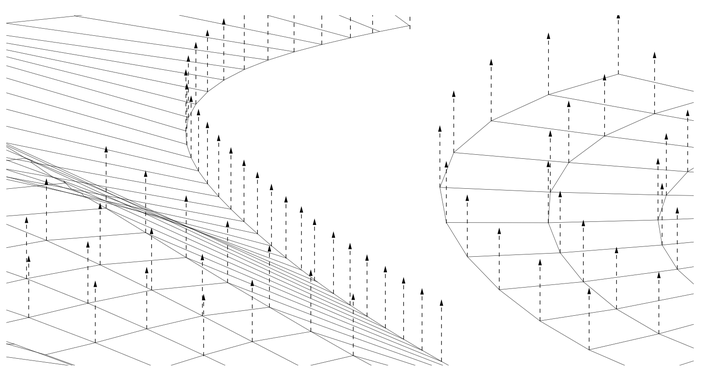


Repulsion Agents Distributed

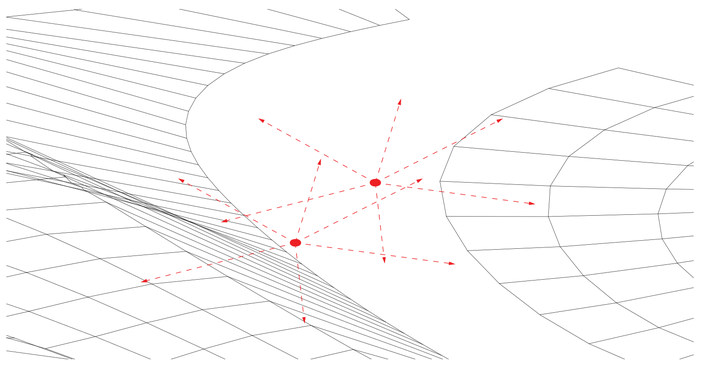
1



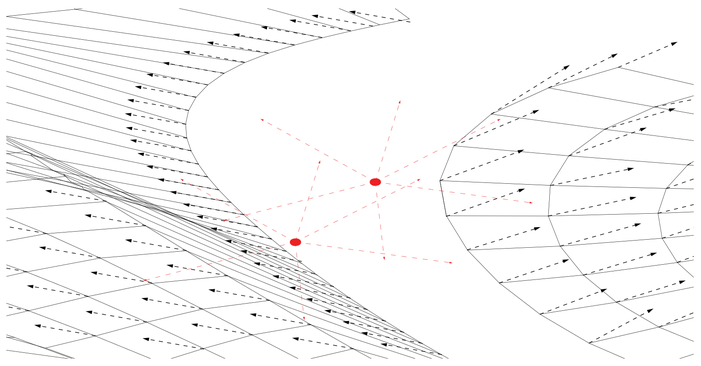
Initial Condition / A



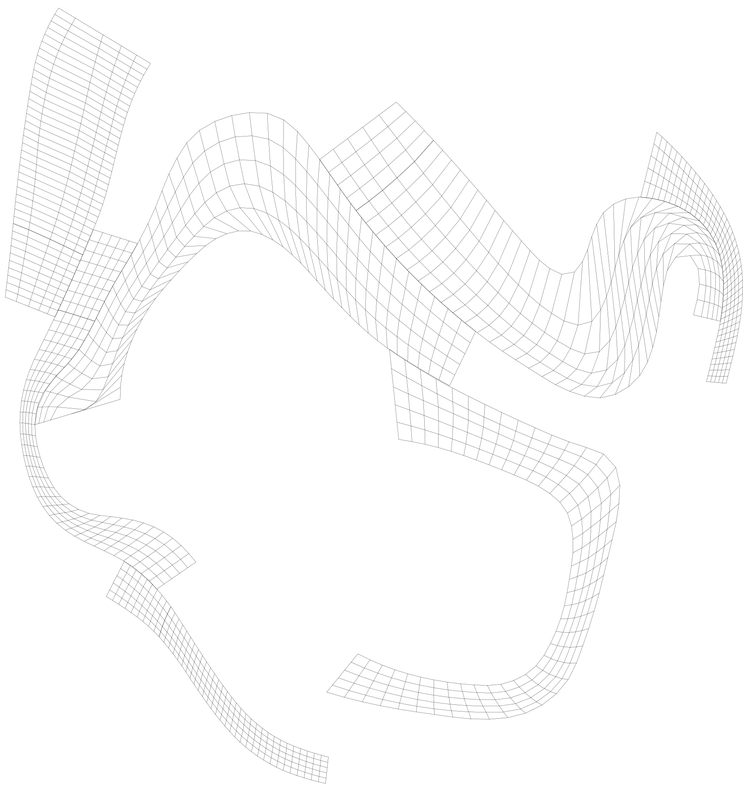
Original Force of Gravity / Applied only in Z-Axis



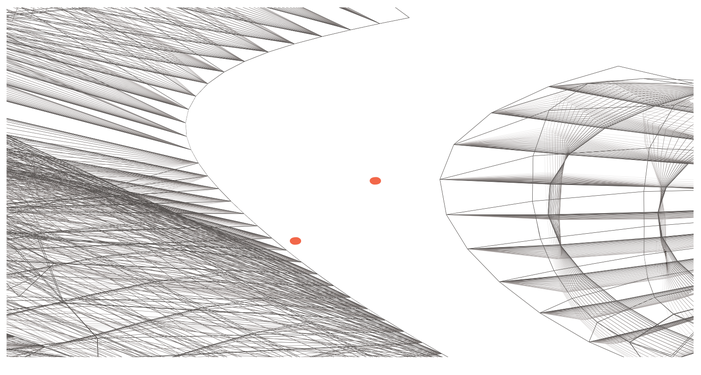
Repulson Agents Distributed



Combination of Repulson Agents and Gravity Force Applied



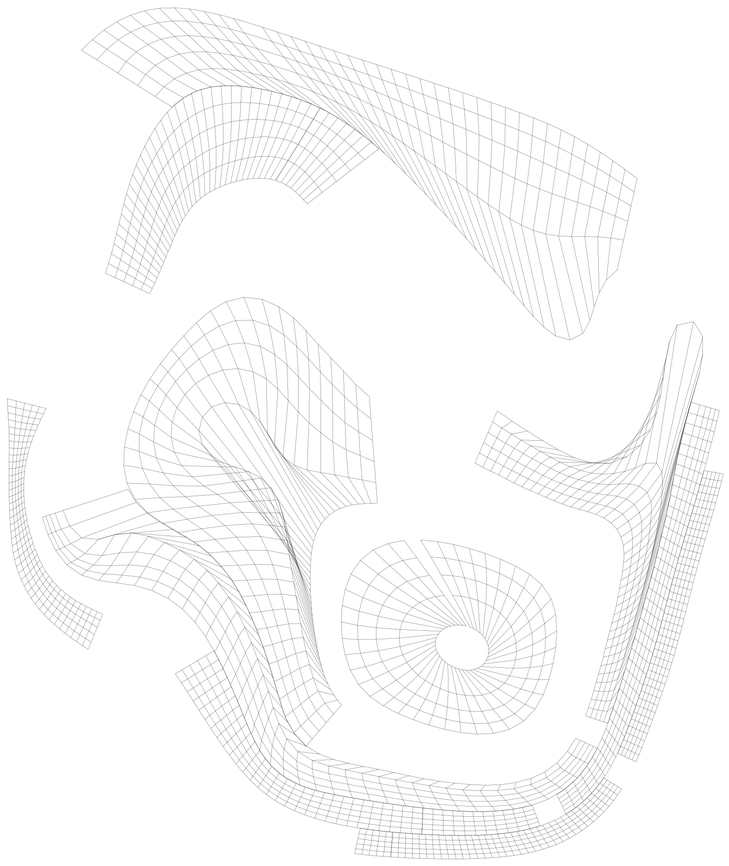
Initial Condition / B



1

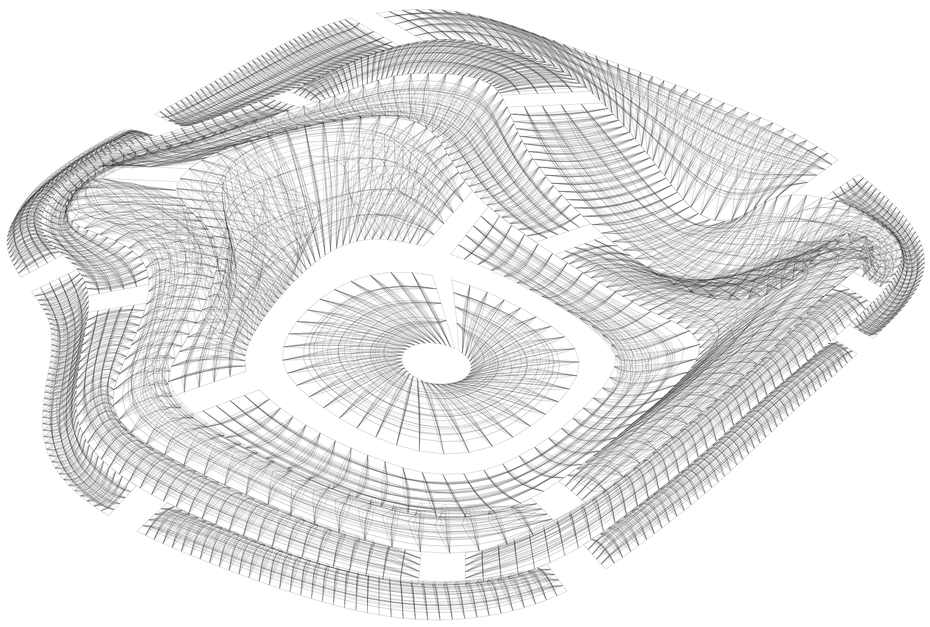
Part2: IGEO Computation

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

void setup(){

  size(800,800,IG.GL);
  IG.open("Contour_01_032212.3dm");
  IG.duration(500);

  ICurve[] lineA = IG.layer("A").curves();
  IPoint[] pointA = IG.layer("A").points();


  ICurve[] lineB = IG.layer("B").curves();
  IPoint[] pointB = IG.layer("B").points();


  ITensileNet.particleClass(ParticleA.class);
  ITensileNet.tensionClass(MyTensionA.class);
  ITensileNet.create(lineA, pointA);

  ITensileNet.particleClass(ParticleB.class);
  ITensileNet.tensionClass(MyTensionB.class);
  ITensileNet.create(lineB, pointB);


  new MyGravityA(IG.v(0,0,0.4));

  new MyGravityB(IG.v(0,0,-0.4));

  for(int i=0; i < 40; i++){
  new RepulsionAgent(IRand.pt(-60,-60,-5,60,60,5)).clr(1.,0,0);
  }

}


class ParticleA extends IParticleAgent{
  double thresholdB = 0;
  double repulsionB = 0;

}



class ParticleB extends IParticleAgent{
  double thresholdA = 0;
  double repulsionA = 0;

}


class MyTensionA extends ITensionLine{

  MyTensionA(IParticleAgent p1, IParticleAgent p2){
    super(p1,p2);
  }

  void update(){
    if(IG.time()%10==0){
      IVec p1 = pos1().cp();
      IVec p2 = pos2().cp();
      new ICurve(p1, p2).clr(IG.time()*0.002);
    }
  }
}

class MyTensionB extends ITensionLine{

  MyTensionB(IParticleAgent p1, IParticleAgent p2){
    super(p1,p2);
  }

  void update(){
    if(IG.time()%10==0){
      IVec p1 = pos1().cp();
      IVec p2 = pos2().cp();
      new ICurve(p1, p2).clr(IG.time()*0.002);
    }
  }
}


class MyGravityA extends IAgent{
  IVec gravity;
  MyGravityA(IVec g){ gravity=g; }
  void interact(IDynamics agent){
    if(agent instanceof ParticleA){
      IParticleAgent particleA = (ParticleA)agent;
      particleA.push(gravity);
    }
  }
}


class MyGravityB extends IAgent{
  IVec gravity;
  MyGravityB(IVec g){ gravity=g; }
  void interact(IDynamics agent){
    if(agent instanceof ParticleB){
      IParticleAgent ParticleB = (ParticleB)agent;
      ParticleB.push(gravity);
    }
  }
}


  class RepulsionAgent extends IPointAgent{
  double threshold = 10.0;
  double minDist = 1.0;
  double repulsion = 3.0;

  RepulsionAgent(IVec v){ super(v); }

  void interact(IDynamics agent){
    if(agent instanceof IParticleAgent){
      IParticleAgent particle = (IParticleAgent)agent;
      IVec dif = particle.pos().dif(pos());
      double dist = dif.len();
      if(dist < threshold){
        if(dist < minDist){ dist = minDist; }
        double strength = repulsion/dist;
        IVec force = dif.len(strength);
        particle.push(force);
      }
    }
  }
}
```

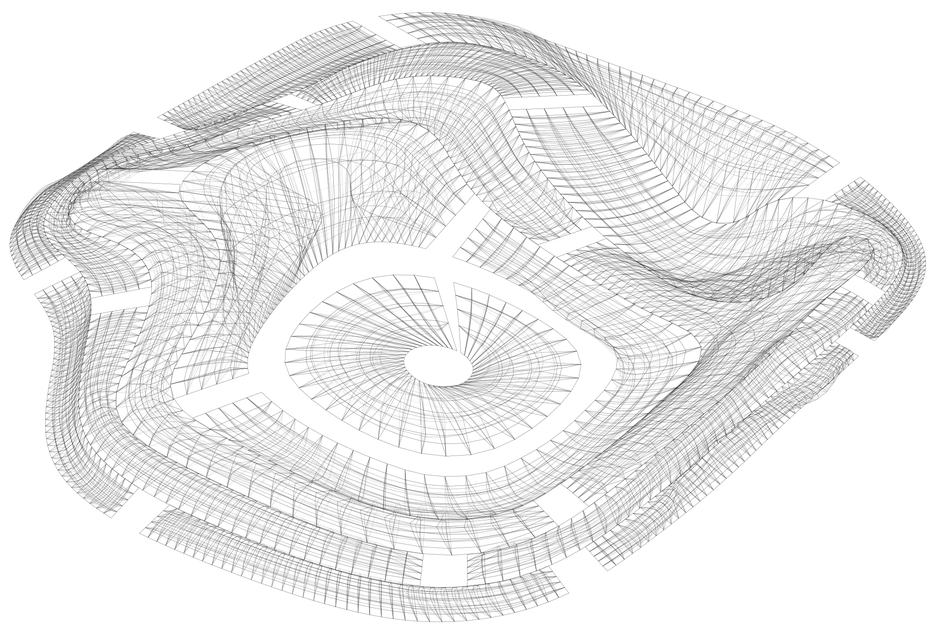


Gravity Force Applied +0.4 in Z-Axis

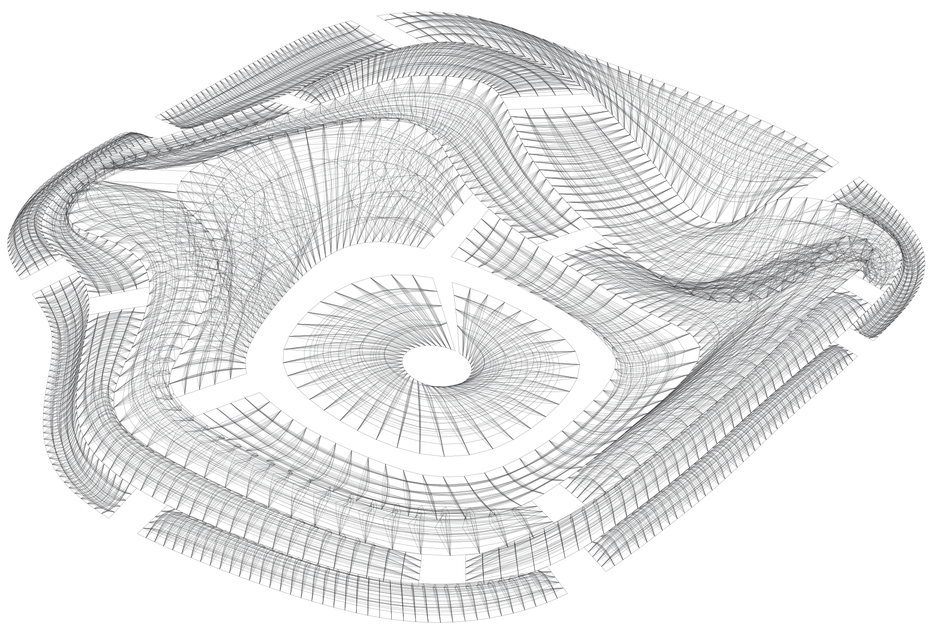
Gravity Force Applied -0.4 in Z-Axis



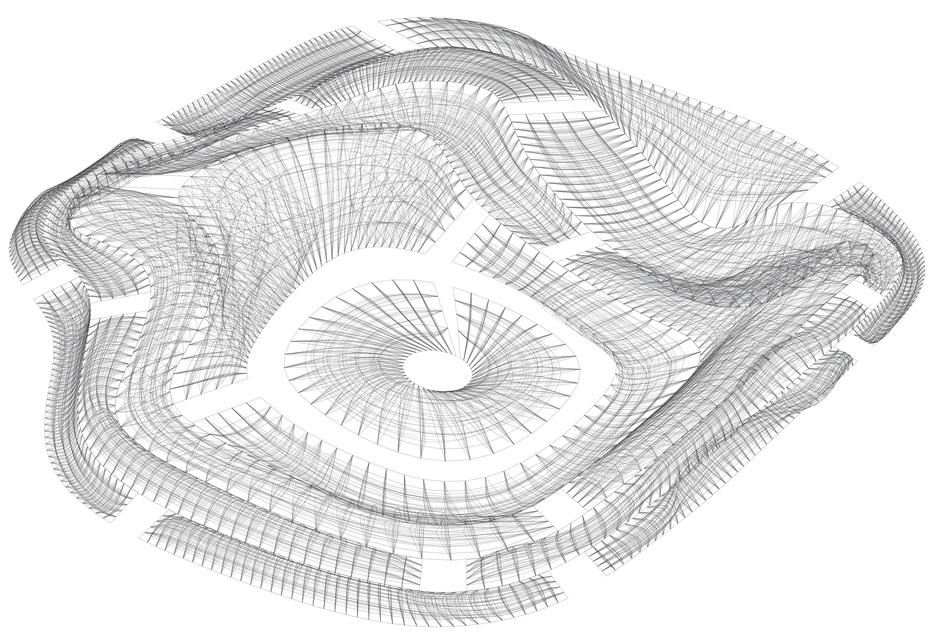
Duration 250 without Repulsion



Duration 250 with Repulsion



Duration 500 without Repulsion



Duration 500 with Repulsion