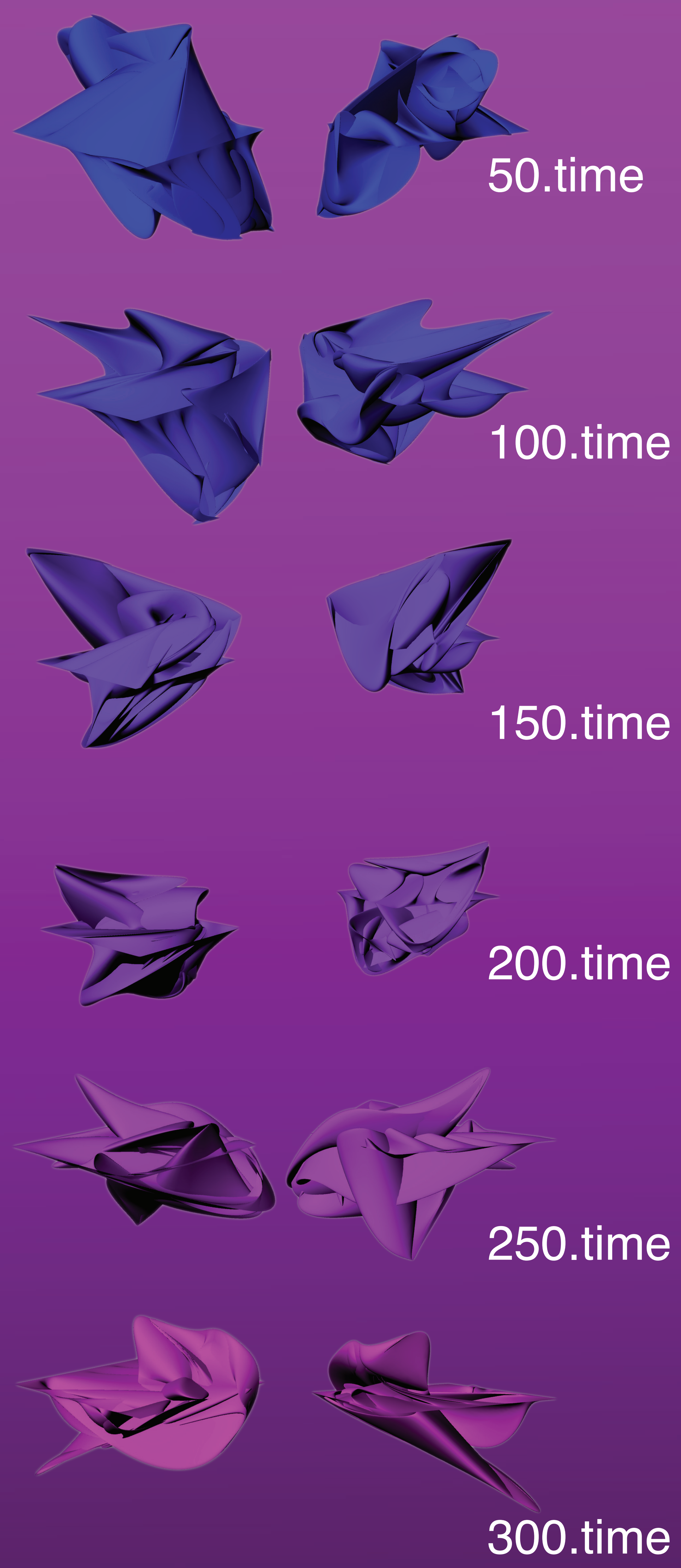

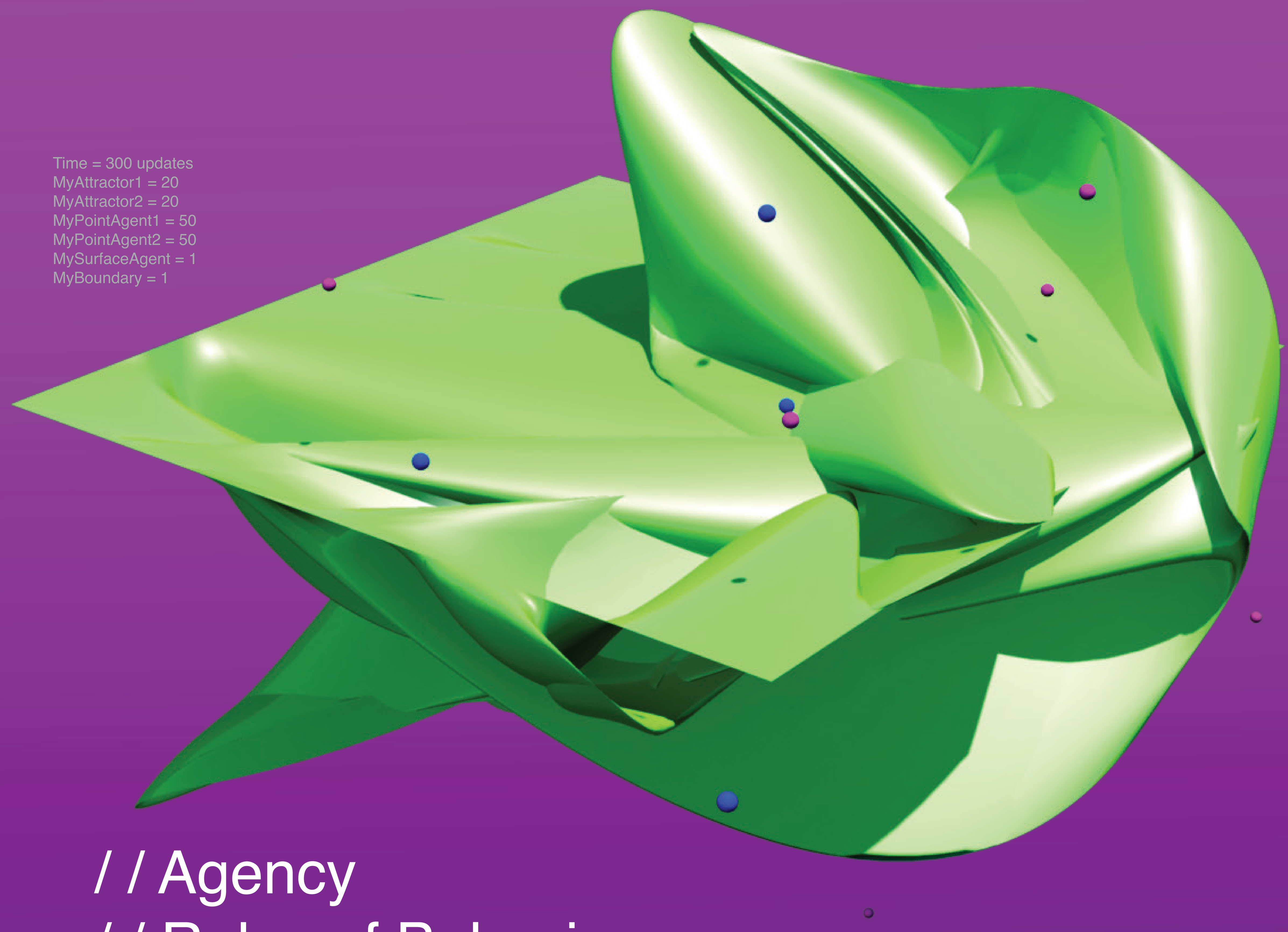

/ / Agency
/ / Rules of Behavior
/ / Interaction
/ / Time

+Attractor
+Follower
+Boundary
+Geometry

Modify a Geometry with the use of different types of agents.

Define rules of behavior and interaction for diferent type of agents.

Archieve control by defining values directly related to the agents behavior rules.

Point agents move and try to reach the Attractor Agents through an Interact method by adding a direction vector to their position vector. Having two different AttractorAgents and PointAgents allow a crossed and different behavior. On the other hand the AttractorAgents move through the space by using a random vector as direction.

The SurfaceAgent calls an array of PointAgents to create geometry generating a grid with a checkers like organization of two different PointAgents. At the same time the PointAgents on the edge of the surface are fixed by using a Boolean Operation.

The behavior of the PointAgents as attracted agents and at the same time control points on a surface, modify the SurfaceAgent by deforming it by changing their position due to the attractorAgents. In the meantime in order to control the size and deformation of the SurfaceAgent, the BoundaryAgent limits the field of movement for the AttractorAgent by reflecting its direction vector when reaching the boundaries.

Libraries
Open GL+IGeo

Void SetUp

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

void setup() {
  size(1280, 820, IG.GL);
  IG.duration(300);
  IRand.init(2);


  new MyBoundary(-18,-18,-36,54,54,36);

  for (int i=0; i < 20; i++) {
    //new MyAttractor3(IRand.pt(0, 0, 0, 10, 10, 20));
    new MyAttractor3(new IVec (17.9,17.9,0));
  }
  for (int i=0; i < 20; i++) {
    //new MyAttractor2(IRand.pt(0, 0, 0, 10, 10, 20));
    new MyAttractor2(new IVec (17.9,17.9,0));
  }
  //agents in a matrix
  int unum=10;
  int vnum=10;
  MyPointAgent[][] agents = new MyPointAgent[unum][vnum];
  for(int i=0; i<unum; i++){
    for(int j=0; j<vnum; j++){
      if((i+j)%2==0){
        agents[i][j] = new MyPointAgent3(new IVec(i*4, j*4, 0),
        new IVec(0, 0, IRandom.get(1.,1.1)));
      }
      else{
         agents[i][j] = new MyPointAgent2(new IVec(i*4, j*4, 0),
         new IVec(0, 0, IRandom.get(1.,1.1)));
      }
      if(i==0) agents[i][j].fixed=true;
      if(j==0) agents[i][j].fixed=true;
      if(i==9) agents[i][j].fixed=true;
      if(j==9) agents[i][j].fixed=true;
      if(i==4&&j==4) agents[i][j].fixed=true;
      if(i==5&&j==4) agents[i][j].fixed=true;
      if(i==4&&j==5) agents[i][j].fixed=true;
      if(i==5&&j==5) agents[i][j].fixed=true;
    }
  }
  new MySurfaceAgent (agents);
}
```

Boundary Agent

```
class MyBoundary extends IAgent {
  double minx, maxx, miny, maxy, minz, maxz;
  MyBoundary(double x1, double y1, double z1,
    double x2, double y2, double z2) {
      minx = x1;   miny = y1;   minz = z1;
      maxx = x2;  maxy = y2;  maxz = z2;
  }
}
static class MyAttractor extends IAgent {
  IVec pos;
  IVec dir;
  IPoint point;

  MyAttractor(IVec p) {
      pos = p;
      dir = new IVec(0,0,0);
      point = new IPoint(pos).clr(1.0, 0, 0);
  }
  void interact( ArrayList < IDynamics > agents ) {
      for (int i=0; i < agents.size(); i++) {
        if (agents.get(i) instanceof MyBoundary) {
          MyBoundary boundary = (MyBoundary)agents.get(i);
          //checking if next position is out of the boundary
          IVec nextPos = pos.cp(dir);
          if (nextPos.x < boundary.minx) {
            dir.ref(IG.xaxis); //reflect on x-plane
          }
          else if (nextPos.x > boundary.maxx) {
            dir.ref(IG.xaxis); //reflect on x-plane
          }
          if (nextPos.y < boundary.miny) {
            dir.ref(IG.yaxis); //reflect on y-plane
          }
          else if (nextPos.y > boundary.maxy) {
            dir.ref(IG.yaxis); //reflect on y-plane
          }
          if (nextPos.z < boundary.minz) {
            dir.ref(IG.zaxis); //reflect on z-plane
          }
          else if (nextPos.z > boundary.maxz) {
            dir.ref(IG.zaxis); //reflect on z-plane
          }
        }
      }
    }
   void update() {
    // random walk
      dir.add(IRandom.pt(-.2, -.2, -.2, .2, .2, .2));
      pos.add(dir);
  }
}
```

Attractor Agent

```
static class MyAttractor3 extends IAgent {
  IVec pos;
  IVec dir;
  IPoint point;

  MyAttractor3(IVec p) {
      pos = p;
      dir = new IVec(0,0,0);
      point = new IPoint(pos).clr(0, 0, 1.);
  }

  void interact( ArrayList < IDynamics > agents ) {
      for (int i=0; i < agents.size(); i++) {
        if (agents.get(i) instanceof MyBoundary) {
          MyBoundary boundary = (MyBoundary)agents.get(i);
          //checking if next position is out of the boundary
          IVec nextPos = pos.cp(dir);
          if (nextPos.x < boundary.minx) {
            dir.ref(IG.xaxis); //reflect on x-plane
          }
          else if (nextPos.x > boundary.maxx) {
            dir.ref(IG.xaxis); //reflect on x-plane
          }
          if (nextPos.y < boundary.miny) {
            dir.ref(IG.yaxis); //reflect on y-plane
          }
          else if (nextPos.y > boundary.maxy) {
            dir.ref(IG.yaxis); //reflect on y-plane
          }
          if (nextPos.z < boundary.minz) {
            dir.ref(IG.zaxis); //reflect on z-plane
          }
          else if (nextPos.z > boundary.maxz) {
            dir.ref(IG.zaxis); //reflect on z-plane
          }
        }
      }
    }


  void update() {
    // random walk
      dir.add(IRandom.pt(-.2, -.2, -.2, .2, .2, .2));
      pos.add(dir);
  }
}
```

Control Points Agent

```
static class MyPointAgent2 extends MyPointAgent {

  MyPointAgent2(IVec p, IVec d) {
    super(p,d);
  }

  void interact(ArrayList < IDynamics > agents) {
    //searching the closest attractor
    MyAttractor2 closestAttractor=null;
    double minDist=-1;
    for (int i=0; i < agents.size(); i++) {
      if (agents.get(i) instanceof MyAttractor2) {
        MyAttractor2 attractor = (MyAttractor2)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;
    }
  }
}
```

Surface Agent

```
static class MySurfaceAgent extends IAgent {
  //IVec pos1;
  MyPointAgent[][] agents;
  ISurface surf;

  MySurfaceAgent(MyPointAgent[][] agts) {
    agents = agts;
  }

  void interact(ArrayList < IDynamics > agents) {
    super.interact(agents);

  }

  void update() {
    super.update();

    int unum = agents.length;
    int vnum = agents[0].length;

    if(time%800 != 0){
      if(surf!=null) surf.del();
    }

    IVec[][] cpts1 = new IVec[unum][vnum];

    for (int i=0; i < cpts1.length; i++) {
      for (int j=0; j < cpts1[i].length; j++) {
        cpts1[i][j] = agents[i][j].pos;
      }
    }
    surf = new ISurface(cpts1,2,2).clr(time*.0033,0.,1);
  }
}
```

/ / Salvador F. Cortez Hultsch / / Coding Form Fall 2011
/ TutorSatoru Sugihara / / SCIArc