VS_ADVANCED CODING FORM

Urban Growth Patterns - the Randstad Holland

Sarah Maansson, SCIFI
spring 2013

Map of the Randstad Holland

Urban Growth Patterns _ Agent behavior

The intend of the project is to simulate the different possible growth patterns of a polycentric region, using The Randstad Holland as casestudy. The different polycentric growth patterns are studied as a 2 dimensional phenomenon - resulting in maps depicting possible growth patterns within the Randstad region - simulating how urban growth can occur according to the different polycentric patterns, while also taking into account the physical environment of the region - topology and other natural elements - and certain economic factors that play a role in the distribution of urban growht.

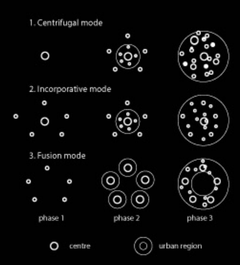
Polycentricity / modes of polycentric development

The Interaction between agents varies in the three different simulations of polycentric growth:

- in centrifugal mode growth starts from one primary center and expands from there in a randomly dispersed pattern

- in incorporative mode growth starts both from a primary center and sevelral surrounding smaller centers being incorporated as the urbanization grows. this type of growth, as opposed to centrifugal growth, will not create a totally random dispersal of urbanization, but maintain the primacy of the initial centers

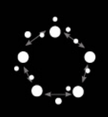
- in fusion mode several city centers grow at the same rate, creating a non-centralized urban pattern, though still maintaining higher densities around the initial urban centers



Champions' diagram of the development of polycentric cities

1. Centrifugal mode - agents aggregate more or less randomly within the region starting from a distincts center

2. Incorporative mode - agents aggregate primarily around existing centers, while urbanization thins out between centers



3. Fusion mode - agents aggregate so as to create connections between urban centers,

Interaction rules _ Centrifugal mode

- a former monocentric city developed into a polynuclear urban system, due to its continuous growth.

- in centrifugal mode growth starts from one primary center and expands from there in a randomly dispersed pattern.

- agents aggregate freely to create centers of random sizes within the urbanized field

Interaction rules _ Incorporative mode

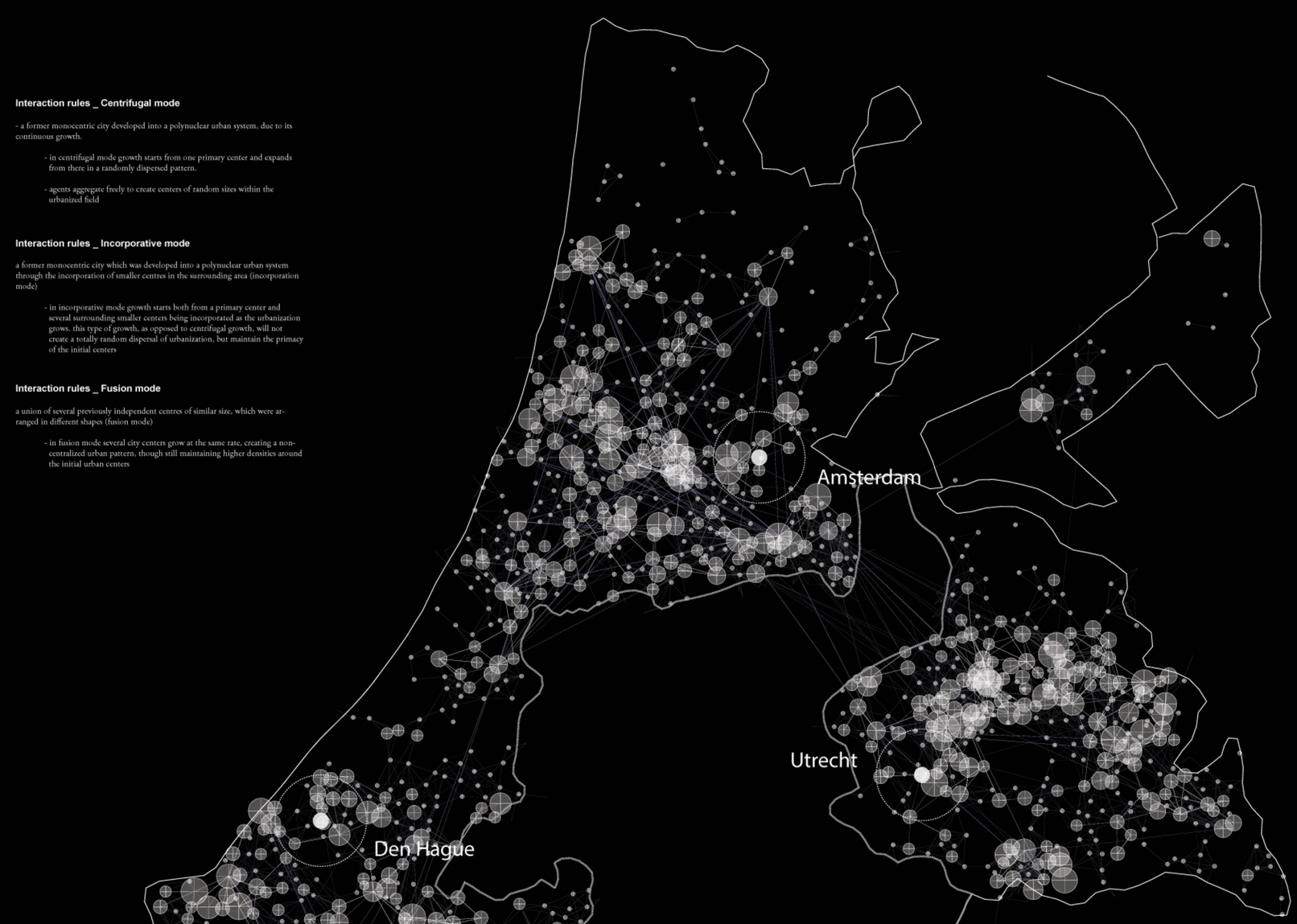
a former monocentric city which was developed into a polynuclear urban system through the incorporation of smaller centres in the surrounding area (incorporation mode)

- in incorporative mode growth starts both from a primary center and several surrounding smaller centers being incorporated as the urbanization grows. this type of growth, as opposed to centrifugal growth, will not create a totally random dispersal of urbanization, but maintain the primacy of the initial centers

Interaction rules _ Fusion mode

a union of several previously independent centres of similar size, which were arranged in different shapes (fusion mode)

- in fusion mode several city centers grow at the same rate, creating a non-centralized urban pattern, though still maintaining higher densities around the initial urban centers



Urban Growth Simulation, Existing conditions

1. The physical environment has an impact on the distribution and aggregation of agents - lower areas decreasing growth rates, while waterways increase possibility of growth

2. Economic factors increases possibility of growth around larger urban centers.

```
import igeo.*;
import igeo.p.*;
import igeo.io.*;
import igeo.gui.*;

import processing.opengl.*;

void setup() {
  size (800, 800, IG.GL);
  IG.bg(0);
  IG.top();
  IG.duration(1000);

  //passive behaviors - more growth around city centers of certain sizes (economy
     and at the large ports, Rotterdam and Amsterdam

  IG.open("boundary_greenheart.3dm");
  ICurve bounds = IG.layer("boundary_greenheart").curve(0);
  ICurve outlines = IG.layer("boundary_outline").curve(0);
  IPoint[] attractors1 = IG.layer("attractors1_cities").points();
  IPoint[] attractors2 = IG.layer("attractors2_water").points();
  IG.p(outlines);
  outlines.clr(255, 255, 255);
  bounds.clr(50, 50, 50);
  for (int i=0; i<1; i++) {
     new CityCenter(new IVec(30,50,0), bounds, outlines, attractors1, attractors2);
  }
  for (int i=0; i<1; i++) {
     new CityCenter(new IVec(-100,-60,0), bounds, outlines, attractors1,
     attractors2);
  }
  for (int i=0; i<1; i++) {
     new CityCenter(new IVec(-80,-110,0), bounds, outlines, attractors1,
     attractors2);
  }
  for (int i=0; i<1; i++) {
     new CityCenter(new IVec(85,-45,0), bounds, outlines, attractors1,
     attractors2);
  }

  IConfig.syncDrawAndDynamics=true;
}
```

setup of primary code parameters

```
class CityCenter extends IParticle {
  double size;
  ISphere sphere;
  ICurve bounds;
  ICurve outlines;
  IPoint[] attractors1;
  IPoint[] attractors2;
  ArrayList<ICurve> lines;

  CityCenter (IVec p, ICurve b, ICurve o, IPoint[] a1, IPoint [] a2) {
    super(p);
    size = 2.5;
    bounds = b;
    outlines = o;
    attractors1 = a1;
    attractors2 = a2;
    lines = new ArrayList<ICurve>();
  }
```

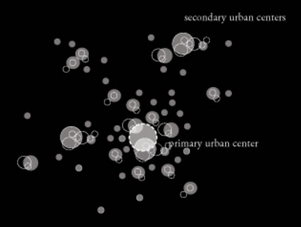
deifning the class CityCenter

```
  void interact (ArrayList<IDynamics>agents) {
    int mergeNum = 0;
    int maxMergeNum=1;
    if (alive()) {
      for (int i=0; i<agents.size(); i++) {
        if (agents.get(i) instanceof CityCenter) {
          CityCenter p = (CityCenter)agents.get(i);
          if (p!=this) {
            if (p.pos().dist(pos()) < (p.size/2+size/2)) {

              // merge two agents to create larger agent if two or more agents collide

              if (mergeNum<maxMergeNum) {
                double sz = Math.sqrt(size*size + p.size*p.size);
                double size2 = size + p.size;

                // if merged to certain size stop merging

                  if (sz < 8.5) {
                  size=sz;
                  p.del();

                  for (int j=0; j<p.lines.size(); j++) {
                    p.lines.get(j).del();
                  }
                  for (int j=0; j<lines.size(); j++) {
                    lines.get(j).del();
                  }
                  if (p.sphere!=null)
                    mergeNum++;
              }
            }
          }

          else if (p.pos().dist(pos()) < 85 && p.pos().dist(pos()) > 65 && size >
          7 && p.size < 6 && p.size > 5  && outlines.isInside2d(pos) &&
          !bounds.isInside2d(pos)) {

            //drawing lines between cityenters of same size larger than certain number

            ICurve line = new ICurve(p.pos().cp(), pos().cp()).clr(0.6, 0.6, 1.0, 0.05);
            lines.add(line);
          }

          else if (p.pos().dist(pos()) < 10 && p.size == size && outlines.
          isInside2d(pos) && !bounds.isInside2d(pos)) {

            //drawing lines between cityenters of same size

            ICurve line = new ICurve(p.pos().cp(), pos().cp()).clr(0.2, 0.2, 0.24, 0.1);
            lines.add(line);
          }

          else if (p.pos().dist(pos()) < 15 && p.size > 3 && p.size < size && out
          lines.isInside2d(pos) && !bounds.isInside2d(pos)) {

            //drawing lines from larger centers to new smaller centers

            ICurve line = new ICurve(p.pos().cp(), pos().cp()).clr(0.98, 0.98, 1.0, 0.15);
            lines.add(line);
          }
        }
      }
    }
  }

  void update() {

    if (!outlines.isInside2d(pos)) {        // only generate growth inside country boundary
      IG.p("out of outline");
      del();
      return;
    }

    if (bounds.isInside2d(pos)) {           //no or little growth inside green heart boundary
      IG.p("inside boundary");
      del();
      return;
    }

    if (time()==0) {
      sphere = new ISphere(pos.dup(), size/4).clr(255, 255, 255);
      new CityCenter(pos.dup().add(IRand.pt(-50, -50, 50, 50)), bounds, outlines,
      attractors1, attractors2);
    }

    else {
      if (sphere!=null && !sphere.radius.eq(size/4)) {
        sphere.del();
        sphere = new ISphere(pos.dup(), size/2).clr(255, 255, 255);
      }
    }

    if (IRand.pct(0.5)) {
      new CityCenter (pos().cp(IRand.dir(IG.zaxis).len(size*2)), bounds, outlines,
      attractors1, attractors2);
    }

    if (attractors1!=null) {
      if (IRand.pct(1.5)) {
        new CityCenter (pos().cp(IRand.dir(IG.zaxis).len(size*2)), bounds, outlines,
attractors1, attractors2)
      }
    }

    if (attractors2!=null) {
      if (IRand.pct(1)) {
        new CityCenter (pos().cp(IRand.dir(IG.zaxis).len(size*2)), bounds, outlines,
        attractors1, attractors2);
      }
    }
  }
}

//void draw(){
//saveFrame("final_simulation4-###.png");}
```

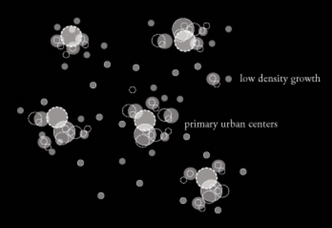
output of code to image

Simulation of urban growth patterns - Centrifugal mode, Incorporative mode and Fusion mode

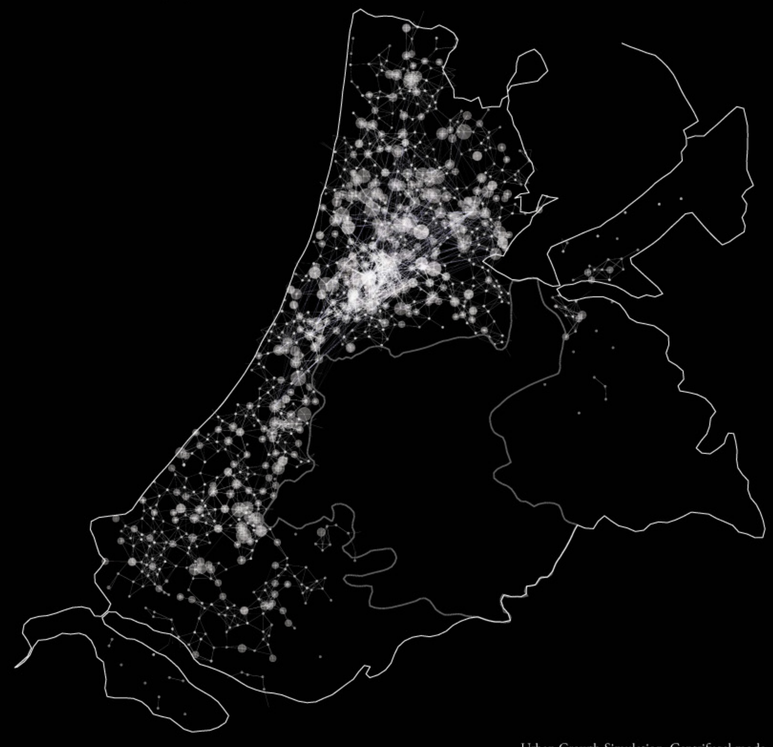
The different patterns are simulated with the chosen site as a case study.



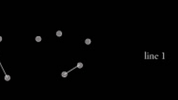
1. Centrifugal mode - urban growth starts from a primary city center, from which agents disperse randomly within the urbanized region.



3. Fusion mode - agents primarily aggregate between existing urban centers, creating dense urbanized zones between existing centers

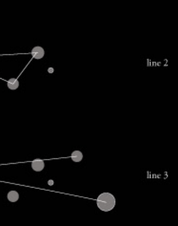


Urban Growth Simulation, Centrifugal mode



line 1

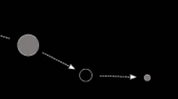
Interaction rule for LineAgents -
agents react to the aggregation, connecting aggregations across the urbanized field



line 2

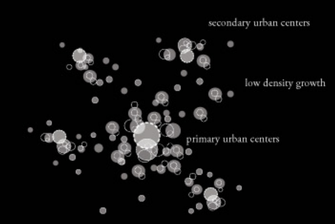
line 3

Update rule for all agents -
passive environmental factors control growth within the region

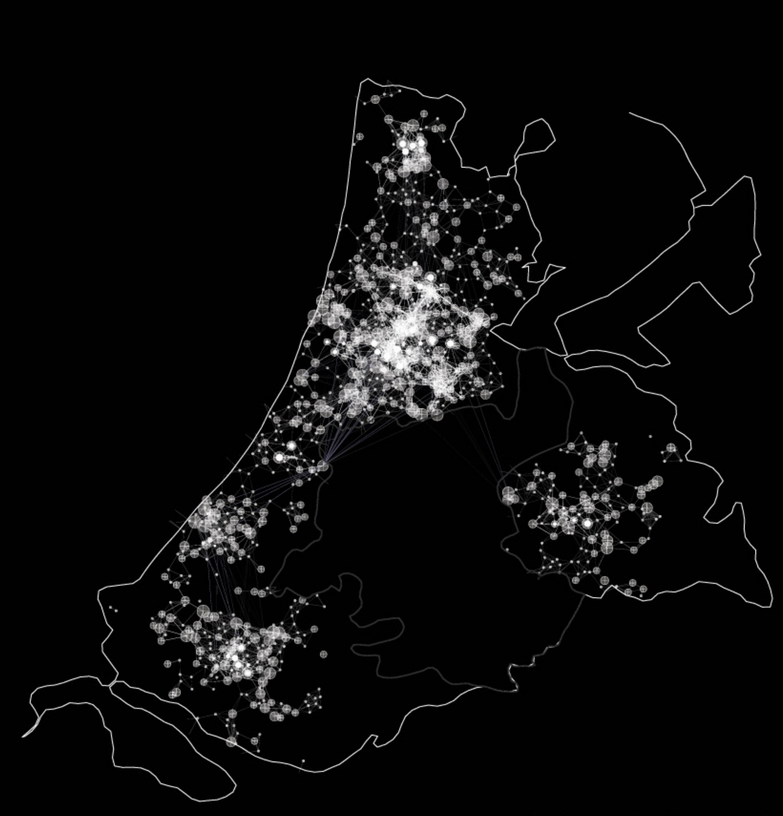


Update rule for point agents -
point agents duplicate to produce new agents of smaller size emitted in random directions from first agent witing a set distance from previous agent

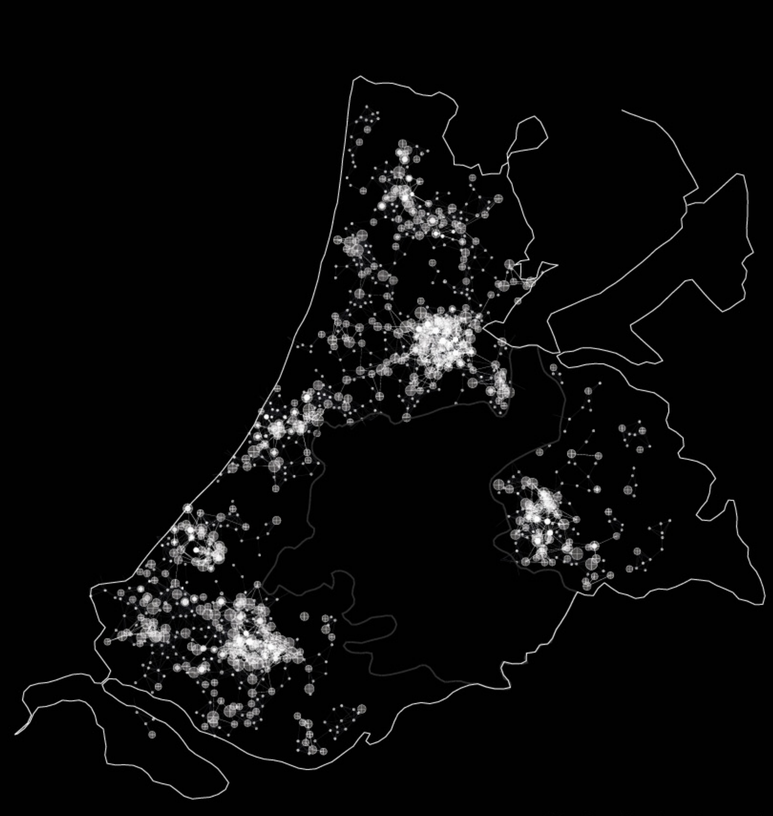
Update rule for all agents -
passive environmental factors and economic factors control growth within the region



2. Incorporative mode - agents disperse equally throughout the urban region defined by several existing city centers. Outside of the existing centers, urbanization spreads out thinly by maintaining a minimum distance between point agents once agents merge to a certain size



Urban Growth Simulation, Incorporative mode



Urban Growth Simulation, Fusion mode