Spatial Configurations: Complex Systems Experiment of Design Automation

Choesnah Idarti, MSc., B.Eng Arch.

Submitted for an MPhil.
Postgraduate Research MPhil Computing and Technology
School of Architecture, Computing and Engineering
University of East London

28 February 2013

Abstract

The study of design does not seem to shift from the paradigm that design process is a complex and seems similar to the nature of a black box system. It is a process which can be viewed solely in terms of its input and output, without a detail prescription of the process to produce that output. The complex nature of design process, and architecture design in particular have been explained by experts (Alexander 1964, Anderson 1966 to name a few), but this thesis underlines the work around of the rigid clear box system of computing to get a reliably working non-wired-in process to support the notion of architecture design as a complex system.

The clear box nature of computing process is a fundamental characteristic of computing, so the process to be proposed has to work within this framework. The process would have to be prescribed or otherwise it cannot be executed. Many machines which given the same input and the same process, will all result in the same output. A single machine which given the same input and the same process to execute for many times will all result in the same output each time. However, the development of computing has enabled the processing of multiple inputs. The significant of parallel computing is that it seems to provide a window of epistemic autonomy within a process.

There is a large philosophical and theoretical discussion behind the notion of epistemic autonomy which this thesis tried to introduce a preliminary summary, and sums it into the following description. A system consists of one single bird and a process of how to fly may result in a bird flying. Given many birds and implementing flying process synchronously to all the birds could result in a swarm of birds. There is not much different to see one bird or many birds flying in the sky, except that in many birds that each are using the same flying process would result in an underlying flying configurations. The underlying flying configuration is not part of the system; it is an emergence of the system. So the emergence structure of flying bird was enabled by a window of epistemic autonomy which comes from the use of many flying birds as opposed to a single flying bird.

A window of epistemic autonomy seems to have been created in the programming experiments with the implementation of a basic Agent-Based Model (ABM) as ABM is inherently an autonomous non-wired in process (Cilliers, 1993). The coding is based on a system introduce by Reynolds (1987) which a program was already built and modified many times in projects within CECA – UEL. To put simply, the inputs are multiple copies of one type of entity placed in using randomizing code, and the process synchronously applied to all these copies are to move towards the closest out of other three neighboring entities.

The utilization of ABM into the programming experiment seems parallel with the findings in the literature review where it proposes a summary of a production of space by way of using a binary approach known to be brought up by Lefebvre (1974). Lefebvre seems to suggest that the very basis of spatial production is that space consists of either a moving or a non-moving social entity. Thus the criteria above since then been adapted to accommodate a simple social relations and this is called Social Preference Matrix (SPM). SPM is an original contribution in the form of coding that comes out of this thesis's programming experiment. To put simply, SPM enables the identification of heading towards the nearest out of three neighboring entities only when it is the specific entity it relates to in the SPM. When this is triggered, both entities i.e. the one moving towards to and the designated

entity both will eventually within a specific constant distance with each other, and these will stay in a loop of attract and repel, which is perceived as simulation of these being stop moving.

The development of the programming experiment have found that when all entities are identified as occupying entities, eventually all entities will loop in attract and repel, i.e. all eventually will be non-moving. When introduce with non-occupying entities, i.e. those not identified and included in the SPM, then the possibility to have similar characteristic to what is known in the Configurational Theory as the movement space appears.

The specific of architectural production may be seen as an opposite of the dynamics seen in an underlying configuration of bird swarm. Architectural subjects particularly regarding spatial configurations seems to be required to be static; there is not known liquid or ever-changing spatial configurations. Thus instead merely producing a system of moving around entities in space, this then had to be translated into some static versions of events. In this programming experiment, these were built on *the basis of notations* provided by the Configurational Theory (Hillier, 1996). Interestingly, he also stated in his previous work (Hillier, 1978) that there is a production of space where a larger space is being divided into smaller space.

There is a body of research into programming headed by Mitchell, et.al (1976), which stretches for about three decades afterwards. However as far as the literature investigation went, none has seemed to explore the notion of epistemic autonomy in the production of spatial configurations. Following this finding, the programming experiment then added in a program called Voronoi Diagram (version coded in the programming experiment is as prescribed by Akl and Lyons, 1993) which enables several significant developments in producing spatial configurations by way of dividing space.

Firstly, the production of space by way of dividing space is based on an emergence of underlying configuration out of possible social relations between entities occupying the space. Secondly, the division is based on maximum arrangement of occupiable space between all occupying and non occupying entities, because Voronoi Diagram divided space equally in between all of the identified entities. Thirdly; a part from an emergence spatial relations by way of utilization of SPM within the ABM, the employment of Voronoi Diagram also enables the emergence of shapes and dimensions out of the divided space.

At the end of this study, the programming experiment has resulted in a programming framework named Spatial Languaging. A part from that there seems to be a promising field of research into programming under the notion of epistemic autonomy specifically develops for architectural systems, because there are numerous methods of parallel processing and many different media of implementing ABM. More importantly, the notion of structural emergence seems applicable to many aspects of architecture and all worth exploring.

Contents

Table of Figures

Table of Codes

To my family, Erlangga, Haniya, Aleena, my late Mum and my Dad for their reimbursable continuous supports,

And

To my programming tutor, Paul Coates who understood my vision and helped me to manifest the other pair of boots gifted by Hillier to both of us his pupils.

1. Introduction

... ARCHITECTURAL RESEARCH SHOULD ALWAYS FALL BETWEEN WHATEVER POLAR OPPOSITES ONE CARES TO DEFINE.. (COATES, 2010, P.1)

Ludwig von Bertalanffy (1968) aims for a theory that unifies all explanations with his General System Theory (GST). GST is not the only field of study that works towards theory of everything, but recent development in agent-based modeling seems to provide sufficient verification for GST visions of universality. The grand aspiration may already been fulfilled when Stephen Wolfram published an empirical and systematic study of simple programs. Wolfram (2002, p.465) concludes that simple programs with different rules can have similar behavior, such as some natural systems which seem very different but behave in similar ways. It is proposed that such scheme is applicable across different fields of sciences. This emphasizes on isomorphism between different systems and might explain similarity in either their internal structures or their external behaviors.

Paul Coates (2010, p.1) suggested that architects and other designers would be interested in this development because it manifests a new epistemology. Form is a complex with many interrelated aspects, creating a simple program based on a particular known aspect of form might lead to a production of interesting forms. A simple program which is parallel to an existing system related to form may demonstrate a similar structure or behavior and pattern of forms. Among these programs are computer models which explore the notion such as how space produces society and how society produces space.

Preliminary works for such production of space has been done by Bill Hillier with “a theory of space with its own descriptive autonomy, i.e. a theory without interpreting other theory” (Hillier & Hanson, 1984, pp. 5-9). A space of a social theory of space is an elementary cell which is a dichotomy of the outside and the inside. For that space to grow there are two ways; firstly, by subdividing it thereby maintaining the internal permeability, or secondly, by aggregating it thereby maintaining the external continuous permeability (*ibid*., p.19)

Accompanying Hillier proposal above, the Alpha Syntax program was developed by Paul Coates. Alpha Syntax exclusively an application of the second method of the proposed spatial growth, it aggregates cells which connect throughout with an axiomatic permeable cell. There are two versions of Alpha Syntax program developed; a grow method by orthogonal constrains i.e. within cellular grids (*ibid.*, chapter 2) and the tree branching which is a grow method free from orthogonal constrains (Coates, 2010, pp.153-154). So far, there is no development of a specific computer program in any of Hillier’s publications which applies the growth method of dividing the internal space.

Later on, Hillier proposed a theory of architecture and a social theory of architecture (Hillier, 1996). Theory of architecture stated that *architecture* is a theory applies to building. It implies that architecture is a physical manifestation of *how the architect* regulates the elements of a building. A social theory of architecture introduced by Hillier stated that the configuration of space influenced and can be influenced by the configuration of space (*ibid.,* p.31). The social theory of architecture seems to suggest isomorphic mapping between configuration of people and spatial configuration. It is *how a spatial system* would influence a configuration of people who are using that space, whereas those changes in spatial configuration might have been caused by the same configuration of people. If a space as defined by the social theory of architecture represents a dichotomy of inside and outside, it might indicate that spatial systems is a dual to a system created by configuration of people.

The proposition would be that an isomorphic mapping of space and social simulation would add to the theory of architecture; it would be that architecture is a physical manifestation of *how social system* might regulate the elements. The aim of this thesis is to create a simple program that might lead to a production of space using the assumption that if there is a dichotomy between a spatial configuration and a configuration of people, then a simple program simulating configuration of people is isomorphic to a production of spatial configuration.

Simulating configuration of people requires a model of spatially occupying entities which follow simple configurational rules. This can be achieve using agent-based model of spatially occupying entities such as Boids (Reynolds, 1987) which coupled with agent's preference such as Thomas Schelling Frog Pond scheme (see similar model by Epstein & Axtell, 1996). Thomas Schelling racial dynamics model could be expanded into more categories as it has already contain a very simple social rules; either an agent is a friend or not, to the other agent.

With isomorphic assumption, a simulation of agent-based model of configuration of people would mean simultaneous productions of spatial configurations. However how to map the emergent spatial configurations forming out of the feed provided by the agent-based model simulation? Referencing to a couple of projects related to dividing a space, the possible solution is to represent the emerging spatial configurations as a specific bounded universe consist of many of bounded cells mapped by each spatially occupied agents.

Accordingly, this program can be proposed as the first growth method of a social theory of space by Hillier & Hanson (1984, chapter 2), which is a method of subdividing the space. Using algorithmic geometry planar dividing method called Voronoi Diagram (Boisonnat & Yvinec, 1998, p.405) and a color scheme representation of agents which also use to color the divided cells, spatial configurations as isomorphic map of Euclidian bounded social interactions can be manifested.

The existing algorithms modified in the program are agent-based simulations already developed at the Centre for Evolutionary Computing in Arcitecture (CECA) and a newly developed computational geometry method called Voronoi Diagram. The specific small original contribution made in the program is a social preference module within the agent-based simulation. It is a two dimensional array that feed from a social relationship table. A text document containing a matrix of that table; either a friend or not friend (binary) between agents of different types is prepared by the user and this will feed into the module; it is the basic of social construct of otherwise random localized chances of the forming of configurations of people.

This program is a specific framework for exploration of spatial design with notion of actual spatial use by configurations of people as its foundation and thus it is called Spatial Languaging. The isomorphic assumption follows the principle of General System Theory, thus mapping of other aspects of environment in a design process would made visible development of more spatial generators.

2. Literatures

2.1. Space: a social theory of space

Any computer program basically consists of elements and methods processing these elements. A simple program seems to refer to the use of the most basic elements and particular methods that process elements within micro context or local rules however results in observable un-programmed behavior of the whole. For example a simple program consists of random mobile spatial entities with an instruction to follow other entities that is closer to itself. After a while, this set of basic elements and local method would be perceived to give rise to; firstly a flocking configuration, and later on a notion of a leader. These emergent behaviors of both flocking and leadership were not programmed by the programmer. These are simple programs as explained by Stephen Wolfram (2002) and which Paul Coates and students programmed for architecture.

For spatial configuration within a social theory of space, Hillier suggested that the most basic element would be *a cell* which represents a dichotomy of the inside and the outside. Therefore space could be identified into two categories; the interiors and the (collective) exteriors (1984, p.19).

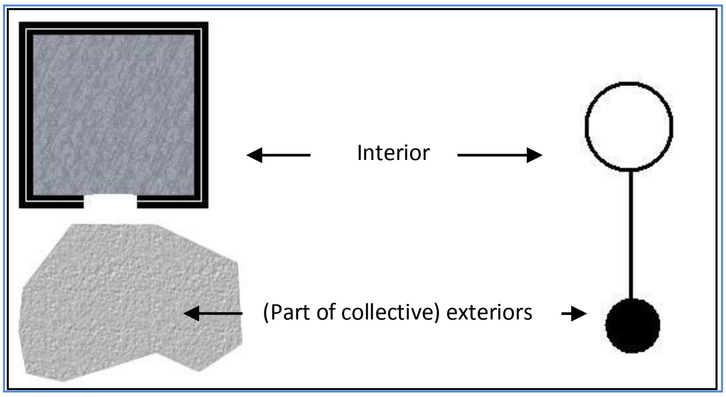


FIGURE 1 ELEMENTARY SPACE

There were several kinds of notations adopted by Hillier in his publications, therefore a convention is required. The graphical notation as seen on the right above in Fig. 1, and there is also notation which seems to be use for written discussion or in paragraphs, where he uses the letters X and Y. The letter X is used for interior and Y for exterior. The graphical notation will not be use within this document, but diagrammatic notation as seen on the left in Fig. 1 above will be use in diagrams and the use of X and Y in corresponding texts.

This elementary space serves the purpose as (Hillier & Hanson, 1984, p.52):

1. an elemental structure that characteristically is an irreducible objects and relations,
2. independent notation which can be use in discussion or analysis about space,
3. being part of a coherent system where many elementary spaces related to one another,
4. the only element of combinatorial system based on specific syntax which makes up more complex structures.

There is no formal mention of local rules for processing the elements given by Hillier, therefore the assumptions are as follows; there is the interior to exterior spatial relation, and the exterior to exterior spatial relation. In Hillier terminology these are the syntaxes.

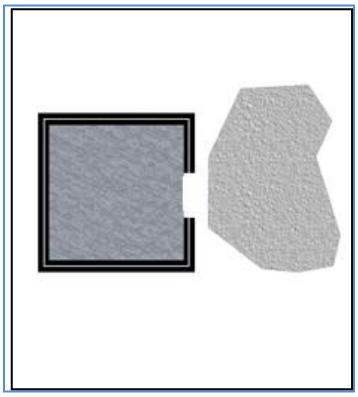


FIGURE 2 (INTERIOR-EXTERIOR) SPATIAL RELATION

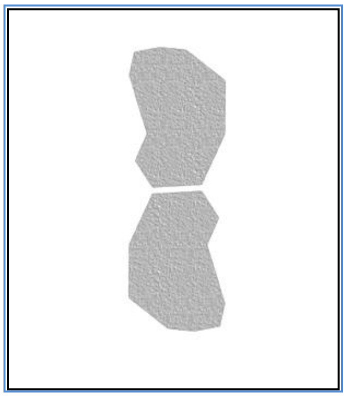


FIGURE 3 (EXTERIOR-EXTERIOR) SPATIAL RELATION

Originally, there were 8 different set of syntaxes but only one set of syntax as shown above was developed because the rest were over-constrained, and had been tested and summarized as incapable to produce varied results as what had been given by the chosen set (Coates, 2010, p.138).

2.2. Alpha Syntax: methods

The elementary space and the micro spatial relations as explained by Hillier Space Syntax theory provide the ingredient for generations of more complex form. Generative methods such as Diffusion Limited Aggregation (Batty, 1994) and Lindenmayer System (Prusinkiewicz & Lindenmayer, 1990) which uses elementary form and local rules, and both produce similarly complex fractal forms. However they consist of entirely different algorithms and thus different in the way they process the basic elements and use different context in defining local rules.

Although each algorithm is unique, they could produce forms which as a whole can be perceived as similar forms produced by two different methods. There are two ways to process the basic elements which concern with an application of local rules; grid base or non-grid method. The grid method is known as Cellular Automaton, and the non-grid is graph rewriting method, for example the Lindenmayer system. Alpha Syntax features developments which seems to incorporate these methods.

2.2.1. Grid Method: Cellular Automaton

Celullar Automaton (or CA) is a successive addition of patches on grid following a set of state-bound rules, leading to more complex universe of patches than the initial state of that universe. Wolfram (2002) provides the most extensive study of Cellular Automaton. The universe changes from a simple to a complex one step at a time. Parts are added at each step and their assign locations are in correspond to the set of existing local parts. Chronology and local continuity are the key features of this method of form where previous forms will dictate the later forms, and form at the next stage is within local of form at the previous stage.

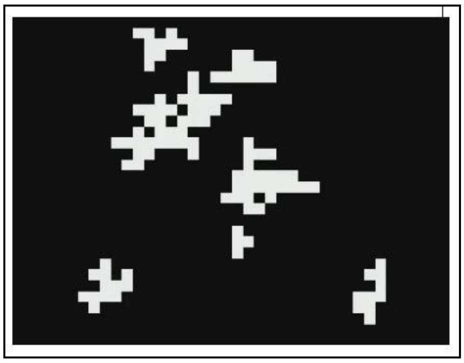


FIGURE 4 LEAK METHOD (HOVER OVER PICTURE TO FOLLOW LINK HTTP://WWW.YOUTUBE.COM/WATCH?V=QRJ5GKVJFSK)

Alpha Syntax was developed by Paul Coates following Space Syntax framework and inspired by Maruyama 1969 flow diagram (Coates, 2010, p.131). It employs CA method known as the leak method. The above figure is the Monte Carlo simulation created using the same method.

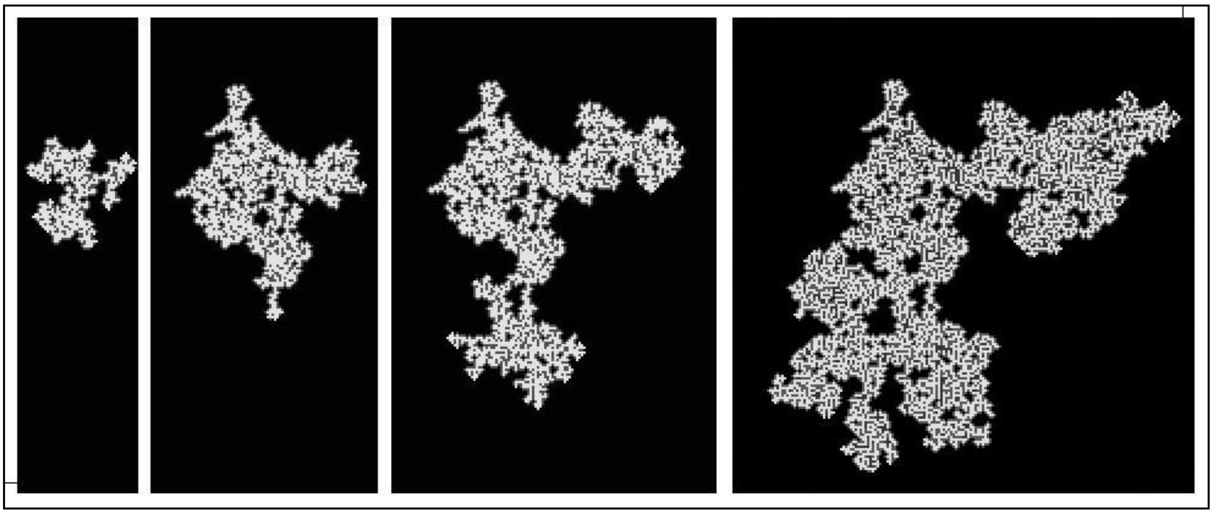


FIGURE 5 THE ALPHA SYNTAX - COPYRIGHT PAUL COATES 2010

2.2.2. Non-grid method: Formal Languages

This is a successive part-replacing method of a simple initial object using a set of rewriting rules, which lead to more complex form of its initial. Formal Languages is made of sets of strings and methods for generating, recognizing and transforming these strings (Chomsky, 1957). Formal Languages, including L-System applies initiator and recursive generator to the basic element, i.e. strings to get the more complex form.

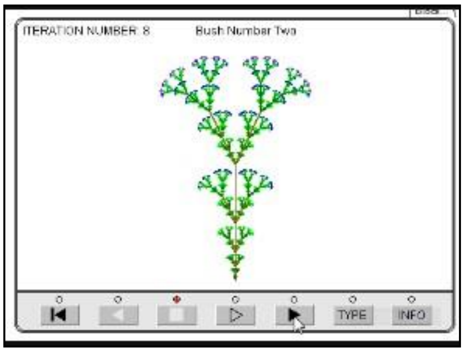


FIGURE 6 L-SYSTEM (HOVER OVER PICTURE TO FOLLOW LINK HTTP://WWW.YOUTUBE.COM/WATCH?V=WD8RT-DIAKS)

Paul Coates further on to develop the Alpha Syntax program based on Diffusion Limited Aggregation method which result is a resonance to an L-System method known as the Branching structures (2010, pp. 152-157). Branching structures characteristically has a sequence of parts which is known as axis. An axis is essentially a chain which the beginning of form is connected through out by the same elements. From this main structure, the rules will produce branches adding more elements and thus the form becomes more complex. Alpha Syntax has Y spaces creating the continuous axis, and it branches as X spaces are added at each step of the program.

Figure below shows an emerging axis created by simple rule of f f [-f] [-f] using L-System. Later version of Alpha Syntax was created with similar continuous axis.

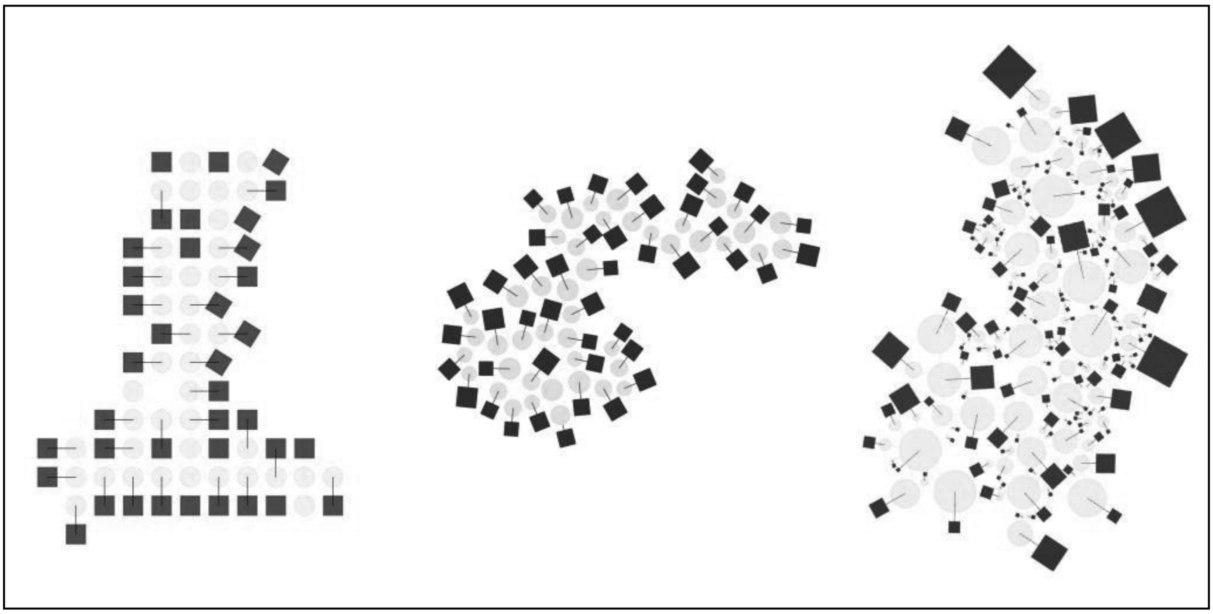


FIGURE 7 ALPHA SYNTAX 2.0 EARLIER TO RECENT – COPYRIGHT PAUL COATES 2010

2.3. Voronoi Diagram: a method to divide space

Voronoi Diagram is an algorithm which manifests smaller bounded spaces within a large undefined plane. It divides congruent space between points. To draw a Voronoi diagram is to connect all Voronoi vertices. The algorithm will search for the Voronoi vertices which are the centers of a circle where the nearest (least) three points are on its circumference, and such no other point will be inside the circle (Akl & Lyons, 1993, pp.99-101).

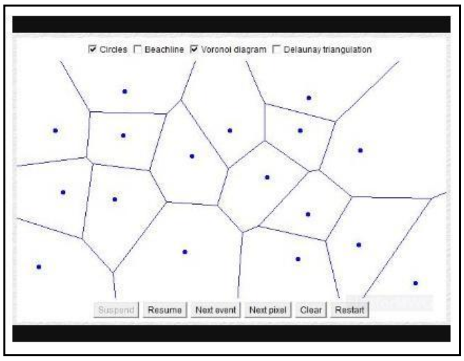


FIGURE 8 VORONOI DIAGRAM (HOVER OVER PICTURE TO FOLLOW LINK HTTP://WWW.YOUTUBE.COM/WATCH?V=E9QHESSEWGU)

The shape of each Voronoi cell is unique because its sides are the manifestation of otherwise invisible exact middle spaces in between the centre of one cell to another point of the opposing cell. These shapes can be altered and such Voronoi Diagram is known as generalized Voronoi Diagram. Because these shapes created entirely based on the positions of the centre of the cells, they would have irregular angles. The generalized Voronoi Diagram contains rules which alter the angles and further on some generalized Voronoi Diagram alter the shape entirely by ruling how many sides there should be out of a cell.

2.3.1. Voronoi Diagram as Social Art

Scott Snibbe used Voronoi Diagram to create an interactive social art installation (Snibbe, 1999). The space is divided synchronously according to people who join in a specific bounded plane. As they walk around, these lines are updated to note the exact boundaries where spaces between people are congruent at each time.

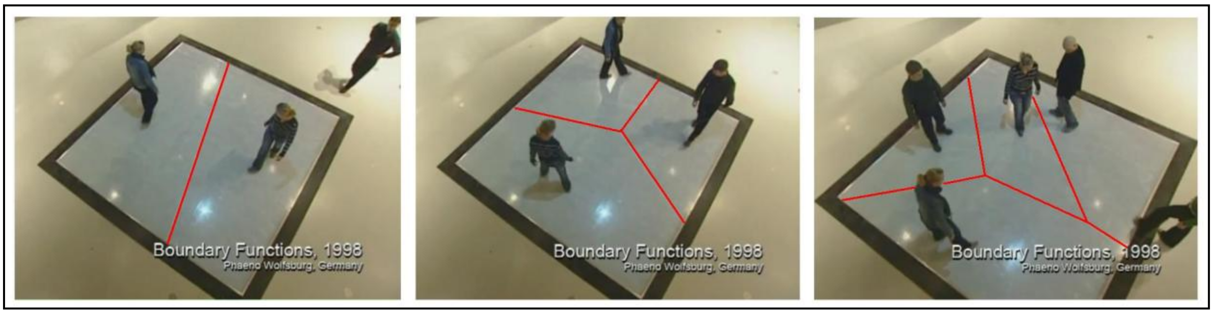


FIGURE 9 SCOTT SNIBBE INTERACTIVE ART (HOVER OVER PICTURE TO FOLLOW LINK HTTP://WWW.YOUTUBE.COM/WATCH?V=1P96BTARFKC)

This project shows significant insight into how space could be synchronously mapped in configurational terms. It demonstrates the simulation of spatial configurations which are made of the occupied space (X space) and the movement space (Y space) in real time.

2.3.2. Voronoi Diagram as Urban Plan Generator

Kaisersrot project use Voronoi Diagram as an urban plan generator (Kaisersrot, 2001 & 2003). It is a generalized Voronoi Diagram which use repels and attracts rules between the points which triggered straight after the points were placed in and thus creates congruent space in between them. Voronoi then divides the plan, and naturalizing the organic shapes of the original Voronoi cells into rectangular shapes. Kaisersrot also programmed in nodal configurations synchronously. For example one can drag a church across and a group of specified buildings will follow and configure all around it following the same process as if it is placed in as in the beginning. A fixed size road is then added up by the user. The road has its starting point, crossroads and ending point placed in manually.

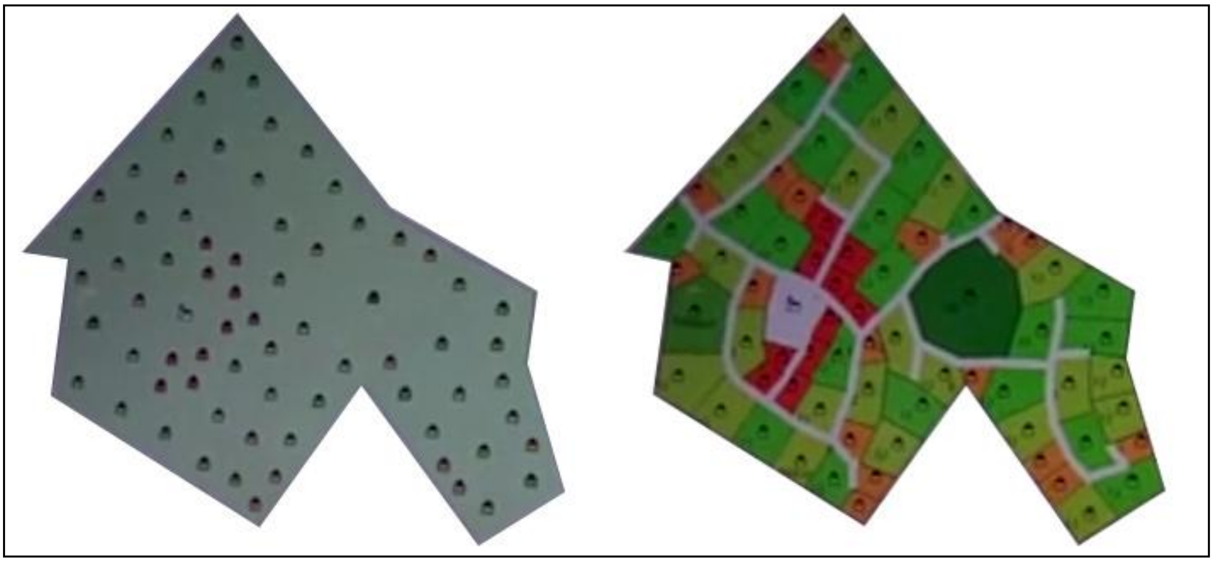


FIGURE 10 KAISERSROT VORONOI DIAGRAM (HOVER OVER PICTURE TO FOLLOW LINK HTTP://WWW.YOUTUBE.COM/WATCH?V=ZW7_JFHI5HK&FEATURE=PLAYER_EMBEDDED)

2.3.3. Voronoi Diagram as Spatial Configurations generator

Snibbe's art and Kaisersrot projects seems to suggest that Voronoi Diagram is a good medium to enable isomorphic mapping of spatial configurations and configurations of people. In Snibbe's art, Voronoi Diagram seems to convey a natural sense of structural mapping of space as it is being used by configurations of people. In Kaisersrot, Voronoi Diagram functions directly as configurational generator.

Spatial configurations as explained by Hillier emerged from the relations between one cell to other cells. Similarly, configurations of people must have embedded within them the relations between one individual to the others. Under this assumption, there would be preliminary work into developing a set of meta-relations between the entities which represents configurations of people.

Kaisersrot has been successfully generating urban plans. However the use of Voronoi Digram was to generate a pre-set assumption of how one cell should be related to each other. The church nodal configurations and particularly the use of repel and attract are the procedures coded in to ensure there are certain relations between a cell to the other cells. The notion of isomorphic mapping on the other hand is about resulting in emergent configurations as opposed to the generation of pre-set configurations.

Therefore, the program should be a manifestation of exploring the randomness of configurations of people, which in part can be made possible by manifesting real-time random movement of people as much as has been shown in Scott Snibbe's project and thus to capture the emergent spatial configurations using Voronoi Diagram.

2.4. Agent-based Model: Simulation of Random Movement

ABM or Agent-based Model is a type of simulation which is developed to observe emergent behaviors of a dynamic complex system. In Boids (Reynolds, 1987) flocking behavior and a notion of leader are noted to have been emerged out of an initially random movement of spatially occupying movement entities called boids. Each boid were simply programmed to direct its own heading to follow the nearest other moving boids. Once all boids have the same headings, the notion of flocking boids and its leader appear.

Emergent properties within such simulation can be exploited to produce emergent spatial configurations. Un-programmed configurations of people can be manifested through random movement of the agents, and such program already existed on AutoCAD platform in the Centre for Evolutionary Computing in Arcitecture (CECA) research archive. However, there is not a function of class types of agents and how different types of agents relate to each other. Classification of agents will create more structured chances of configuration which some of such behaviors have been demonstrated by Schelling's frog pond (Epstein & Axtell, 1996).

Thomas Schelling pond model was critical in population dynamic research, because it maps the otherwise unseen configuration of frogs and turtles in space based on each individual preference criteria. The simulation was really simple; there were two groups of turtles and frogs (*ibid.*) and each frog was programmed to only settle at a locality which has a minimum number of other frogs, and the same rule applies to turtles. The emergent spatial segregation of frogs and turtles was not program into each individual frog or turtle but a result of local interactions between them.

The lesson from Schelling frog pond model is that by having different individuals with different preferences can result in interesting spatial configurations.

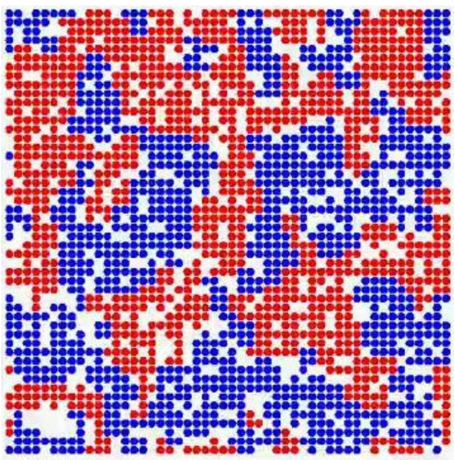


FIGURE 11 SCHELLING SEGREGATION (HOVER OVER PICTURE TO FOLLOW LINK HTTP://WWW.YOUTUBE.COM/WATCH?V=A_XTBOYO8JC)

3. Spatial Languaging: The Program

3.1. Agent-based Model Basic Procedures

The Centre of Evolutionary Computing in Architecture (CECA) had several types of ABM applications available in various computing platforms. Some of these are in various programming language such as Visual Basic which can be modified in AutoCAD programming platform from version 2004 onwards. To have the program built in AutoCAD platform or any other drawing platform would enable efficient transition from generations of conceptual spatial configurations into eventual development of spatial designs.

There are basic procedures within these programs which identifies with ABM and they are as follows

3.1.1. Create Agents

This procedure enables manifestation of how an agent should look like on screen. There are programs in CECA that based on ABM platforms such as NetLOGO (Wilensky, 1999). There are also programs which were developed using software such as AutoCAD and these use circles as manifestation of agents (DRAW CIRCLE is a command available in AutoCAD). This command is coupled with a random placement of the circle.

3.1.2. Move Agents

There are basic geometrical calculations to get the circles moving around the screen, which pull local data of each agent to process the next step. To move forward, there are calculations related to random heading and this will use a constant as the step range. The use of random heading to get the agent on to the next step would enable greater chances of emergent behavior. Besides these calculations, there are other calculations which involved with situations each agent would be in, such as too close to other agents, or too close to the boundary of the universe. And as the simulation involves many agents, there is a state called *limbo world* or *synthetic synchronicity*; a state where each agent's corresponding parameters are calculated for the next step to get all agents simultaneously into their next positions.

3.2. Agent-based Model for Spatial Languaging

This program Spatial Languaging (SL) requires development of ABM basic procedures of creating agents. There are two hierarchies of creations to be manifested; the visitors and the occupiers, and the type of each agent created within the occupiers. Then there is a Social Preference Matrix that relates to how each agent should move when it bumps into a particular agent.

3.2.1. Create Visitors and Occupiers

The first hierarchy is whether or not an agent is a visitor or an occupier. The early version of SL has shown that all agents within ABM will eventually settle and thus all of them identify with occupiers. This is unfortunate for generations of spatial configurations as these will need “roads” or movement spaces, or Y spaces as known in Hillier’s terminology. To avoid top down entry of such spaces (as shown in Kaisersrot project), it is logical to embed them directly into ABM and thus these spaces would be an emergent property of the overall spatial configurations.

The program has a procedure to control how many percentages of visitors out of the overall agents’ population. This is currently coded within the program as opposed to a user controlled entry, although it is possible to change it into a user entry.

```
For c = 1 To pts
   randpoint(0) = random(-universe * 0.8, universe * 0.8)      'choose random x y z the array is autocads way of holding a point
   randpoint(1) = random(-universe * 0.8, universe * 0.8)
   randpoint(2) = 0
      walker.diameter = startdiam 'sizs(Int(random(1, nsize)))
      walker.steplength = walker.diameter
      walker.heading = random(0, 360)
           Set acircle = ThisDrawing.ModelSpace.AddCircle(randpoint, walker.diameter / 2)     'walker is represented as circle
               walker.circleid = acircle.ObjectID
                If Rnd > 0.2 Then 'change here to control how many percentage of visitors
                    walker.colour = Int(random(1, CDbl(groups)))
                Else
                    walker.colour = 256 ‘these will be visitors
                End If
                   acircle.color = walker.colour 'colour the circle according to the required number of types of agents
                   acircle.Layer = "circle_layer"
                   acircle.Update
       walker.begin.x = randpoint(0)      ' set walker's position to be that same as the circle
       walker.begin.y = randpoint(1)      ' (using my preferred way of defining a point
       walker.begin.z = 0
           thecircles(c) = walker
Next c
```

CODE 1 ASSIGNING VISITORS AND OCCUPIERS

The proportion in the code above correspond to 20% of the whole population of circles created on the screen will be assigned as visitors agents. These circles will be assigned different colors, which is specific and not available to assign occupier agents with the same color of visitors (i.e. color 256).

3.2.2. Create Types of Occupying Agents

The next hierarchy of agent relates to creating entities which enables manifestation of different preferences, i.e. different types of agents identify with different preferences and thus would yield to interesting configurations as has been indicated by Schelling’s frog pond simulation.

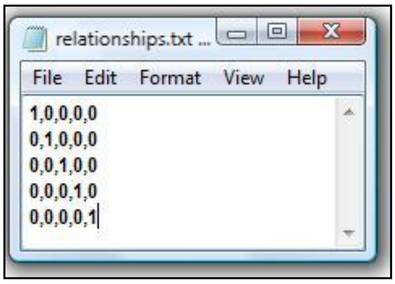
The occupier agents will then be assigned random proportions of different types of agents. The number of types is a user controlled entry and is entered at the beginning of the simulation. Random proportions of types refer to creating more chances as oppose to coded-in proportions of each different agents. So user will be able to enter how many types of occupying agents but cannot control how many agents of each type would be.

Different color circles will then assign randomly to agents, but these will not use the specific color has already assigned for visitor agents.

3.2.3. Social Preference Matrix

The Schelling frog pond model (Epstein & Axtell, 1996) uses a proportion of local friend for a type to settle in the locality. This is simplified within SL; for each three closest neighbor, if there is a friend nearby then move towards it and then slow down. Therefore Social Preference Matrix (SPM) relates to procedure about agent's movement; it controls when agent will change its movement.

SPM is a user entry in a form of a text file and corresponds to the number of types will be required for the simulation. The user needs to write a preference matrix which has the same size as the number of types of agents that will be in the simulation. For example, 5 different types of agents will need a SPM in a size 5 by 5 as follows.



	Agent A	Agent B	Agent C	Agent D	Agent E
Agent A	1	0	0	0	0
Agent B	0	1	0	0	0
Agent C	0	0	1	0	0
Agent D	0	0	0	1	0
Agent E	0	0	0	0	1

FIGURE 11 SOCIAL PREFERENCE MATRIX, TEXT FILE AND THE CORRESPONDING MATRIX

The binary 1 and 0 is in use to identify the other agent is a friend or not friend. Thus SPM above correspond to dynamic segregation within the population where each agent will only settle with the same type. Integration or segregation within the population can be structured using this preference matrix by changing any 0 to 1; however random placement and movement will enable emergent configurations as opposed to structured and coded-in configurations.

```
ReDim originalpoints(1 To pts) As mypoint
ReDim thecircles(0 To pts) As agent
'fills in the array thecircles with the randomly scattered circles
ReDim limbocircles(1 To pts) As agent
ReDim relations(1 To groups, 1 To groups) As Boolean
'2d array of compatible agents
ReDim cols(1 To groups) As Integer
Dim rel As Integer
 rellies = "c:\stuffs\relationships.txt"
   Open rellies For Input As 2#
     For i = 1 To groups
        For j = 1 To groups
              Input #2, rel
              relations(i, j) = (rel = 1)
        Next j
     Next i
   Close #2
```

CODE 2 SOCIAL PREFERENCE ARRAY

3.3 Voronoi Diagram

Voronoi Diagram is coded according to the following steps:

1. Find all voronoi vertices (vv) which are the centre of circles defined through sets of three nearest points (points are supplied by the locations of all agents), as such that there are no other points inside these circles.
2. Draw a line from a vv to the nearest vv, and do this to all vv.
3. At the perimeter, a vv will be connected to an imaginary vv which defined as an extended line from that vv to the middle of the two points which defined that vv.
4. Then each polygon can be created as voronoi cell, where each will contain one location of the corresponding agent to that cell.

Voronoi Diagram procedure is called in at appointed loop of step. In the code below it is called in at each 60th step. The user enters how long the simulation will run by entering how many steps the simulation would be. If they enter 1200 steps, then Voronoi Diagram will be created at the 60th, 120th, 180th, 240th, etc. until the 1200th steps. This will correspond to a production of 20 pieces of Voronoi Diagram and thus 20 pieces of individual spatial configurations.

```
If counter Mod 60 = 0 Then          'Mod 50 = 0 means every 50-th to get Voronoi Diagram
      gestalt counter, ci, cj        'gestalt is the subroutine where Voronoi Diagram procedures contained in it
          If savestuff Then
             ThisDrawing.SendCommand ("_vscurrent" & vbCr & "R" & vbCr)
                   Set allpolys = ThisDrawing.SelectionSets.add("allofit")
                       allpolys.Select acSelectionSetAll
             ThisDrawing.Regen acActiveViewport
             ThisDrawing.SaveAs (pathname + Str$(counter))
             ThisDrawing.Export (pathname + Str$(counter)), "BMP", allpolys
                      allpolys.Delete
           End If
                    thelay.Lock = True
                    boundarylayer.Lock = True
             ThisDrawing.SendCommand "_erase" & vbCr & "all" & vbCr
             ThisDrawing.SendCommand vbCr
                    thelay.Lock = False
                    boundarylayer.Lock = False
        ThisDrawing.Regen acActiveViewport
End If
```

CODE 3 VORONOI DIAGRAM PROCEDURE

3.4. The Main Procedure

The program in common language can be described as follows.

1. At t = 1 agents created and then move randomly in space.
2. At t = 2 agent's next step is triggered by predefined conditions such as three nearest neighbors, then agents take action according to SPM
3. At t = 3 if the relations between self and the other agent in SPM = 1 then this agent heads towards, and stays in slow down loop.
4. At t = 4 the Voronoi Diagram is called at a specific step and divides the space.
5. At t = 5 the Voronoi Diagram colours the cells corresponding to the colour of agents.
6. At t = 6 the Voronoi Diagram picks up any trapped Y spaces, then move them randomly back into the universe.
7. The program loops back to t = 1

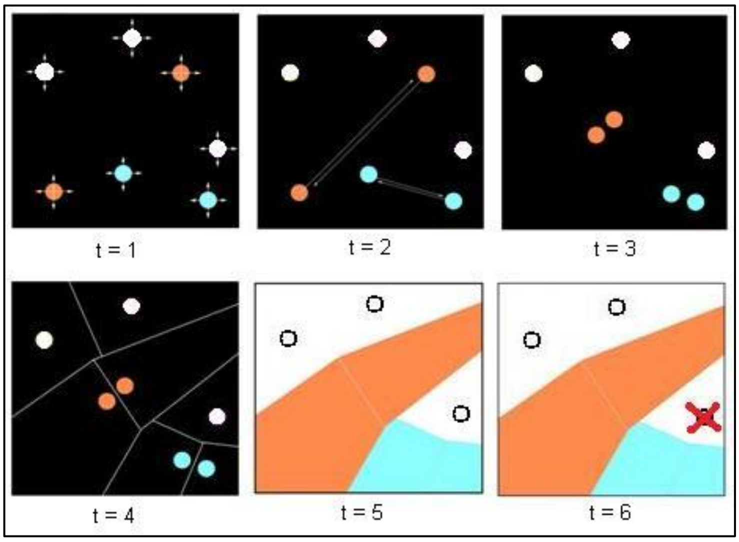


FIGURE 12 SPATIAL LANGUAGING MAIN PROCEDURE

3.5. Program Setup

3.5.1. Global Setup

The simulation was built on AutoCAD software, thus it needs to run on AutoCAD version 2004 onwards with Visual Basic programming platform installed.

It needs a new folder to be named “stuff” in the root drive C:/. Otherwise this new folder can be called under different name but one needs to modify the code within the subroutine as shown below.

```
Set thelay = checkforlayer("circle_layer")
Set boundarylayer = checkforlayer("boundary") 'dont erase boundary as well as circles
    seed = val(InputBox("seed", "any numbers", 978345210))
    ticks = val(InputBox("how may times round the block ", "steps", 500))
           'this is how many steps to run program which is proportional to    mod n =0, so adjust as necessary
    Rnd (-1)
Randomize (seed)
       savestuff = (InputBox("want to save stuff", "y, folder c:\stuffs") = "y")
       If savestuff Then
            pathname = "c:\stuffs\" + InputBox("type name for saved drawings", "naming the drawings")
            Open globaldatapathname For Append As #1
              Write #1, "counter ¦ number of occupied cells ¦ total area of occupied cells"
            Close #1
        End If
     thelay.Lock = False
     boundarylayer.Lock = False
     groups = InputBox("how many groups ?", "different groups represented by different colours", 6)
     pts = InputBox("how many people?", "size of population", 70)
ReDim originalpoints(1 To pts) As mypoint
ReDim thecircles(0 To pts) As agent    'fills in the array thecircles with the randomly scattered circles
ReDim limbocircles(1 To pts) As agent
ReDim relations(1 To groups, 1 To groups) As Boolean          '2d array of compatible agents
ReDim cols(1 To groups) As Integer
Dim rel As Integer
     rellies = "c:\stuffs\relationships.txt" ‘make sure text file called relationships.txt is available in c:\stuff
    Open rellies For Input As 2#
```

CODE 4 PROGRAM SETUP

In the folder stuff there should be 2 text files; relationships.txt and data.txt, and dwg file; basepoly.dwg. Relationships.txt contains SPM data and data.txt should be a blank file. Basepoly.dwg contains specific drawing layers that make up a bounded universe in a size of 200 x 200 unit of space. The size of 1 agent is unit of space.

Data.txt will be written with statistic information which currently contains three parameter: *counter*, *totalspacetypes*, *totalareas*. *Counter* is the step at which the Voronoi Diagram is called, *totalspacetypes* is how many agents are currently occupying, and *totalareas* is the size of occupation.

From these two numbers we can tell if the simulation had finished with all occupiers had occupying the space or not, and if the total occupation match to the percentage set on for the creation of the visitors. If the *totalareas* shown the number corresponding to 80% of the size of the universe when the set for visitors is 0.2, then it confirms that the program compiled accordingly.

3.5.2. Run the Program

These are the steps to run the program:

1. Open AutoCAD, then open basepoly.dwg
2. Type in AutoCAD command line vbaman and search directory to open the program spatial_languaging.dvb. Once vb window opens, go back to AutoCAD drawing window.
3. Type in AutoCAD command line vbarun, and choose run macro.
4. Enter all ABM input parameters as required. Choose save file if want to produce spatial configurations, otherwise SL will run ABM and create Voronoi Diagram without saving image files of spatial configurations. Saved images can be found in C:/stuff

Note: Basepoly.dwg should not be saved at the end of each simulation. Basepoly.dwg should be re-open to run new simulation with different ABM input parameters.

3.6. Programming Results

3.6.1. Program Execution

Spatial Languaging program executes simulations which are corresponding to the proposed mechanics of isomorphic mapping of ABM into Voronoi Diagram. It generates ABM with correct number of agent's color as entered in the beginning of the simulation and these agents randomly placed and then move about the universe (Appendix 3). It also executes SPM correctly (Appendix 2). It generates spatial configurations in the form of dwg and bmp files. It also manages data output in data.txt (Appendix 3). All input parameters for ABM and embedded parameters (which have to be access from within the code in order to modify them, see Appendix 4) are sensitive to changes where different value will resulted in different ABM behaviours and different spatial configurations.

3.6.2. Graphic Output

The graphical outputs correspond to the parameters as expected. The proportion of occupier gives results to the colored occupied cells with exactly the same spatial proportion; for example generation of 80% occupiers will result in 80% colored space (Appendix 3). The integrated society where SPM contains more relations with value of 1 and combine this with larger occupier percentage will fill the occupational space quicker.

Interestingly the emergent geometry demonstrated forms analog to that of real settlements, where segregated SPM creates small scattered spaces, and the integrated creates dense spatial configurations. All spatial arrangements of these spaces and their dimensions were entirely an emergent property of the simulation. How space is being used in terms of movement by its inhabitants give results to not only the arrangements of "rooms" but also its dimensions. It is a phenomenon that can be described as self-organizing spatial configurations which produce arrangement of space according to its uses with the corresponding dimensions of each space simultaneously.

The population density coupled with occupier and visitor ratios are fundamental to the kind of space which would emerge. From the observer's viewpoint, a population which is made of 50% occupier agents and 50% visitor agents seems to generate interesting graphics compared to other settings (see Appendix 1 Graphic set 4, 5, and 6). These graphic sets comparatively produce same effect of perception although different social matrix was applied to each set. This might be explained as the gestalt effect of background and foreground. Furthermore some graphics show the emergent of beady ring phenomenon (Hillier, 1984, p.59)

The underlying gestalt effect computed by the Voronoi diagram follows the theoretical wholeness of a spatial configuration and fully implemented under these principles into each of the graphics. This is done by responding to a non-conforming configuration, such as a trapped Y space (movement space) in the middle of occupied spaces. This step enables a continuation of movement space over the whole space created without having to coded-in a continuous movement space.

3.6.3. Version Development

The ABM with Social Preference Matrix (SPM) and the Voronoi diagram manages to produce spatial configurations. The early version of ABM setup did not create visitors and all agents were identified as occupiers. The results did not conform to syntactical configurations because there were no Y spaces or movement spaces created. All agents in the early version were eventually found their preference and thus all cells generated were colored/occupied cells.

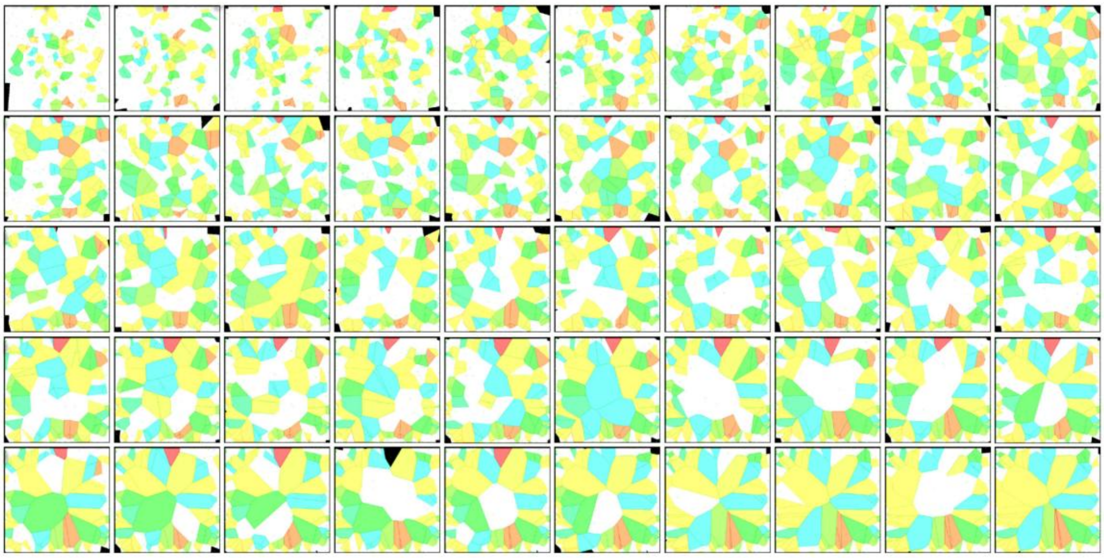


FIGURE 13 A SET OF RESULTING SPATIAL CONFIGURATIONS (HOVER OVER PICTURE TO FOLLOW LINK HTTP://WWW.YOUTUBE.COM/WATCH?V=ME0CI5SDMUC)

The series of images above is a result of simulation based on the second version ABM setup and it is the program submitted for this thesis. At the end of the first row, more agents had grouped together indicating almost all agents found their preferences. The second row and the third row are images which shown emergent beady rings and these configurations are more conforming to syntactic configurations in comparison to the first and the fourth row. Halfway over the third row and the fourth row show that configurations had reached equilibrium and all these had more or less the same syntactic configurations and more or less the same total area where differ only in their cell shapes.

Is this mean Y space or movement space is somehow a form of an axiom? Considering there is no starting point of Y space indicated from the beginning of the simulation, these movement spaces could be considered as indirectly axiomatic. The present of equilibrium configurations seems to suggest that there is a maximum point in time to which all occupier agents would map into X spaces where these occupied spaces. At this point there will only be minimal varieties in geometry (shapes of cells and their dimensions) of a stable topology.

4. Discussions

4.4. Self-organizing Spatial Configurations

A model is an imitation, which is produced from simplification of the original. The convention states that the original is always a lot more complex than its model (Cilliers, 1998, p.10). In architecture, a model is part of the design process and thus the simplified precedes the complex actual. Christopher Alexander (1964) explains that design is a synthesis of form which production of form is about finding a good arrangement and proper relations of the parts. The dilemma is that how is the search for good arrangement and relations of the parts using simplified concepts would match the complexity of the real?

Mechanically, design is an act applying a *formula*, or recipe, to manifest *form* using known parts via specific arrangements or relationships which believe to give good result. Science as we know it today was brought to such advancement because scientists can explain subjects in a mechanic sense. The specific study which puts forward the possibility to explain anything in terms of its mechanics is known as the field of System Theory and its newest form is known as the field of Dynamical System Theory. Thus, architecture also been subjected to the mechanic frame of thinking, and there would be a mechanic interpretation of the production of architecture through the medium of modelling.

Christopher Alexander showed that eventually design process will result in some degree of fitness of the form to its context. What search method prescribed above which could produce form that is a 'good fit' for its complex context? Stanford Anderson (1966) criticized that there is too much oversimplification of the mechanical method that does not regard the fact that a good arrangement of parts varies from time to time, because context (all parameters that should be taken into account in the production of form) is believed to always in states of changes. There is not simply a set that will guarantee a good fit for all times. Thus Anderson argued for the lack of sensitivity of problem-solving method as one as proposed by Alexander. He then suggested that a mechanical method could be used to produce complex objects such as architecture only when it is responsive to the initially undefined problems and equally responsive to the infinite of potential problems that come during, and later on the process.

This means the challenge is to define a mechanic of design that adapts. Throughout such a process, it should be made possible to introduce new parameters of good fit, so that the eventual fitness is an emergence of all possible interactions within the universe until that specified time frame within the formal evolutionary process. If one steps back to the notion of a model, it is intriguing to find out why a formula manifested from deducing a complex, can be expected to produce something quite complex. John Frazer (1995) uses evolutionary methods for the production of architectural form and demonstrated that using nature's recipe of genetics, the aim to produce an object that is far more complex than its formula can be achieved. Genetics is widely accepted as the mechanical process for the ever increasingly complex forms. But how does this evolutionary process become responsive to the ever changing context?

The phrase "form that fits its context" is an architectural ideal. Taking all the physical and environmental, and all other possible complex problems into account and process it; then let's suppose it is a dwelling design proposed for a happy couple. This is initially called "a flat". Later in time, it will need to be called "a house" to serve the couple and their children. And later, it may turn back as "a flat". It is not possible for the first "flat" be the same as the later "flat", because it is in a different time with a different set of complex problems. There is also no guarantee that the initial physical and environmental setting will be the same. The "form that fits a context" could only be true for an architectural production only when it is about a production at any particular point in time. Genetics on its own cannot deliver a "fit to context" if it has not a way to take into account whatever required at different times. This is what is required for the mechanics of adaptation, the ability to enable the processes of interactions at micro level within all the new systems that create the context of that form which feeds it back iteratively into the form.

Such setting means that any form along an infinitely continuous process will need to take information from its context and to use it. The process means invoking new arrangements, new relations between parts; and perhaps even new parts to be defined. The new form itself then will invoke feedback which is then fed again into the process. In this way, the context itself evolves, since it requires information to be processed and action to be taken; just as what was needed for the production of form. Accordingly, it is not simply an evolution of form, but it is a dynamic self-organizing system with interacting systems of form and its context, are changing through times and thus increasingly becoming more complex. This self-organizing system is characteristically infinite, which is guaranteed to continue forever as long as the feedback mechanisms between these they are still going on.

Essentially, the self-organizing system of space and people as an infinite loop functions between form and context. John Holland (1998, pp.125-141) named such a scheme as the Constraint Generating Procedure (CGP). He presented the concept as the foundation of emergent phenomena in complexity science. Agent-based Model is the significant element in the construction that leads to emergence behaviour (*ibid.*, pp.116-118 and Ferber, 1999, pp.15-16). ABM structures interaction between parts locally. It enables synchronous processing of information thus possible to treat changes gradually. However, construction of each ABM will need to find its own mechanics of self-organization. There is no specific recipe or over-arching procedure of self-organizing mechanism.

The work of Ashby, Beer and Pask (Cariani, 1993) suggests that the mechanics of emergence and its underlying self-organization is a paradox. Emergence can only be shown to occur when the structural outcome is not dependant on the details of the epistemic definition. Once again, there is no definition for specific mechanics at micro level. However, developing a computational method is fundamentally constructing basic or micro elements of a system and applying structural interactions between them. The feedback mechanism between interacting systems is another layer of structural interaction which consists of interactions between different elements of different systems. When *emergence* is literally happening, is it then possible then to identify the mechanics underlying it? The clue would be in the interactions at these different levels, and the mechanism of emergence can be identified when one knew how these systems interact.

The complex system study of biology came up with some clearer sense of what are interactions between dynamic systems. Chilean biologist, Maturana explained that interaction is essentially a structural coupling which is the state of coordination of *coordination of* actions (Maturana & Varela, 1980, pp.xx-xxi). This could be interpreted as a mechanism of arrangement and rearrangement at micro level within infinite iteration of feedback out of (at least) two systems that are structural coupling. The program Spatial Languaging is an application of such mechanics which in this case is based on mutual perturbation of architectural forms in a social context. It demonstrates that system such as configuration of people (as people would have used space) can be developed as an open-ended interactive system. An open-ended interactive system would have to have a flexible structure so that structural coupling between its micro elements and other system’s micro elements; i.e. the individual parts of the system can be mutually modified (*ibid., p.107)*. The flexible structure seems to suggest that the most basic relations come from the most basic element such as binary system, from which would set the ground for the most wide combinatorial possibility.

Space Syntax theory (Hillier & Hanson, 1984) is a true form of deduction, synthesis, and formulation of space hence it provides elements which is truly basic for the construction of Spatial Languaging program. Space Syntax is the structuralist approach to architecture, very close to Nietschzian primordiality of space as described by Lefebvre (1974, p.22) that space can only be defined as either occupied or unoccupied; that is in binary form could be expressed as a 1 and 0 or X and Y. The space syntax theory also provides binary relationships between the occupied and the unoccupied space. It opens a wide combinatorial possibility because rules of relationships only apply locally. As long as there is a way to feed this system where individual space can be defined randomly, the possible combination of cell spaces that creates the overall spatial configuration is always indefinite at the time of initialisation.

Thus, ABM would provide isomorphic feeds for the mapping of spatial configurations using the locations of its agents in space. An agent is characteristically a randomly mobile and thus inherently unpredictable in terms of their location in space. The relations between agents in ABM could only be defined by the types of interactions. For Spatial Languaging program, the type of interaction is as per preference which are either 1 or 0. When an agent is near enough to three other agents it would go towards the one it prefers to be local with. The Space Syntax theory seems to strongly suggest an isomorphic mechanism between spatial configurations and configurations of people. Space can only be defined by those who are currently in that space, and thus creation of space is identified with a being to be in a certain place in space.

This creates a form of 'fit'. Which specific configuration of people corresponds better to a specific configuration of space? Both Hillier (1996) and Alexander (1964) seems to agree that configuration is a form of combination of binary form of 1 and 0 (i.e. x space and y space). Combinations of these occupied and unoccupied spaces could result in a complex (of space) which at the same time (*when* it *interacts* with a configuration of people) fit into the context of how it can be use by particular socio-spatial configurations. So thus my understanding is that an experiment to produce complex spatial configuration can be done using principles introduced by Space Syntax with a mutual perturbing framework as the mechanism.

4.5. Epistemic Autonomy

For many researchers in the field of design computing, the goal would be to develop design generators which as much independent of human as the creator of space (who draw it) and to enlarge the capacity of the processor (which process it)to design. This aim backs up with the realization of the contemporary tools and thoughts about how we could design. The approach implemented in Spatial Languaging program is to achieve this kind of self-organization. It aims to process design in a self-organizing way.

According to Pask (Cariani, 1993), in a self-organizing set there should be some degree of epistemic autonomy. Literally this means there is no such thing as total autonomy, where a result comes from nothing. Computer modelling requires input to process, and processing methods keeps on advancing through times. In Spatial Languaging program, multiple processing is done by the Agent-based Model (ABM), and there is also a feedback process between ABM and Voronoi Diagram where actual changes of forms of different stages could be visualized. Although it is not possible to have a total autonomy, epistemic autonomy is suggested to be fundamental to a self-organizing system. This could mean that an epistemic autonomy should not preclude the inherent isomorphic quality of all systems involved. As such, the spatial configurations and the configurations of people would be isomorphic and that epistemic autonomy of these configurations only applied to the emergence of geometry, morphology and topology of the configurations.

Secondly, it would also mean that there is a degree of autonomy in regards to structural emergence. The structure of spatial configurations within this project is a dual of the social structure manifested in the use of space and then mapped so it is visible by the Voronoi Diagram. This type of structural emergence is different to axiomatic structure in at least two aspects; axiomatic structuring uses the structural elements as part of the emerging structure, and self-organizing structure does not. Axiomatic structure starts with one of its own structural element, and grows into larger structure by combining other structural elements with that axiom. This kind of emergence was incorporated in the Alpha Syntax (Coates, 2010, pp.153-157). Contrastingly, self-organization give rise to its own structural elements manifested through the map with underlying mapping conventions involved in it. Spatial Languaging program produces structures which are *manifested*, as opposed to *growing*.

Furthermore the other aspect is of the convention itself; the mutual perturbation is a map which relies on its mapping convention, whilst the axiomatic structure uses a convention to emerge. The mapping convention is the key to the main co-morphogenetic process since it enables feedback mechanisms. In agreement with the contemporary philosophy of computer modelling each interacting part has its own formative structure (Cilliers, 1998, p.10). In analogy spatial configurations are theorized by the spatial sciences, whilst social interactions are similarly theorized by the social sciences.

Specific mapping convention accommodates the emergence of those particular structures recognizable as spatial configurations and as social interactions because by way of feedback both systems will keep changing. Otherwise its structural parts would not comply with what is recognizable as spatial configurations or social interactions and therefore the development of Social Preference Matrix (SPM) is inevitable because the matrix feeds the system with a *would be* social structure, it is the mapping convention of who should interact with whom.

Self-organization implies a set where some elements already exist, in which there is some degree of epistemic autonomy of the form. It is not possible, working within a self-organizing framework to get a total autonomy. The epistemic autonomy achieved within the emerging spatial configurations of this project is gained through the isomorphic understanding of spatial structure and social structure, which is made possible by ABM and the spatial configurations is an emerging structure brought out by the Voronoi Diagram.

The complexity of architectural design means there are more isomorphic layers within the self-organization of form where all systems are simultaneously self-perturbing. Thus it is interesting to know how to set up these interacting layers; is there a hierarchy where social structure and spatial structure should be within this set of layers? Or perhaps are these layers inter-connected real time as such only the required interactions brought in the specific layer forward to co-evolve at any particular time?

Post-structuralist view of modelling (Cilliers, 1998, pp.58-88) requires a particular theory to be fundamental to a model. Thus if there are more architectural theories which involves different systems which structurally coupling then is it possible to construct more self-organizing design process? From these experiments there would be new architectural theories as a result of observing these self-organizing simulations.

4.6. Re-iterating Distributed Representation for Design Process

FIGURE 14 THE DOG PICTURE BY DAVID MARR 1982 (HOVER OVER PICTURE TO FOLLOW LINK HTTP://EN.WIKIPEDIA.ORG/WIKI/GESTALT_PSYCHOLOGY)

Distributed representation is an approach to the problem of representations for building a complex system, where the complex system is built as a tool to learn about that complex system (Cilliers, 1998, pp.12-13). Design process in some way is learning about the environment and the requirements of the built environment. Considering the availability of ABM and the understanding of self-organizing systems, then it should be possible to represent the environment more thoroughly than ever before. Therefore, it seems this is the right time to embrace systems thinking and to apply new techniques to process design.

Agent-based Model (ABM) is a typical framework for distributed representation because its characteristics are sufficient in enabling emergence phenomena. Self-organization is a process where a simple system can develop a complex structure from unstructured beginnings (*ibid.*, p.12). Distributed representation of design process seems feasible only via ABM because it is characteristically parallel processing.

ABM is made out of many individual units but these units by themselves are much less meaningful without the emergence of global structure from which the observer can perceived (or learned about) it. Consequently, the observer is an essential part of a complex system that made out of distributed representation. ABM is the dual of spatial configurations in order to produce them. The interacting network of ABM, spatial configurations and the observer embodies the principles of distributed representation of design. A missing piece within the network would mean a failed distributed representation of design process.

Distributed representation of design requires a strong over-arching design theory. Theory is a form of existing knowledge about complex system, and it precedes the model by providing the elements of the model (*ibid*., p.130*)*. An observer would learn from the model as they perceived the model and can decide which behaviours already known and which are emergent. There is nothing new if the existing is not identifiable. Distributed representation of design process would use the elements provided by design theory in order to enable the emergence of new kinds of spatial designs.

Distributed representations of design not only have a specific interacting network of ABM – space - observer, but also produce a continuity of evolving results. Designs produce by utilizing distributed representation will result in series of spatial designs and all these results are individual designs. Design is then a complex system which as a whole and as its parts is always undergoing repetitions, reiterations, and transformations because this is the means of design to becoming into being; i.e. when design is actually identified by the observer. These are the specific characteristics of distributed representation of design process; autonomous, self-organizing, cognitive and observer-related. These are designs which are identified as the second-order characteristics.

Distributed representation of design is also characteristically a continuously co-evolving system because it is based on isomorphic structural coupling. Society and space are isomorphic in the way it will influence each other; anything happened to an individual element in one system will affect element or elements in the other system which will eventually feedback iteratively. The mechanism to describe the relations between the model and the observer is *interactive*, from which the series of operating procedures can be explained using an autopoietic framework. Autopoietic framework for this kind of interaction is known as *syn-referential*, i.e. the *coordination of coordination of* actions, where innate structure of both the model and the observer is coordinating with each other over time(Maturana & Varela, 1980, pp.9-11). This is a structural coupling and thus isomorphic, embodies in autopoietic term *languaging*.

Spatial Languaging is a program which generates spatial configurations. Spatial Languaging is also proposed as a term to describe a general framework in which the problematic representation and systematic understanding of architecture and design process can be addressed and sufficient simulations of design process as a complex can be developed. *Languaging* in itself is a specific term in relation to SL program, which points to the corresponding Space Syntax theory that has been interpreted in the identification of the mechanics of interactions between configuration of space and configuration of people.

Interactions or *languaging* in autopoiesis terminology is an infinite recursive coordination of coordination of action. This essentially means that to respond to the received information from outside its own system, it would have to be flexible structurally and capable to re-arrange its parts accordingly in response to that information. The internal changes will be feedback to the outside system, where there will be more information available for it to respond. The term *languaging* invokes sufficiency so that *interactions* between configuration of space and configuration of people are meaningful. The recursive coordination of coordination also defines the method of distributed representation of design process between the interacting systems in it.

4.7. Synthetic Gestalt

One of the main findings of this study is that feedback mechanism that work for analyzing parts of a system seems to work at a global level. Spatial Languaging program simulates a similar phenomenon of global observation among other observable behaviours it produced. The agents are individuals with simple embedded features to act locally, but Voronoi Diagram process and analyze these agents as a whole entity. This is called a *synthetic gestalt* phenomenon.

What happened in the program was that where an agent creates a cell of space which does not conform to the local neighbourhood rules, then this agent will be relocated in the next step. This affects agents as a form of feedback perturbing into the system of agents without intervening with the epistemic autonomy of the program to produce permeable configurations. It indicates that a local or bottom-up network could send and receive feedback by employing a global or top-down processor. This is significant when one believes a machine is made of parts and they all connected in such a way with feedback mechanisms that help the machine to restructure its parts over time. If architecture and or design process is such machine, the existence of a top down processor like synthetic gestalt in an evolutionary scheme could be the key to hold all the interacting elements of design process.

Synthetic gestalt could be existed within all self-organizing systems, it could be use to explain the process known as homeostatic state. This is a state where elements of a system that were put together could stabilize and reach equilibrium which then could demonstrate some recognizable function (Cariani, 1993). Distributed representation of design process is a self-organizing set which will always require an element within it which will act as synthetic gestalt.

Where elements of design identified by a strong over-arching design theory such as Space Syntax theory, other theories related in the actual construction of built environment would be the one which will be the synthetic gestalt of that model.

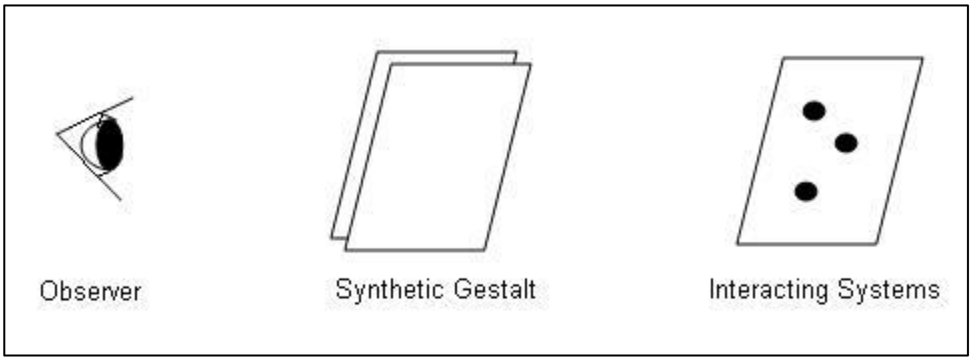


FIGURE 15 LAYERS OF DISTRIBUTED REPRESENTATION

Therefore, the system of structural constructions embodies, and itself is the embodiment of synthetic gestalt. This is the main principle that enables automating design in a self-organizing way. If spatial configurations theory enables the processing of many spaces into actual spatial configurations, then other theory and other system will enable the processing of other aspects of space they relate to.

Bibliography

Abraham, Matthew, 2000 “The Critical Idiom of Postmodernity and Its Contributions to an Understanding of Complexity. A review of Paul Cilliers, Complexity and Postmodernism: Understanding Complex Systems”. London: Routledge, 1998. in *Postmodern Culture* Volume 10, Number 2, the Johns Hopkins University Press

Akin, Omer, Dave B and Pithavadian S. 1992 “Heuristic Generation of Layout (HeGeL) based on a paradigm for problem structuring” in *Environmental and Planning B* 19 pp. 33-59

Akl, S.G., Lyons, K.A., 1993, *Parallel Computational Geometry*, Prentice Hall: New Jersey

Alexander, C, 1964, *Notes on the Synthesis of Form*, Harvard University Press: Cambridge, Massachusetts

Alexander, C., Ishikawa, S., Silverstein, M., 1977, *the Pattern Language: Towns, Buildings, Construction*, Centre for Environmental Structure, University of California: Berkeley, California

Anderson, SO. 1966, *Problem-Solving and Problem-Worrying*, Unpublished lecture manuscript given at Architectural Association, London, March 1966 and ACSA Teachers Seminar, Cranbrook, June 1966

Axelrod, R, 2005, Agent-based Modelling as a Bridge between Disciplines. In *Handbook of Computational Economics Vol. 2: Agent-based Computational Economics, Handbooks in Economic Series*. North-Holland Eds Kenneth L. Judd and Leigh Tesfatsion. Forthcoming available online http://www-personal.umich.edu/~axe/research/ABM_Perspectives.pdf last accessed 15 March 2006

Bayazit, N., 2004, “Investigating Design: a Review of Forty Years of Design Research”, *Design Issues* Volume 20, Number 1, Winter 2004, MIT Press: Cambridge Massachusetts avalaible online last accessed 6 February 2006 http://mitpress.mit.edu/journals/pdf/desi_20_1_16_0.pdf

Baybars I, 1982, "The generation of floor plans with circulation spaces" Environment and Planning B 9 445-456

Batty, M., Longley, P., 1994, *Fractal Cities: a Geometry of Form and Functions*, Academic Press Professional :San Diego

Baudrillard, J., 1994, *Simulacra and Simulation*, trans Sheila Faria Glaser, The University of Michigan Press

Baudrillard, J., 2003, *Mass, Identity, Architecture. Architecture Writings of Jean Baudrillard*, Francesco Proto (Ed.), Wiley Academy, England

Barney, B., 2006, *Introduction to Parallel Computing*, for Lawrence Livermore National Laboratory, a website operated by University of California for the Department of Energy’s National Nuclear Security Administration, California, USA. Available at http://www.llnl.gov/computing/tutorials/parallel_comp/#Whatis last accessed 1 September 2006

Bertalanffy, L V, 1962 “General System Theory - A critical review”, *General Systems*, 7:1-20

Bertalanffy, L V, 1968, *General System Theory Foundations Development Applications*, Penguin Books, England

Bloch C J, 1979, "Catalogue of small rectangular plans", *Environment and Planning B* 6 155-190

Boissonnat, J-D., Yvinec, M., 1998, *Algorithmic geometry*, Cambridge University Press: Cambridge

Bonabeau, E, Dorigo, M. & Theraulaz, G., 1999, *Swarm Intelligence From Natural to Artificial Systems*, Oxford University Press: New York.

Bridges A H, 1979, "Analysis in architectural design" architectural design" in *PArC 79 Proceedings* (Online Publications, Pmner, Middx; AMK, Berlin) PP 175-185

Bryne D., 1999, "Book Review Paul Cilliers Complexity and Postmodernism: Understanding Complex System", *Journal of Artificial Societies and Social Simulation* Volume 2 Issue 2 available at http://jasss.soc.surrey.ac.uk/2/2/review1.html Last accessed 27 August 2006

Cariani, P., 1993, "To Evolve an Ear: Epistemological Implications of Gordon Pask's Electrochemical Devices", *Systems Research*, 10 (3):19-33, available at http://homepage.mac.com/cariani/CarianiWebsite/PaskPaper.html last accessed 11 August 2006

Chomsky, N., 1957, *Syntactic Structures*, The Hague, Muton

Cilliers, P., 1998, *Complexity and Postmodernism: Understanding Complex Systems*, Routledge, London

Coates, P., 2010, *Programming.Architecture*, Routledge, London

Corbusier, 1923, *Towards a New Architecture*

Cousin J, 1970, "Topological organization of architectural space" *Architectural Design* 10, pp. 491-493

Craik, K.J.W., 1943, "The nature of Explanation"

Cullen, P.-A., 2000, "Contracting, Co-operative relations and Extended enterprises", *Technovation*, 20 (7), pp. 363-372.

Damski, Jose C. and Gero, John S. 1997, "An Evolutionary Approach to Generating Constraint-Based Space Layout Topologies", *CAAD Futures 1997* [Conference Proceedings / ISBN 0-7923-4726-9] München (Germany), 4-6 August 1997, pp. 855-864

Dawkins, R., 1986, *the Blind Watchmaker*, Norton and Company Inc

Derrida, J., 1976, *Of Grammatology*, trans. Gayatri Chakravorty Spivak, Baltimore & London: Johns Hopkins University Press

Eastman C M, 1970, "Representations for space planning" *Communications of the ACM* 13 242-250 Abstraction as a tool of automated floor-plan design

Eliasmith, C., 2004, *Distributed Representation Dictionary of Philosophy of Mind* last edited 2004, last accessed 11 August 2006 available at http://philosophy.uwaterloo.ca/MindDict/distributedrepresentation.html

Eastman C M, 1972, "Preliminary report on a system for general space planning" *Communications of the ACA* 11 15 76-87

Epstein, J. M., & Axtell, R. L., 1996, *Growing Artificial Societies: Social Science from the Bottom Up*, The MIT Press.

Ferber, J. 1999, *Multi-agent Systems an Introduction to Distributed Artificial Intelligence*, Addison-Wesley, London

Flemming U, 1977 *Automatisierter Grcrndrissentwurf. Darstellung, Erzeuguiig und Dimensionierung Vorl dicht gepacktcn, rechtwinkligen Fla'chenanordnungen* doctoral dissertation, Department of Building Planning and Construction, Technical University of Berlin, Berlin

Flemming U, 1978a. "Representation and generation of rectangular dissections" *Proceedings of the Fifteenth Design Automation Conference* (ACM-SIGDA/IEEE-DATC, New York) pp 138-144

Frazer, JH., 1995, *an Evolutionary Architecture*, Architectural Association: London

Garson, J., 2002, "Connectionism", *The Stanford Encyclopedia of Philosophy (Winter 2002 Edition)*, Edward N. Zalta (ed.), available at http://plato.stanford.edu/archives/win2002/entries/connectionism/ last accessed 21 August 2006

Galle P, 1981, "An algorithm for exhaustive generation of building floor plans" *Communications of the ACM* 24 813-825

Gero J S, 1977, "Note on `Synthesis and optimization of small rectangular floor plans' of Mitchell, Steadman, and Liggett" *Environment and Planning B* 4 81- 88

Gero, JS and Damski, J, 1997, "A symbolic model for shape emergence", *Environment and Planning B: Planning and Design* **24**: 509-526.

Giddens, A, 1984, *The Constitution of Society. Outline of the Theory of Structuration*, Polity, Cambridge

Giedion, S., 1948, *Mechanization Takes Command*, Oxford University Press: Oxford

Gilbert, N., Troitzsch, K.G., 1999, *Simulation for the Social Scientist*, Open University Press, Buckingham

Gilbert, N., 1995, "Emergence in Social Simulation", in *Artificial Societies the Computer Simulation of Social Life*, Editor Nigel Gilbert, Rosaria Conte, UCL Press: London

Gilleard J, 1978, "LAYOUT-a hierarchical computer model for the production of architectural floor plans" *Environment and Planning B* 5 233-241

Goldspink, C., 2000, "Modeling Social Systems as Complex: Towards a Social Simulation Meta-model", *Journal of Artificial Societies and Social Simulation* Vol. 3 No. 2 available at http://jasss.soc.surrey.ac.uk/3/2/1.html last accessed 9 September 2006

Goldspink, C., 2002, "Methodological Implications of Complex System Approaches to Sociality: Simulation as a foundation for knowledge", *Journal of Artificial Societies and Social Simulation* Vol. 5 No. 1 available at http://jasss.soc.surrey.ac.uk/5/1/3.html last accessed 9 September 2006

Grason J., 1968, "A dual linear graph representation for space-filling location problems of the floor plan type" in *Emerging Methods in Environmental Design and Planning* Ed. G T Moore (MIT Press, Cambridge, MA) pp 17U-178

Hayes, B, 1999, "E-pluribus Unum Simulating Herds. Flocks and Schools". *American Scientist Online* January-Volume 87 No. 1, available online http://www.americanscientist.org/amsci/issues/Comsci99/compsci1999-01.html last accessed 3 March 2006

Hejl, P.M., 1984, "Towards a Theory of Social Systems. Self-Organization and Self-Maintenance, Self-Reference and Syn-Reference". IN H. ULRICH, G. J. B. P. (Ed.) *Self-Organization and Management of Social Systems*. New York

Heylighen, A, 2000, *In Case of Architectural Design: Critique and Praise of Case-based Design in Architecture*, Doctoral Thesis, Katholieke Universiteit Leuven: Leuven, last accessed 6 February 2006 http://www.alnresearch.org/Data_Files/dissertation/full_text/heylighenPhD.pdf last accessed 13 January 2007

Heylighen F., Joslyn, C. 2001, "Cybernetics and Second Order Cybernetics", in: R.A. Meyers (ed.), *Encyclopedia of Physical Science & Technology*, Vol. 4 (3rd ed.), (Academic Press, New York), p. 155-170

Heylighen, F, 1998, *What are Cybernetics and Systems Science? Basic Concepts of the System Approach*, Principia Cybernetica Web http://pespmc1.vub.ac.be/SYSAPPR.html last accessed 6 February 2006

Hillier, B, 1996, *Space is the Machine*, Cambridge University Press: Cambridge

Hillier, B., Leaman, 1978, "Space Syntax", *Environment and Planning B*, Vol. 3

Hillier, B., Hanson, J, 1984, *The Social Logic of Space*, Cambridge University Press: Cambridge

Hofstadter, D.R., 1979, *Godel, Escher, Bach: an Eternal Golden Braid a Metaphorical Fugue on Minds and Machines in the Spirit of Lewis Carroll*, The Harvester Press Ltd, London

Holland, J.H., 1998, *Emergence from Chaos to Order*, Oxford University Press: Oxford

Kaisersrot, 2001, *Design Your Own Neighbourhood*, http://www.kaisersrot.com/kaisersrot-02/2001_DesignYourOwnNeighbourhood.html last accessed 13 January 2007

Kaisersrot, 2003, Madestein, Den Haag (NL), http://www.kaisersrot.com/kaisersrot-02/2003_Madestein,_Den_Haag_%28NL%29.html last accessed 12 October 2010

Korf R E, 1977, "A Shape Independent Theory of Space Allocation", *Environment and Planning B,* 37-50

Krishnamurti, R., Roe, P. H. O'N, 1978, "Algorithmic aspects of plan generation and enumeration," *Planning & Design*, 5, 157–177.

Lawson, B., Park, S., 2000, "Asynchronous Time Evolution in Artificial Society Model", *Journal of Artificial Societies and Social Simulation* vol. 3, no. 1, http://www.soc.surrey.ac.uk/JASSS/3/1/2.html last accessed 13 January 2007

Lefebvre, H., 1991, *the Production of Space 1974*, trans.D. Nicholson-Smith, Blackwell, Cambridge.

Levin, P. H, 1964, "Use of Graphs to Decide the Optimum Layout of Buildings." *Architects*' *Journal* October 7

Levy, S., 1992, *Artificial Life – Quest for a new creation*, Penguin Books

Li, S.-P., Frazer, J.H. and Tang M.-X. 2000, A Constraint Based Generative System for Floor Layouts, CAADRIA 2000 [Proceedings of the Fifth Conference on Computer Aided Architectural Design Research in Asia / ISBN 981-04-2491-4] Singapore 18-19 May 2000, pp. 441-450

Luhmann, N., 1986, 'The Autopoiesis of social systems' in Geyer, F. and van der Zouwen, J. (eds.), *Sociocybernetic Paradoxes*, London, SAGE Publications

Maturana, H.R. & Varela, F, 1980, *Autopoiesis and Cognition Realization of the Living*, Reidel: Holland

Maturana, H., 1995, *The Nature of Time* available at http://www.inteco.cl/biology/nature.htm last accessed 18 August 2006

Maver, TW. 1970, *A Theory of Architectural Design where the Role of Computer is Identified*, Building Science:

Miller, I., 1993, *Self-organization in biological systems: The Holistic Patterning Process of Chaos and Antichaos* http://www.geocities.com/iona_m/ChaosTheory/chaostheory4.html last accessed 10 October 2008

Minsky, M., 1968, “Matter, Mind, and Models”**.** In: Minsky, M. (ed.): *Semantic Information Processing*. Cambridge, Mass.: MIT Press, S. 425-432

Mitchell, W.J., Philip Steadman, and Robin S. Liggett. 1976, "Synthesis and Optimization of Small Rectangular Floor Plans," *Environment and Planning B: Planning and Design* 3, no. 1 pp. 37-70.

Nehaniv, C.L. 1996, *Cellular Automata and Self-reproduction*, University of Aizu, Japan available at hyperactive book http://homepages.feis.herts.ac.uk/~comqcln//CM/ca.html last accessed 22 August 2010

Noble, J. & Biddle, R, 2004, Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications, pp. 112 – 115

Parsons, T., 1982, "Action, Symbols and Cybernetic Control." in Ino Rossi (ed.) *Structural Sociology.* New York: Columbia University Press

Pask, G., 1961, *an Approach to Cybernetics*, Hutchinson & CO: London

Pask, G., 1969, “The Architectural Relevance of Cybernetics”, *Architecture Design Journal*, pp.494-496

Petzinger, T. 2000, *Complexity Reading List,* http://www.petzinger.com/complexity.shtml last accessed 6 February 2006

Preparata, F.P,, Shamos, M.I . 1985, *Computational Geometry - An Introduction*. Springer-Verlag. 1st edition; 2nd printing, corrected and expanded, 1988

Prusinkiewicz, P., Lindenmayer, A., 1990, *The Algorithmic Beauty of Plants (The Virtual Laboratory)*, Springer-Verlag.

Resnick, Mitchell, 1994, *Turtle, Termites and Traffic Jams Exploration in Massively Parallel Microworlds*, MIT Press: Cambridge, MA

Reynolds, C. W., 1987, “Flocks, Herds, and Schools: A Distributed Behavioral Model”, in *Computer Graphics*, 21(4) (SIGGRAPH '87 Conference Proceedings) pages 25-34.

Schelling, T, 1971, *Dynamic models of Segregation*, *Journal of Mathematical Sociology*, 1, 143-186

Schelling, T, 1978 *Micromotives and Macrobehaviour*, W. W. Norton and Company

Sims, K., 1994, “Evolving Virtual Creatures”, *Computer Graphics* (Siggraph '94 Proceedings), July 1994, pp.15-22. available at http://web.genarts.com/karl/papers/siggraph94.pdf last accessed 21 August 2006

Snibbe, S, 1999 “Boundary Functions” http://www.youtube.com/watch?v=1p96bTARFKc Phaeno Wolfsburg Last accessed 13 March 2009

van Gelder, T.J., 1990, “Why Distributed Representation is Inherently Non-Symbolic”, in G. Dorffner (ed.) *Konnektionismus in Artificial Intelligence und Kognitionsforschung*. Berlin: Springer-Verlag, pp; 58-66

van Gelder, T.J., 1999, “Distributed versus local representation”. In R. Wilson & F. Keil ed., *The MIT Encyclopedia of Cognitive Sciences*. Cambridge MA: MIT Press, 236-8. pdf

Varela, F., H. Maturana, and R. Uribe, 1974, “FOCAL REFERENCE: Autopoiesis: The organization of living systems, its characterization and a model”, *BioSystems*, Vol. 5 (1974), pp. 187-196

Varela, F., Thompson E. & Rosch E., 1992, *the Embodied Mind*, MIT Press, Ca. Mass.

Whitaker, R., 1995a, *Autopoietic Theory and Social Systems: Theory and Practice* available at http://www.acm.org/sigs/sigois/auto/AT&Soc.html last accessed 18 August 2006

Whitaker, R., 1995b, *Self-Organization, Autopoiesis and Enterprises* available at http://www.acm.org/sigs/sigois/auto/Main.html last accessed 11 August 2006

Whitaker, R. 2001, *Autopoiesis Checklist: Observer Web Focus File* http://www.enolagaia.com/AT.html last accessed 17 August 2006

Wilensky, U., 1999, *NetLOGO* (Version 4.1.3. year 2011), http://ccl.northwestern.edu/netlogo/ last accessed 8 August 2010

Winograd, T., Flores, F., 1986, *Understanding Computers and Cognition: a New Foundation for Design*, Addison-Wesley Publishing Company, Reading, MA.

Wolfram, S., 2002, *A New Kind of Science,* Wolfram Media Inc., Illinois

Appendix 1 Sample Results

Graphic Set 1. Total Segregation with 30% occupiers

There are 10 different types of agents, friends with the same type.

1. Seed number; i.e. the number of which simulation initialize = 555,

2. Snap at 1000 steps

3. Chances of occupier-visitor generation 30% which under assumption that 70% of space would be movement space,

4. Agent types are 10,

5. Social Preference matrix applied

1,0,0,0,0,0,0,0,0,0

0,1,0,0,0,0,0,0,0,0

0,0,1,0,0,0,0,0,0,0

0,0,0,1,0,0,0,0,0,0

0,0,0,0,1,0,0,0,0,0

0,0,0,0,0,1,0,0,0,0

0,0,0,0,0,0,1,0,0,0

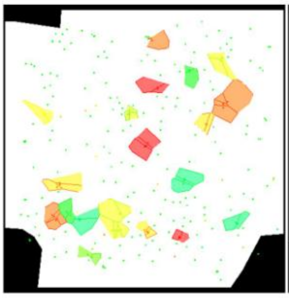
0,0,0,0,0,0,0,1,0,0

0,0,0,0,0,0,0,0,1,0

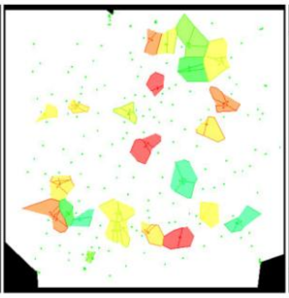
0,0,0,0,0,0,0,0,0,1

6. Size of population at 250 agents.

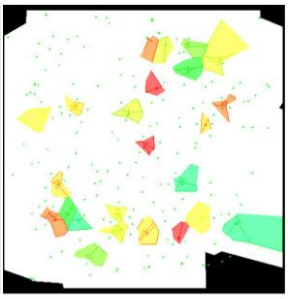
7. Additional output how many agents that have settled in.



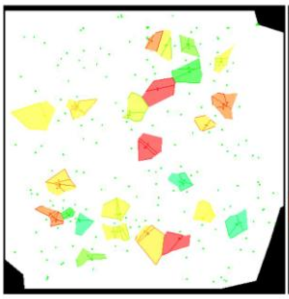
03_10s 1000.BMP



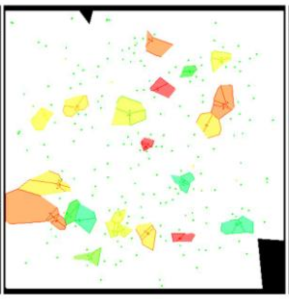
03_10s 10000.BMP



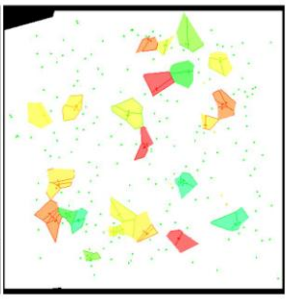
03_10s 11000.BMP



03_10s 12000.BMP

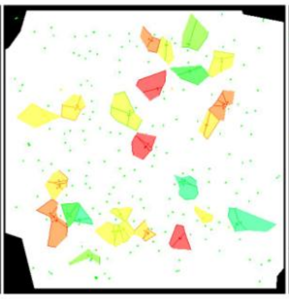


03_10s 2000.BMP

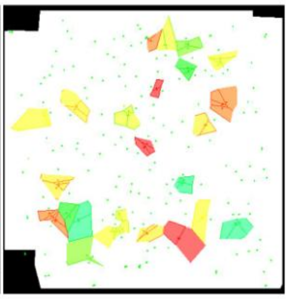


03_10s 3000.BMP

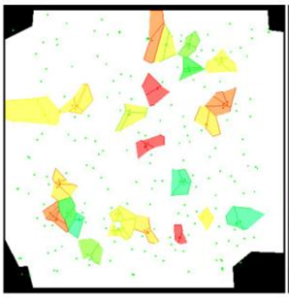
03_10s 4000.BMP



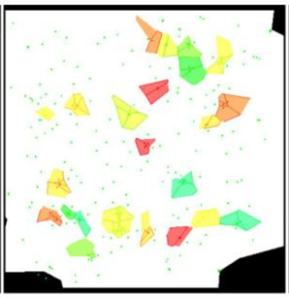
03_10s 5000.BMP



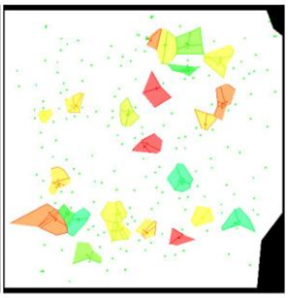
03_10s 6000.BMP



03_10s 7000.BMP



03_10s 8000.BMP



03_10s 9000.BMP

Graphic Set 2. Segregation according to type with 30% occupiers

There are 10 different types; friends with the same type and one type friendly to all other types.

1. Seed number; i.e. the number of which simulation initialize = 555,

2. Snap at 1000 steps

3. Chances of occupier-visitor generation 30% which under assumption that 70% of space would be movement space,

4. Agent types are 10,

5. Social Preference matrix applied

1,1,1,1,1,1,1,1,1,1

1,1,0,0,0,0,0,0,0,0

1,0,1,0,0,0,0,0,0,0

1,0,0,1,0,0,0,0,0,0

1,0,0,0,1,0,0,0,0,0

1,0,0,0,0,1,0,0,0,0

1,0,0,0,0,0,1,0,0,0

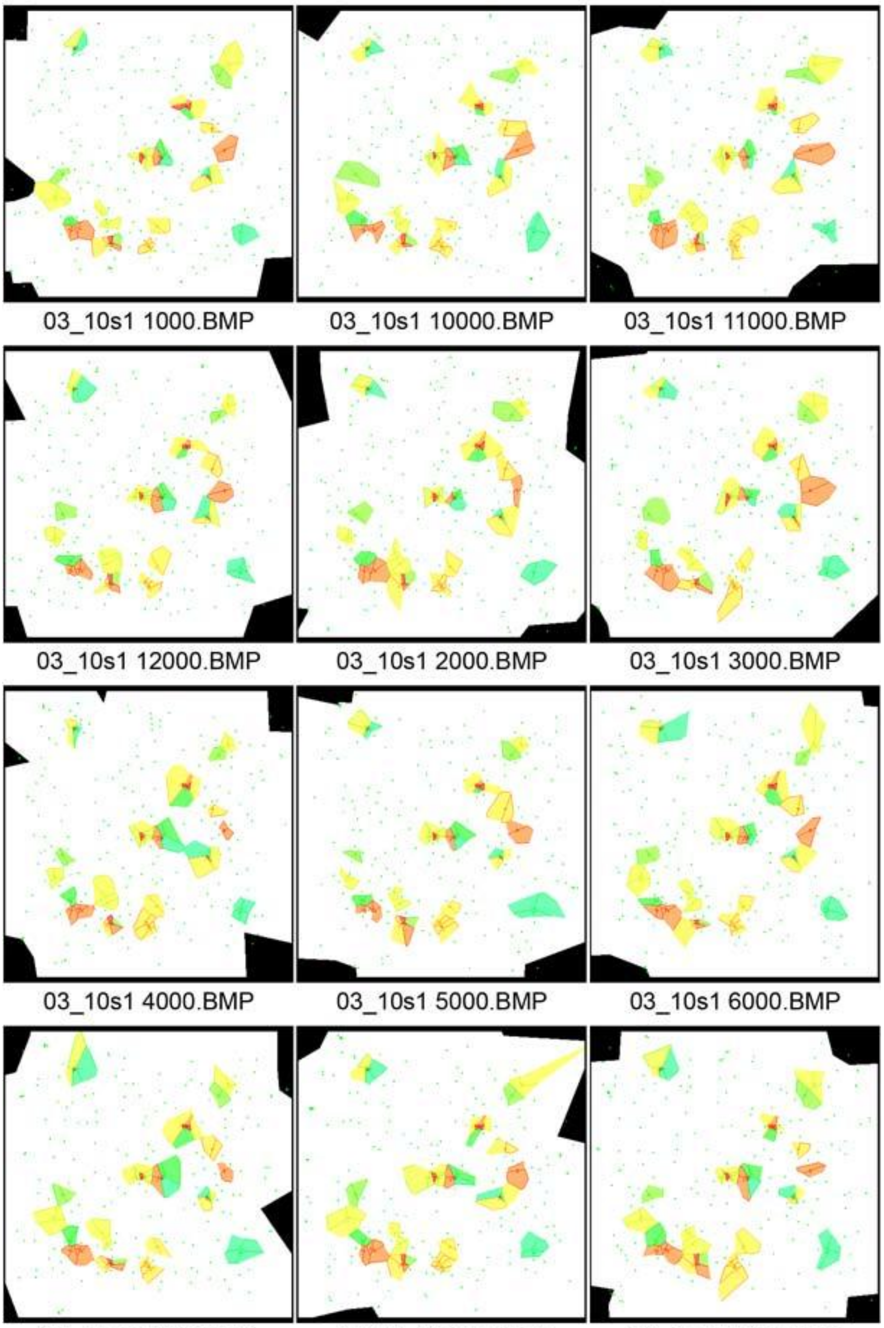
1,0,0,0,0,0,0,1,0,0

1,0,0,0,0,0,0,0,1,0

1,0,0,0,0,0,0,0,0,1

6. Size of population at 250 agents.

7. Additional output how many agents that have settled in.



03_10s1 1000.BMP | 03_10s1 10000.BMP | 03_10s1 11000.BMP

03_10s1 12000.BMP | 03_10s1 2000.BMP | 03_10s1 3000.BMP

03_10s1 4000.BMP | 03_10s1 5000.BMP | 03_10s1 6000.BMP

03_10s1 7000.BMP | 03_10s1 8000.BMP | 03_10s1 9000.BMP

Graphic Set 3. Integrated population with 30% occupiers

There are 10 different types, friends with the same type and 5 types friendly to all other 5 types.

1. Seed number; i.e. the number of which simulation initialize = 555,
2. Snap at 1000 steps
3. Chances of occupier-visitor generation 30% which under assumption that 70% of space would be movement space,
4. Agent types are 10,
5. Social Preference matrix applied

1,0,0,0,0,0,0,0,0,0

0,1,0,0,0,0,0,0,0,0

0,0,1,0,0,0,0,0,0,0

0,0,0,1,0,0,0,0,0,0

0,0,0,0,1,0,0,0,0,0

0,0,0,0,0,1,1,1,1,1

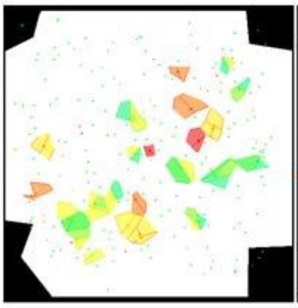
0,0,0,0,0,1,1,1,1,1

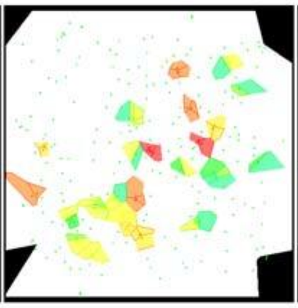
0,0,0,0,0,1,1,1,1,1

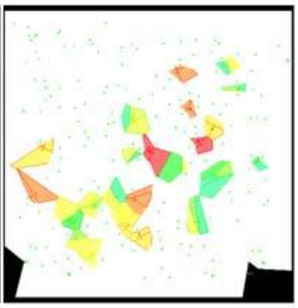
0,0,0,0,0,1,1,1,1,1

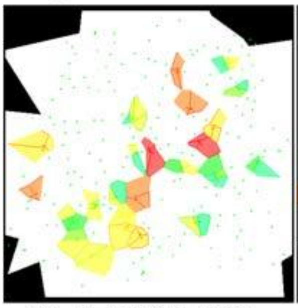
0,0,0,0,0,1,1,1,1,1

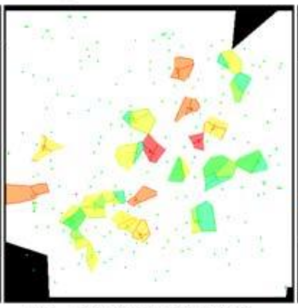
6. Size of population at 250 agents.
7. Additional output how many agents that have settled in.

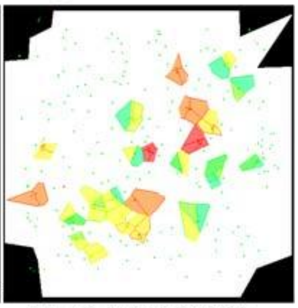

03_10s678910 1000...

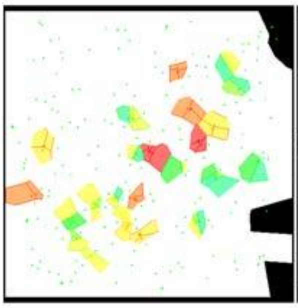

03_10s678910 1000...

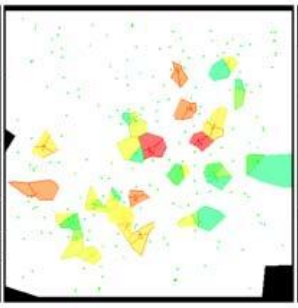

03_10s678910 1100...

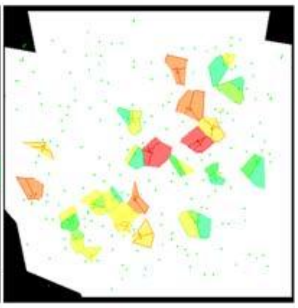

03_10s678910 1200...

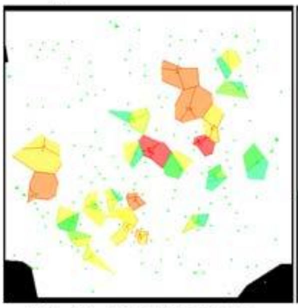

03_10s678910 2000...

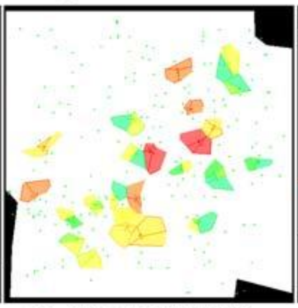

03_10s678910 3000...

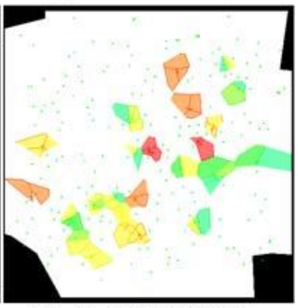

03_10s678910 4000...


03_10s678910 5000...


03_10s678910 6000...


03_10s678910 7000...


03_10s678910 8000...


03_10s678910 9000...

Graphic Set 4. Total segregation with 50% occupiers

There are 10 different types of agents, friends with the same type.

1. Seed number; i.e. the number of which simulation initialize = 555,

2. Snap at 1000 steps

3. Chances of occupier-visitor generation 50% which under assumption that 50% of space would be movement space,

4. Agent types are 10,

5. Social Preference matrix applied

1,0,0,0,0,0,0,0,0,0

0,1,0,0,0,0,0,0,0,0

0,0,1,0,0,0,0,0,0,0

0,0,0,1,0,0,0,0,0,0

0,0,0,0,1,0,0,0,0,0

0,0,0,0,0,1,0,0,0,0

0,0,0,0,0,0,1,0,0,0

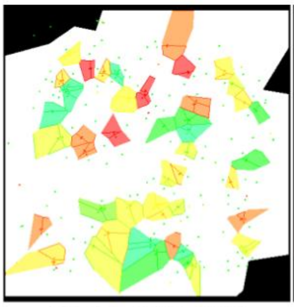
0,0,0,0,0,0,0,1,0,0

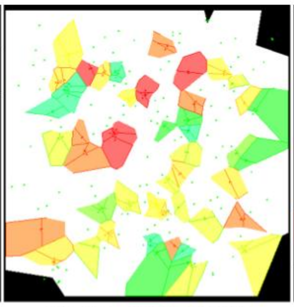
0,0,0,0,0,0,0,0,1,0

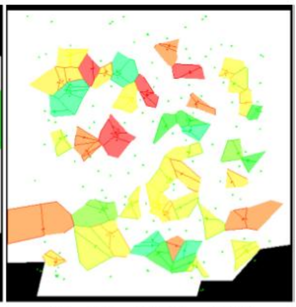
0,0,0,0,0,0,0,0,0,1

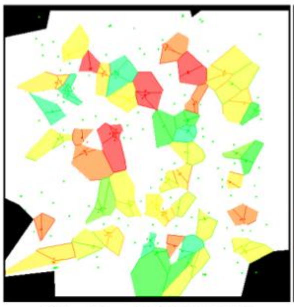
6. Size of population at 250 agents.

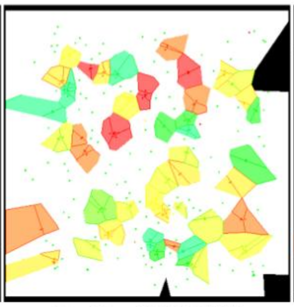
7. Additional output how many agents that have settled in.

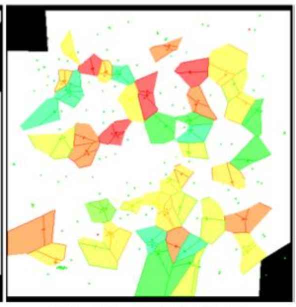

05_10s 1000.BMP

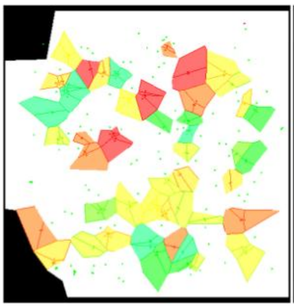

05_10s 10000.BMP

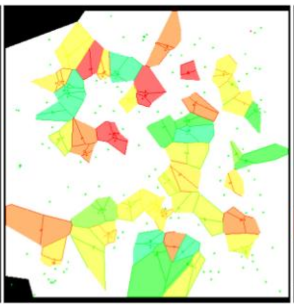

05_10s 11000.BMP

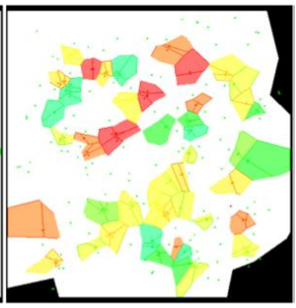

05_10s 12000.BMP

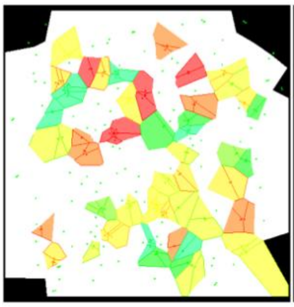

05_10s 2000.BMP

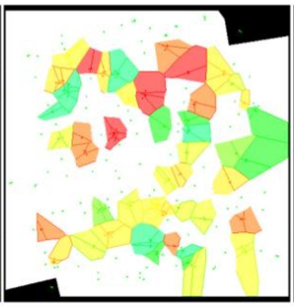

05_10s 3000.BMP

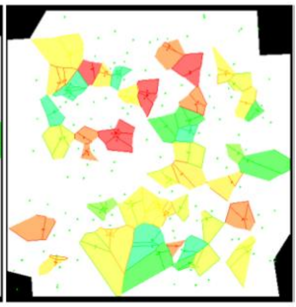

05_10s 4000.BMP


05_10s 5000.BMP


05_10s 6000.BMP


05_10s 7000.BMP


05_10s 8000.BMP


05_10s 9000.BMP

Graphic Set 5. Segregation according to type with 50% occupiers

There are 10 different types; friends with the same type and one type friendly to all other types.

1. Seed number; i.e. the number of which simulation initialize = 555,
2. Snap at 1000 steps
3. Chances of occupier-visitor generation 50% which under assumption that 50% of space would be movement space,
4. Agent types are 10,
5. Social Preference matrix applied

   1,1,1,1,1,1,1,1,1,1

   1,1,0,0,0,0,0,0,0,0

   1,0,1,0,0,0,0,0,0,0

   1,0,0,1,0,0,0,0,0,0

   1,0,0,0,1,0,0,0,0,0

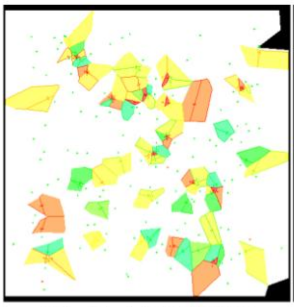
   1,0,0,0,0,1,0,0,0,0

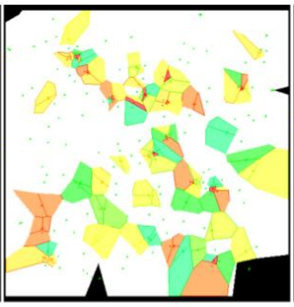
   1,0,0,0,0,0,1,0,0,0

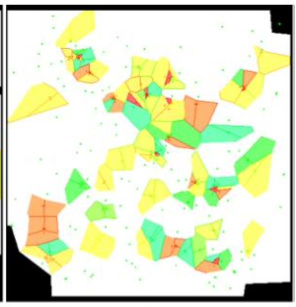
   1,0,0,0,0,0,0,1,0,0

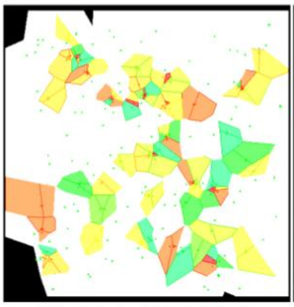
   1,0,0,0,0,0,0,0,1,0

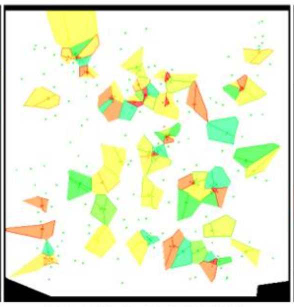
   1,0,0,0,0,0,0,0,0,1
6. Size of population at 250 agents.
7. Additional output how many agents that have settled in.

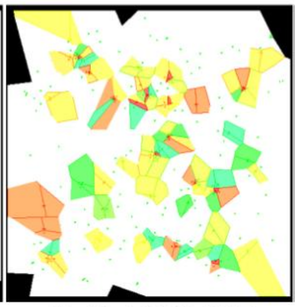

05_10s1 1000.BMP

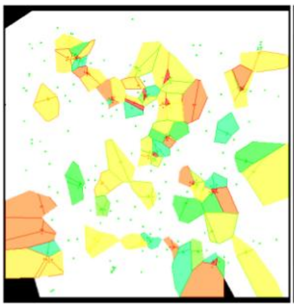

05_10s1 10000.BMP

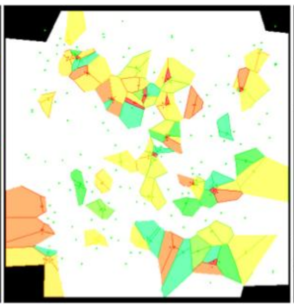

05_10s1 11000.BMP

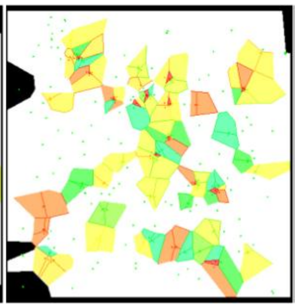

05_10s1 12000.BMP

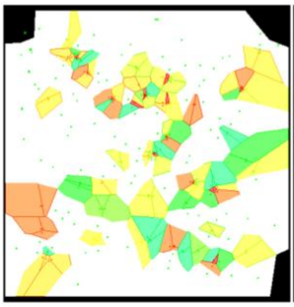

05_10s1 2000.BMP

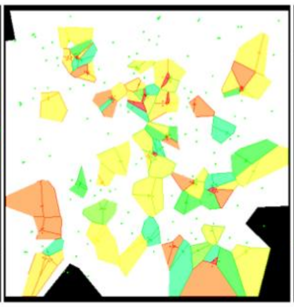

05_10s1 3000.BMP

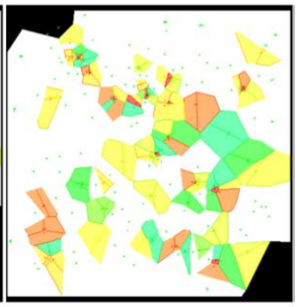

05_10s1 4000.BMP


05_10s1 5000.BMP


05_10s1 6000.BMP


05_10s1 7000.BMP


05_10s1 8000.BMP


05_10s1 9000.BMP

Graphic Set 6. Integrated population with 50% occupiers

There are 10 different types, friends with the same type and 5 types friendly to all other 5 types.

1. Seed number; i.e. the number of which simulation initialize = 555,

2. Snap at 1000 steps

3. Chances of occupier-visitor generation 50% which under assumption that 50% of space would be movement space,

4. Agent types are 10,

5. Social Preference matrix applied

1,0,0,0,0,0,0,0,0,0

0,1,0,0,0,0,0,0,0,0

0,0,1,0,0,0,0,0,0,0

0,0,0,1,0,0,0,0,0,0

0,0,0,0,1,0,0,0,0,0

0,0,0,0,0,1,1,1,1,1

0,0,0,0,0,1,1,1,1,1

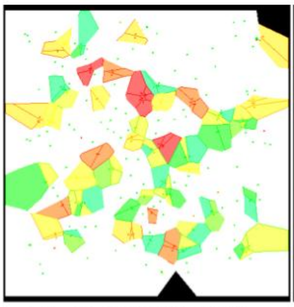
0,0,0,0,0,1,1,1,1,1

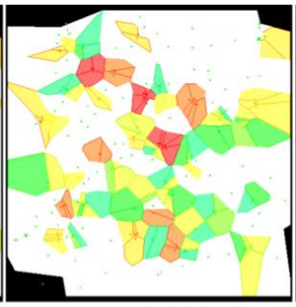
0,0,0,0,0,1,1,1,1,1

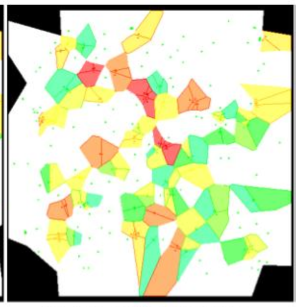
0,0,0,0,0,1,1,1,1,1

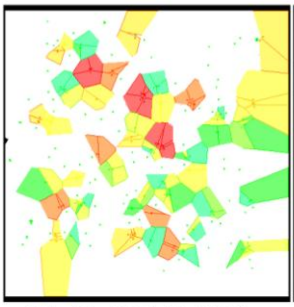
6. Size of population at 250 agents.

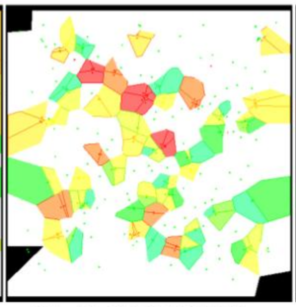
7. Additional output how many agents that have settled in.

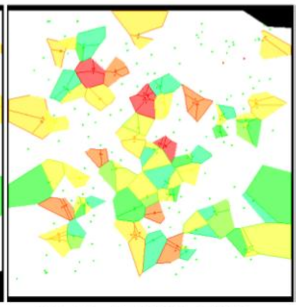

05_10s678910 1000...

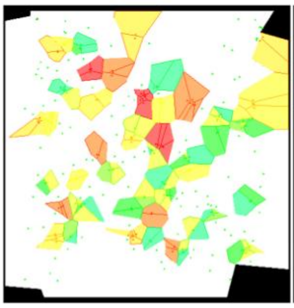

05_10s678910 1000...

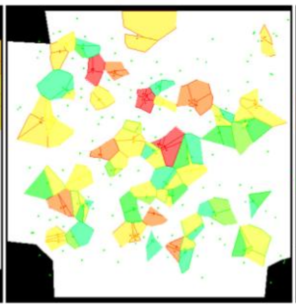

05_10s678910 1100...

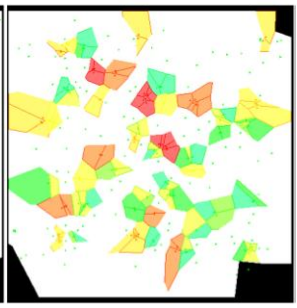

05_10s678910 1200...

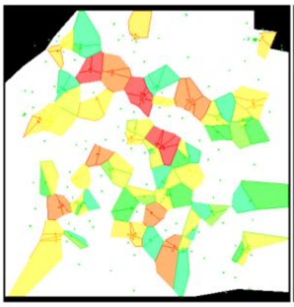

05_10s678910 2000...

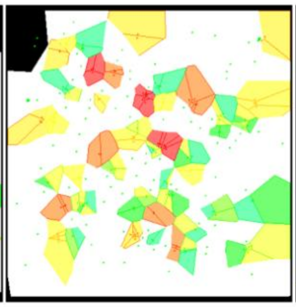

05_10s678910 3000...

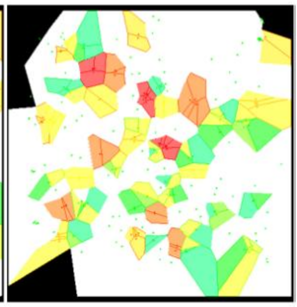

05_10s678910 4000...


05_10s678910 5000...


05_10s678910 6000...


05_10s678910 7000...


05_10s678910 8000...


05_10s678910 9000...

Graphic Set 7. Total Segregation with 80% occupiers

There are 10 different types of agents, friends with the same type.

1. Seed number; i.e. the number of which simulation initialize = 555,
2. Snap at 1000 steps
3. Chances of occupier-visitor generation 80% which under assumption that only 20% of space would be movement space,
4. Agent types are 10,
5. Social Preference 1,0,0,0,0,0,0,0,0,0

   0,1,0,0,0,0,0,0,0,0

   0,0,1,0,0,0,0,0,0,0

   0,0,0,1,0,0,0,0,0,0

   0,0,0,0,1,0,0,0,0,0

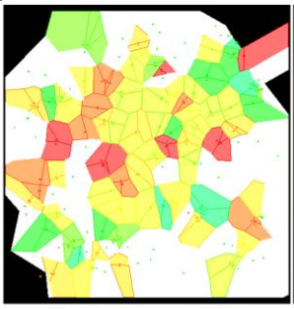
   0,0,0,0,0,1,0,0,0,0

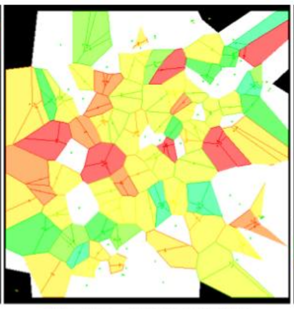
   0,0,0,0,0,0,1,0,0,0

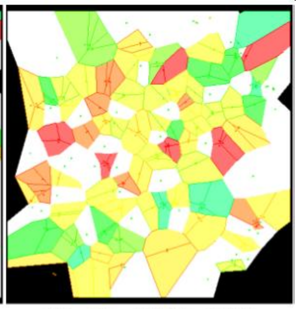
   0,0,0,0,0,0,0,1,0,0

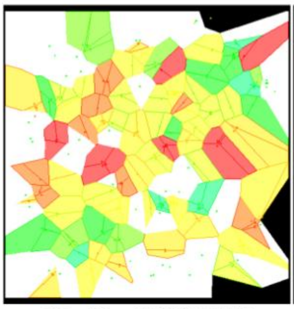
   0,0,0,0,0,0,0,0,1,0

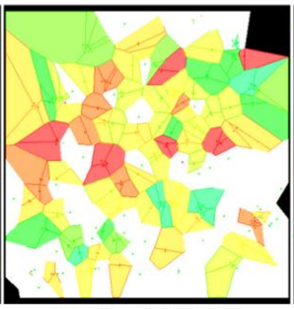
   0,0,0,0,0,0,0,0,0,1
6. Size of population at 250 agents.
7. Additional output how many agents that have settled in.

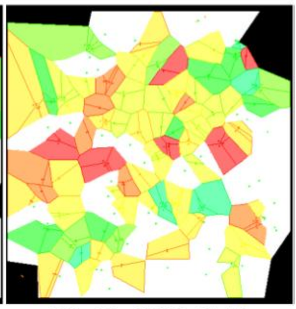

08_10s 1000.BMP

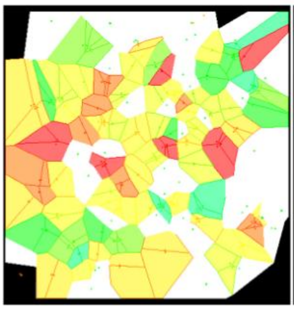

08_10s 10000.BMP

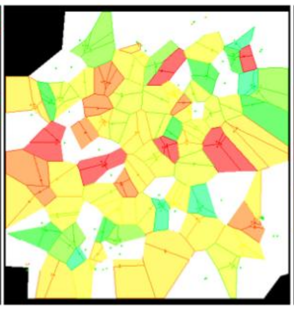

08_10s 11000.BMP

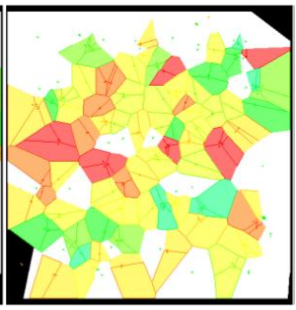

08_10s 12000.BMP

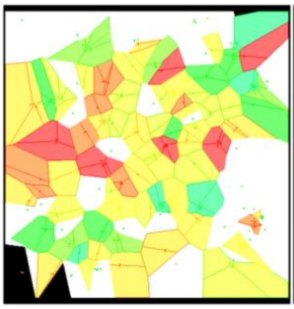

08_10s 2000.BMP

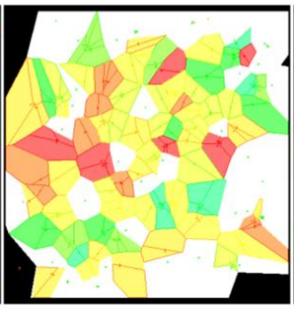

08_10s 3000.BMP

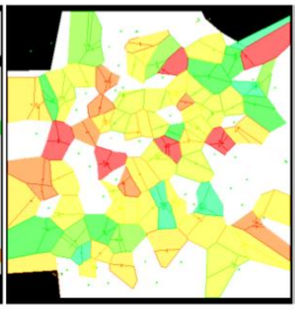

08_10s 4000.BMP


08_10s 5000.BMP


08_10s 6000.BMP


08_10s 7000.BMP


08_10s 8000.BMP


08_10s 9000.BMP

Graphic Set 8. Segregation according to type with 80% occupiers

There are 10 different types; friends with the same type and one type friendly to all other types.

1. Seed number; i.e. the number of which simulation initialize = 555,
2. Snap at 1000 steps
3. Chances of occupier-visitor generation 80% which under assumption that only 20% of space would be movement space,
4. Agent types are 10,
5. Social Preference matrix applied

1,1,1,1,1,1,1,1,1,1

1,1,0,0,0,0,0,0,0,0

1,0,1,0,0,0,0,0,0,0

1,0,0,1,0,0,0,0,0,0

1,0,0,0,1,0,0,0,0,0

1,0,0,0,0,1,0,0,0,0

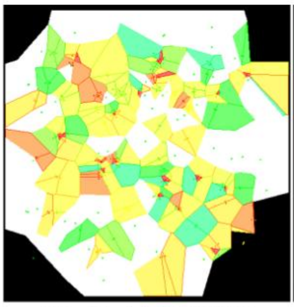
1,0,0,0,0,0,1,0,0,0

1,0,0,0,0,0,0,1,0,0

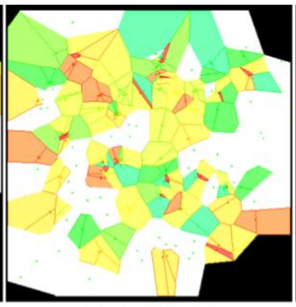
1,0,0,0,0,0,0,0,1,0

1,0,0,0,0,0,0,0,0,1

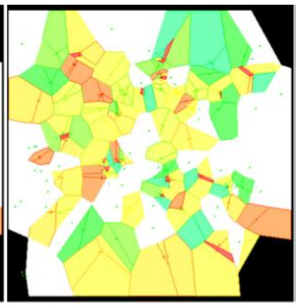
6. Size of population at 250 agents.
7. Additional output how many agents that have settled in.



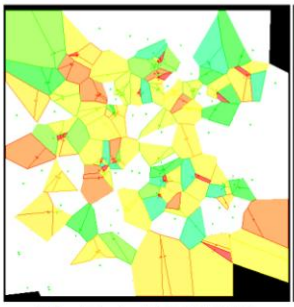
08_10s1 1000.BMP



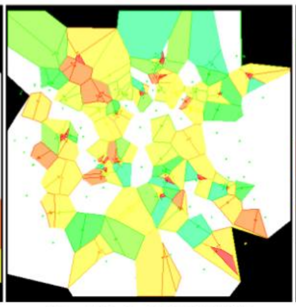
08_10s1 10000.BMP



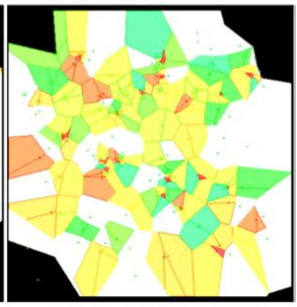
08_10s1 11000.BMP



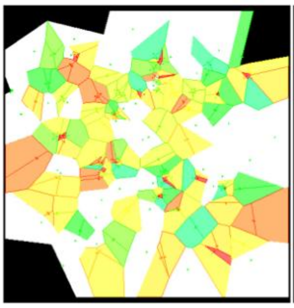
08_10s1 12000.BMP



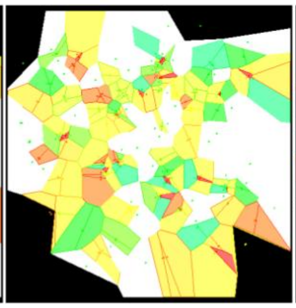
08_10s1 2000.BMP



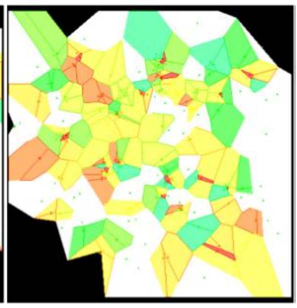
08_10s1 3000.BMP



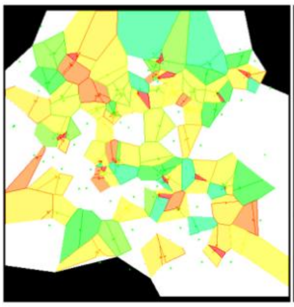
08_10s1 4000.BMP



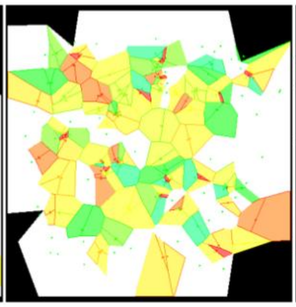
08_10s1 5000.BMP



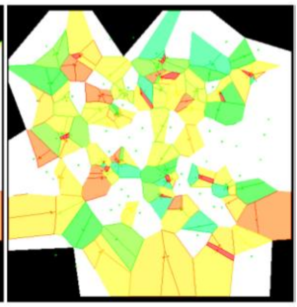
08_10s1 6000.BMP



08_10s1 7000.BMP



08_10s1 8000.BMP



08_10s1 9000.BMP

Graphic Set 9. Integrated population with 80% occupiers

There are 10 different types, friends with the same type and 5 types friendly to all other 5 types.

1. Seed number; i.e. the number of which simulation initialize = 555,

2. Snap at 1000 steps

3. Chances of occupier-visitor generation 80% which under assumption that only 20% of space would be movement space,

4. Agent types are 10,

5. Social Preference matrix applied

1,0,0,0,0,0,0,0,0,0

0,1,0,0,0,0,0,0,0,0

0,0,1,0,0,0,0,0,0,0

0,0,0,1,0,0,0,0,0,0

0,0,0,0,1,0,0,0,0,0

0,0,0,0,0,1,1,1,1,1

0,0,0,0,0,1,1,1,1,1

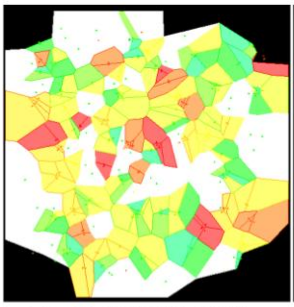
0,0,0,0,0,1,1,1,1,1

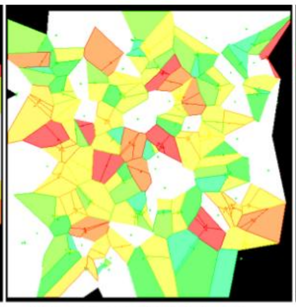
0,0,0,0,0,1,1,1,1,1

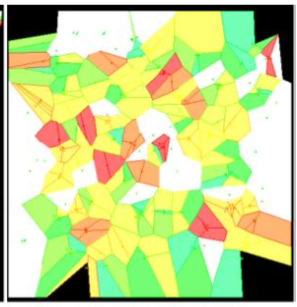
0,0,0,0,0,1,1,1,1,1

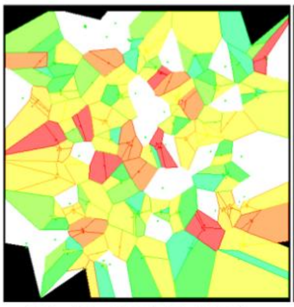
6. Size of population at 250 agents.

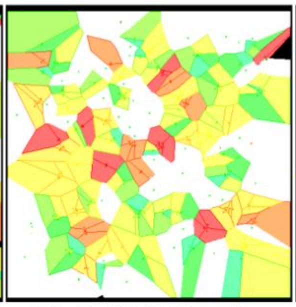
7. Additional output how many agents that have settled in.

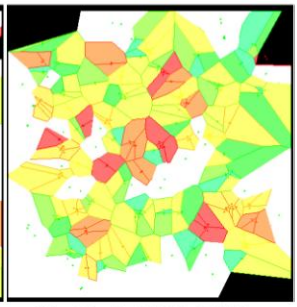

08_10s678910 1000...

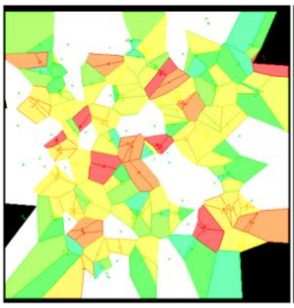

08_10s678910 1000...

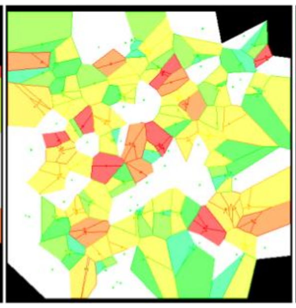

08_10s678910 1100...

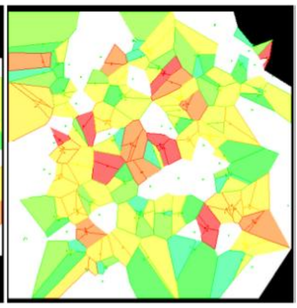

08_10s678910 1200...

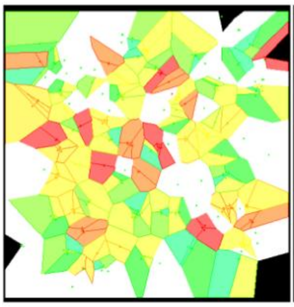

08_10s678910 2000...

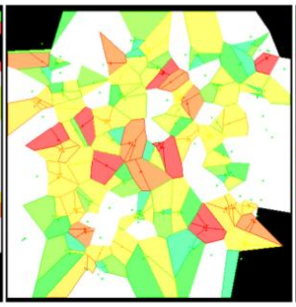

08_10s678910 3000...

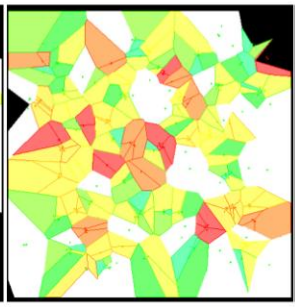

08_10s678910 4000...


08_10s678910 5000...

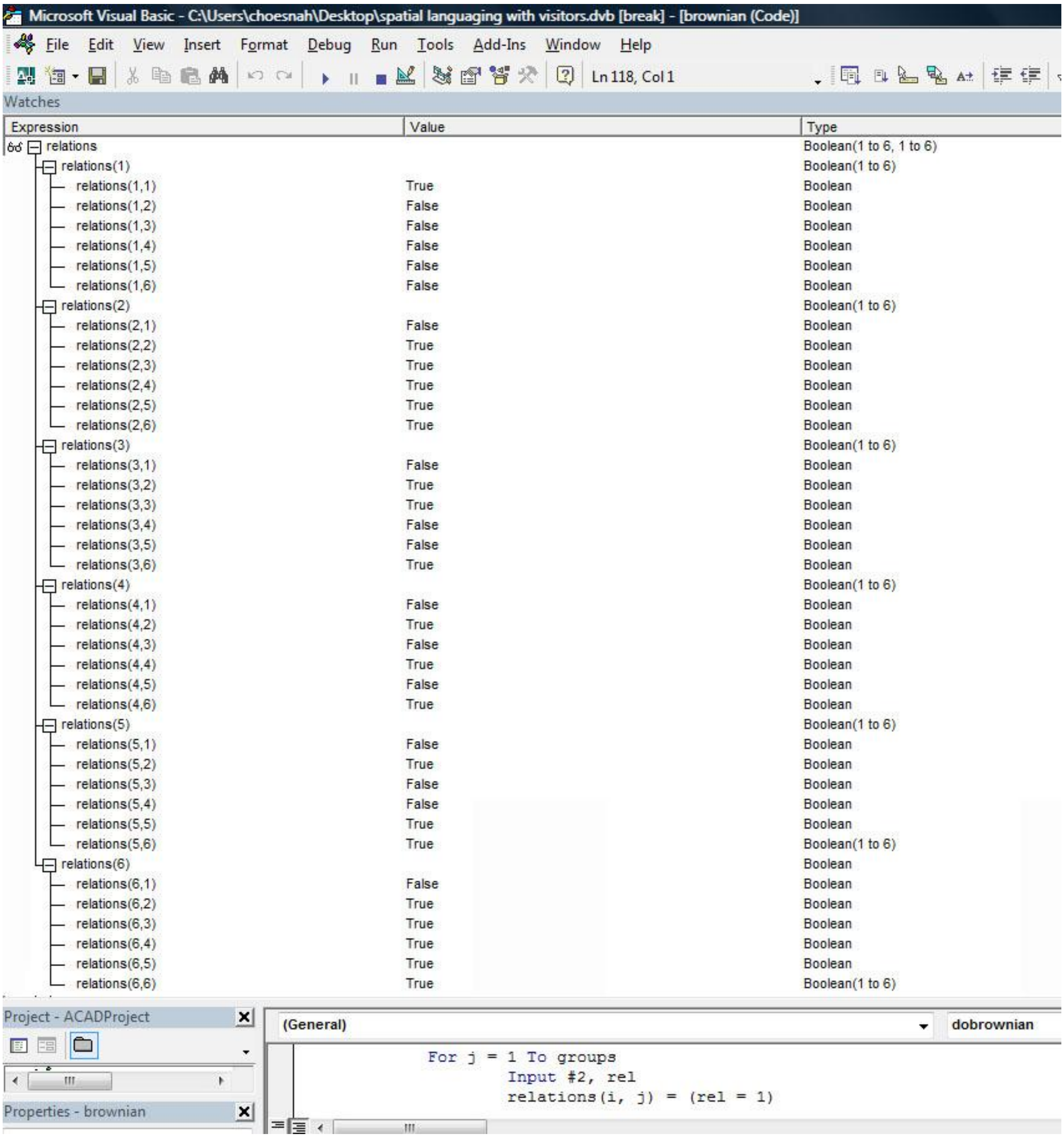

08_10s678910 6000...


08_10s678910 7000...


08_10s678910 8000...


08_10s678910 9000...

Appendix 2 Sample Debugging: Social Preference Matrix



The Social Preference Matrix for the sample above is as below.

1,0,0,0,0,0

0,1,1,1,1,1

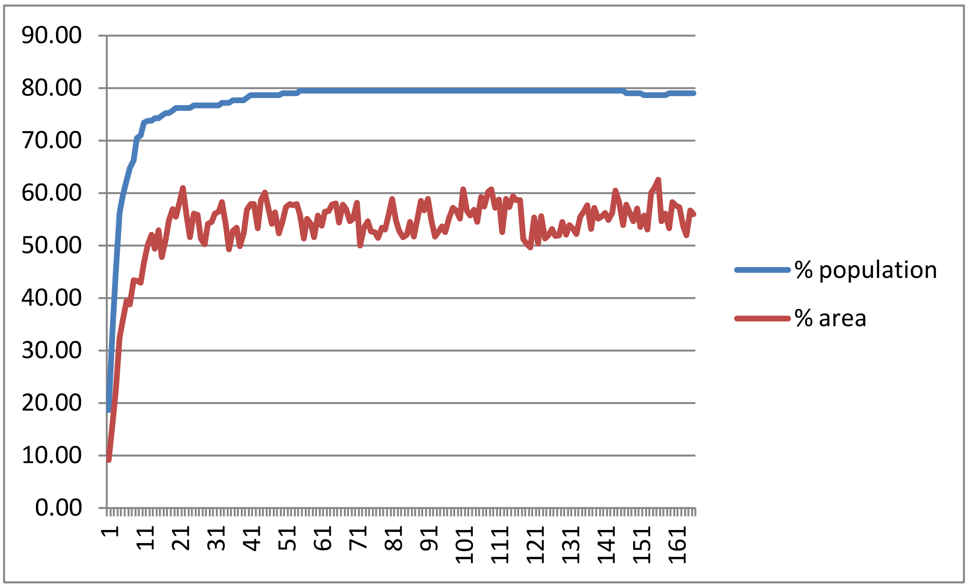
0,1,1,0,0,1

0,1,0,1,0,1

0,1,0,0,1,1

0,1,1,1,1,1

Appendix 3 Sample Numerical Outputs



INPUTS

Populations 210 agents

Universe size 400x400 units

Schelling total segregation matrix with 20% visitors

OUTPUTS

ORIGINAL DATA OUT			PRESENTATION DATA			
Counter	Cells	Total area	Image Number	time line	% population	% area
60	39	14574.54	60	1	18.57	9.11
120	71	25778.33	120	2	33.81	16.11
180	94	35519.00	180	3	44.76	22.20
240	118	51761.67	240	4	56.19	32.35
300	125	57049.38	300	5	59.52	35.66
360	131	63102.09	360	6	62.38	39.44
420	136	61934.58	420	7	64.76	38.71
480	139	69477.81	480	8	66.19	43.42
540	148	69292.88	540	9	70.48	43.31
600	149	68579.15	600	10	70.95	42.86
660	154	74602.57	660	11	73.33	46.63
720	155	80206.85	720	12	73.81	50.13
780	155	83259.79	780	13	73.81	52.04
840	156	79067.33	840	14	74.29	49.42
900	156	84699.35	900	15	74.29	52.94
960	157	76401.08	960	16	74.76	47.75
1020	158	81634.62	1020	17	75.24	51.02
1080	158	87503.61	1080	18	75.24	54.69
1140	159	90990.41	1140	19	75.71	56.87
1200	160	88776.08	1200	20	76.19	55.49
1260	160	93454.35	1260	21	76.19	58.41
1320	160	97443.92	1320	22	76.19	60.90
1380	160	88706.67	1380	23	76.19	55.44
1440	160	82555.87	1440	24	76.19	51.60
1500	161	89630.25	1500	25	76.67	56.02
1560	161	89343.03	1560	26	76.67	55.84
1620	161	82037.65	1620	27	76.67	51.27
1680	161	80304.71	1680	28	76.67	50.19
1740	161	86584.95	1740	29	76.67	54.12
1800	161	87190.27	1800	30	76.67	54.49
1860	161	89737.71	1860	31	76.67	56.09
1920	161	90286.46	1920	32	76.67	56.43
1980	162	93233.59	1980	33	77.14	58.27
2040	162	86310.04	2040	34	77.14	53.94
2100	162	78751.19	2100	35	77.14	49.22
2160	163	84528.95	2160	36	77.62	52.83
2220	163	85494.33	2220	37	77.62	53.43
2280	163	79675.41	2280	38	77.62	49.80
2340	163	83658.09	2340	39	77.62	52.29
2400	164	90863.43	2400	40	78.10	56.79
2460	165	92574.10	2460	41	78.57	57.86
2520	165	92614.09	2520	42	78.57	57.88
2580	165	85127.14	2580	43	78.57	53.20
2640	165	93845.83	2640	44	78.57	58.65
2700	165	96098.81	2700	45	78.57	60.06

2760	165	91800.38	2760	46	78.57	57.38
2820	165	86502.05	2820	47	78.57	54.06
2880	165	90060.80	2880	48	78.57	56.29
2940	165	83563.28	2940	49	78.57	52.23
3000	166	87476.31	3000	50	79.05	54.67
3060	166	91701.59	3060	51	79.05	57.31
3120	166	92542.66	3120	52	79.05	57.84
3180	166	92151.80	3180	53	79.05	57.59
3240	166	92721.17	3240	54	79.05	57.95
3300	167	89355.92	3300	55	79.52	55.85
3360	167	82157.69	3360	56	79.52	51.35
3420	167	88056.18	3420	57	79.52	55.04
3480	167	86505.55	3480	58	79.52	54.07
3540	167	82509.07	3540	59	79.52	51.57
3600	167	89206.91	3600	60	79.52	55.75
3660	167	85972.63	3660	61	79.52	53.73
3720	167	90224.32	3720	62	79.52	56.39
3780	167	90470.58	3780	63	79.52	56.54
3840	167	92427.12	3840	64	79.52	57.77
3900	167	92811.74	3900	65	79.52	58.01
3960	167	86972.14	3960	66	79.52	54.36
4020	167	92406.46	4020	67	79.52	57.75
4080	167	91088.95	4080	68	79.52	56.93
4140	167	87421.54	4140	69	79.52	54.64
4200	167	88217.31	4200	70	79.52	55.14
4260	167	93036.15	4260	71	79.52	58.15
4320	167	79916.13	4320	72	79.52	49.95
4380	167	85358.09	4380	73	79.52	53.35
4440	167	87316.54	4440	74	79.52	54.57
4500	167	84166.83	4500	75	79.52	52.60
4560	167	84028.40	4560	76	79.52	52.52
4620	167	82270.77	4620	77	79.52	51.42
4680	167	85493.67	4680	78	79.52	53.43
4740	167	84739.61	4740	79	79.52	52.96
4800	167	89772.28	4800	80	79.52	56.11
4860	167	94191.51	4860	81	79.52	58.87
4920	167	87576.95	4920	82	79.52	54.74
4980	167	84365.35	4980	83	79.52	52.73
5040	167	82389.28	5040	84	79.52	51.49
5100	167	83245.06	5100	85	79.52	52.03
5160	167	87237.87	5160	86	79.52	54.52
5220	167	82643.31	5220	87	79.52	51.65
5280	167	87621.99	5280	88	79.52	54.76
5340	167	93525.72	5340	89	79.52	58.45
5400	167	90622.11	5400	90	79.52	56.64
5460	167	94151.86	5460	91	79.52	58.84
5520	167	88030.47	5520	92	79.52	55.02
5580	167	82595.33	5580	93	79.52	51.62
5640	167	84007.15	5640	94	79.52	52.50
5700	167	85814.00	5700	95	79.52	53.63
5760	167	84014.47	5760	96	79.52	52.51
5820	167	88389.56	5820	97	79.52	55.24
5880	167	91552.30	5880	98	79.52	57.22
5940	167	90616.26	5940	99	79.52	56.64
6000	167	88132.75	6000	100	79.52	55.08
6060	167	97073.08	6060	101	79.52	60.67
6120	167	90482.75	6120	102	79.52	56.55
6180	167	89185.97	6180	103	79.52	55.74
6240	167	90808.94	6240	104	79.52	56.76
6300	167	87175.06	6300	105	79.52	54.48
6360	167	94795.51	6360	106	79.52	59.25
6420	167	91894.78	6420	107	79.52	57.43
6480	167	96290.18	6480	108	79.52	60.18
6540	167	97113.21	6540	109	79.52	60.70
6600	167	91511.67	6600	110	79.52	57.19
6660	167	93972.27	6660	111	79.52	58.73
6720	167	83960.90	6720	112	79.52	52.48
6780	167	94251.83	6780	113	79.52	58.91
6840	167	91606.15	6840	114	79.52	57.25
6900	167	95056.65	6900	115	79.52	59.41
6960	167	93806.55	6960	116	79.52	58.63
7020	167	93794.16	7020	117	79.52	58.62
7080	167	81933.29	7080	118	79.52	51.21
7140	167	80350.77	7140	119	79.52	50.22
7200	167	79381.41	7200	120	79.52	49.61
7260	167	88469.91	7260	121	79.52	55.29
7320	167	80574.62	7320	122	79.52	50.36
7380	167	88969.37	7380	123	79.52	55.61
7440	167	82051.74	7440	124	79.52	51.28
7500	167	83071.61	7500	125	79.52	51.92
7560	167	85001.19	7560	126	79.52	53.13
7620	167	82945.48	7620	127	79.52	51.84
7680	167	83095.41	7680	128	79.52	51.93
7740	167	87264.97	7740	129	79.52	54.54
7800	167	83192.41	7800	130	79.52	52.00
7860	167	86213.34	7860	131	79.52	53.88
7920	167	85096.78	7920	132	79.52	53.19
7980	167	83430.35	7980	133	79.52	52.14
8040	167	88796.16	8040	134	79.52	55.50

8100	167	90549.14	8100	135	79.52	56.59
8160	167	92224.49	8160	136	79.52	57.64
8220	167	84993.56	8220	137	79.52	53.12
8280	167	91468.37	8280	138	79.52	57.17
8340	167	88132.64	8340	139	79.52	55.08
8400	167	88795.90	8400	140	79.52	55.50
8460	167	89914.22	8460	141	79.52	56.20
8520	167	87786.70	8520	142	79.52	54.87
8580	167	89864.82	8580	143	79.52	56.17
8640	167	96655.03	8640	144	79.52	60.41
8700	167	93086.99	8700	145	79.52	58.18
8760	167	86186.71	8760	146	79.52	53.87
8820	166	92391.91	8820	147	79.05	57.74
8880	166	89648.77	8880	148	79.05	56.03
8940	166	87341.41	8940	149	79.05	54.59
9000	166	91344.91	9000	150	79.05	57.09
9060	166	85620.48	9060	151	79.05	53.51
9120	165	89101.45	9120	152	78.57	55.69
9180	165	84911.91	9180	153	78.57	53.07
9240	165	96239.50	9240	154	78.57	60.15
9300	165	97591.43	9300	155	78.57	60.99
9360	165	100000.04	9360	156	78.57	62.50
9420	165	87452.41	9420	157	78.57	54.66
9480	165	89667.15	9480	158	78.57	56.04
9540	166	85188.29	9540	159	79.05	53.24
9600	166	93193.57	9600	160	79.05	58.25
9660	166	92056.85	9660	161	79.05	57.54
9720	166	91675.43	9720	162	79.05	57.30
9780	166	85900.66	9780	163	79.05	53.69
9840	166	83121.96	9840	164	79.05	51.95
9900	166	90718.33	9900	165	79.05	56.70
9960	166	89459.74	9960	166	79.05	55.91

Appendix 4 The Code

The program is spatial_languaging.dvb and it is compiled in AutoCAD Visual Basic programming.

Spatial_languaging.dvb contains 4 modules as follows.

Module boundarystuff

```
' version based on may 06 for presentation no changes to overall system but
' to get the agents to stop running away had to do a kludge and just trap and chuck back inside
' main breakthrough was to realise that the circle representation became uncoupled from the actual agent
' once i redefined the circle center to syncronise with the agent pos it look like it was
' rewrote push as carefullypush
' teatime is now really straightforward
' 21st jan 2007




Public bound As AcadRegion 'global definition for the polygon on the screen
Public polys() As AcadLWPolyline
Public boundarypoly As AcadLWPolyline
Const fuzz = 0.1  'make fuzz bigger to ignore more points
Const vcolour = acWhite
Public ticks As Integer

Public recs As Integer


Sub main()
   Dim name As String
   pts = 0
   getpolynodes 0


   'recs = InputBox("gens", "howmany", "2", 5000, 5000)
   'ZoomExtents
   dobrownian 0
   'counterform.Show

 End Sub

 Sub recurse(token As Integer)
   Dim filename As String
   Dim topleft As mypoint, bottomright As mypoint
   Dim message, title, default As String, counter As Integer
   Dim j As Integer, k As Integer, i As Integer, seed As Integer
   Dim newpoint As mypoint, status As Integer, newpointscount As Integer
   Dim r As Integer
   Dim newpoints() As mypoint, this As mypoint

   Rnd (0)
   Randomize
   counter = 0

   Do
        If counter > 0 Then

           eraseregions (0)
        End If
        'ThisDrawing.Regen acActiveViewport

           voronoi (i)


          ' For i = 1 To pts
```

```
        ' drawpoint originalpoints(i), acRed, 2
        ' Next i

         For i = 1 To pts
            drawpoly cells(i)
         Next i

         erasepolylines (0)
         'ZoomExtents
         'colourin cells
        ' ThisDrawing.Regen acActiveViewport
         MsgBox ("ok")


         ca (i)
      counter = counter + 1
      filename = caseries + Str(counter) + ".dwg"

      ThisDrawing.SaveAs filename



  Loop Until counter > recs
  'Unload counterform


  End Sub




Sub ca(d As Integer)

  Dim i As Integer, j As Integer, nbs As Integer, totaldist As Double
  Dim mytype As Integer, typecounter As Integer
  Dim count As Integer
  Dim jumps As Integer


  ' neues versie die automatien
  ' each cell has a state 1 or 0
  ' if i am an x then unless i have at least 1 y need agent to jump
  ' if i am a y then unless i have at least 1 y need jump
  jumps = 0
  For i = 1 To pts

  count = 0
  mytype = cells(i).spacetype

   For j = 1 To neighbour(i).tot 'cell i's nbs

      count = count + cells(neighbour(i).item(j)).spacetype
    Next j

     If count = neighbour(i).tot Then

     cells(i).jump = True
     jumps = jumps + 1
    End If


  Next i

End Sub




Function blur(p As mypoint) As mypoint
  p.x = p.x + Rnd * fuzz
```

```
   p.y = p.y + Rnd * fuzz
   blur = p
End Function

Function nottoobig(a As mypoint) As Boolean
If a.x > 100000 And a.y > 100000 Then
      nottoobig = False
   Else
      nottoobig = True
End If
End Function
Function unique(this As mypoint, thepoints() As mypoint, tot As Integer) As Boolean
Dim i As Integer
unique = True
If tot > 0 Then



      'make this a var so i can see why it always fails
         For i = 1 To tot
            If (Abs(this.x - thepoints(i).x) < fuzz) Or (Abs(this.y - thepoints(i).y) < fuzz) Then
            ' coordinates less than fuzz apart treated as the same
               unique = False
               Exit For
            End If
         Next i

 End If

End Function




   Sub erasepolylines(d As Integer)
   Dim gpCode(0) As Integer
   Dim dataValue(0) As Variant
   Dim groupCode As Variant, dataCode As Variant
   Dim allpolys As AcadSelectionSet
   Dim apoly As AcadLWPolyline

   'erase all polylines apart from those on layer boundary

   Set allpolys = ThisDrawing.SelectionSets.add("stuff")
   Mode = acSelectionSetAll
   gpCode(0) = 0
   dataValue(0) = "LWPOLYLINE"
   groupCode = gpCode
   dataCode = dataValue
   allpolys.Select Mode, , , groupCode, dataCode


   If allpolys.count > 0 Then
         For Each apoly In allpolys
         If apoly.Layer <> "boundary" Then apoly.Delete
         Next

         End If


  allpolys.Delete





   End Sub
    Sub eraseregions(d As Integer)
   Dim gpCode(0) As Integer
   Dim dataValue(0) As Variant
   Dim groupCode As Variant, dataCode As Variant
   Dim allregions As AcadSelectionSet
```

```
   Dim aregion As AcadRegion
   Set allregions = ThisDrawing.SelectionSets.add("stuff")
   Mode = acSelectionSetAll
   gpCode(0) = 0
   dataValue(0) = "REGION"
   groupCode = gpCode
   dataCode = dataValue
   allregions.Select Mode, , , groupCode, dataCode

   If allregions.count > 0 Then
         For Each aregion In allregions
         If aregion.Layer <> "boundary" Then aregion.Delete
         Next

         End If

  allregions.Delete



End Sub

Sub getpolynodes(dummy As Integer)

Dim gpCode(0) As Integer
Dim dataValue(0) As Variant
Dim groupCode As Variant, dataCode As Variant
Dim apoint(2) As Double
Dim circ As AcadCircle, count As Integer
Dim ssetobj As AcadSelectionSet
Dim ss As Variant

pts = 1: count = 1
checkforboundary 0

If ThisDrawing.SelectionSets.count > 0 Then

ThisDrawing.SelectionSets.item("stuff").Delete

End If

'-------------------------------------------------------look for site boundary in christians darwing on layer 05 boundary-------------------------
' ------------ and extract points for voronoi -------------------------------------
   Set ssetobj = ThisDrawing.SelectionSets.add("stuff")
   Mode = acSelectionSetAll
   gpCode(0) = 8
   dataValue(0) = "boundary"
   groupCode = gpCode
   dataCode = dataValue
   ssetobj.Select Mode, , , groupCode, dataCode
    bb% = ssetobj.count

   If ssetobj.count > 0 Then  ' found some polys
      If ssetobj.count > 1 Then
      MsgBox ("too many boundaries")
      Exit Sub
      End If

            Set boundarypoly = ssetobj.item(0)         'make boundarypoly global for recreation purposes
            boundarypoly.Layer = "boundary"
            'extractpoints boundarypoly, count

         Else
            MsgBox ("noboundary")
   End If

   ssetobj.Delete

   Set ssetobj = ThisDrawing.SelectionSets.add("stuff")
   '-------------------------------------------------------look for buildings in christians darwing on layer 00 site blds-------------------------
   'and extract points
```

```
   dataValue(0) = "blds"
   groupCode = gpCode
   dataCode = dataValue
   ssetobj.Select Mode, , , groupCode, dataCode
   bb% = ssetobj.count
   If bb% = 0 Then
        MsgBox ("click to continue")
      Else
         ReDim polys(bb% - 1) As AcadLWPolyline  '' make polys global for recreation purposes too


         For i = 0 To ssetobj.count - 1
            Set polys(i) = ssetobj.item(i)
            polys(i).Layer = "boundary"
            extractpoints polys(i), count
         Next i 'each polyline

   End If


'-----------------------------------------------------lsubtract bldgs from boundary region and -------------------------


makeboundaryregion 0

  pts = pts - 1

'-----------------------------------------------------look for circles anywhere ? -------------------------
 ssetobj.Delete

  Set ssetobj = ThisDrawing.SelectionSets.add("stuff")

   gpCode(0) = 0
   dataValue(0) = "CIRCLE"
   groupCode = gpCode
   dataCode = dataValue
   ssetobj.Select Mode, , , groupCode, dataCode

   If ssetobj.count > 0 Then  ' found some circles
     count = ssetobj.count



      For i = 0 To ssetobj.count - 1

        Set circ = ssetobj.item(i)
        If circ.Layer = "circles" Then

          pts = pts + 1
          ReDim Preserve originalpoints(1 To pts) As mypoint
          originalpoints(pts).x = circ.center(0) '+ Rnd
          originalpoints(pts).y = circ.center(1) '+ Rnd
          originalpoints(pts).z = 0
        End If

      Next i
   End If

ssetobj.Delete




End Sub
Sub extractpoints(thispoly As AcadLWPolyline, count As Integer)

Dim coords As Variant, lb As Long, ub As Long, apoint(2) As Double, p As mypoint
''Static count As Integer
```

```
 coords = thispoly.Coordinates
        lb = LBound(coords)
        ub = UBound(coords)
        'pts = pts + ((ub + 1) - lb) / 2

        If count = 1 Then ReDim originalpoints(1 To 1) As mypoint

        For j = lb To ub Step 2

           p.x = coords(j): p.y = coords(j + 1)
           apoint(0) = coords(j): apoint(1) = coords(j + 1)


              If unique(p, originalpoints, count - 1) Then
              ReDim Preserve originalpoints(1 To count) As mypoint
              originalpoints(count).x = apoint(0) '+ Rnd
              originalpoints(count).y = apoint(1) '+ Rnd
              originalpoints(count).z = 0
              count = count + 1
              pts = pts + 1


                    If apoint(0) = 0 Or apoint(1) = 0 Then
                       dd = 0
                    End If


              Else
              'MsgBox ("skipping")
          End If

        Next j 'points of each polyline
End Sub
Function gethatchvalue(pos As mypoint) As String
Dim thingy As AcadSelectionSet, thing As AcadHatch
Dim corner1(2) As Double, corner2(2) As Double
Dim gpCode(0) As Integer, name As String
'Dim aline As AcadLine
Dim dataValue(0) As Variant
Dim groupCode As Variant, dataCode As Variant

   Mode = acSelectionSetCrossing
   corner1(0) = pos.x - 3.5: corner1(1) = pos.y - 3.5: corner1(2) = 0
   corner2(0) = pos.x + 3.5: corner2(1) = pos.y + 3.5: corner2(2) = 0
  ' Set aline = ThisDrawing.ModelSpace.AddLine(corner1, corner2)

  ' aline.Update

   Set thingy = ThisDrawing.SelectionSets.add("things")
   gpCode(0) = 0
   dataValue(0) = "HATCH"
   groupCode = gpCode
   dataCode = dataValue
   thingy.Select Mode, corner1, corner2, groupCode, dataCode


   If thingy.count > 0 Then  ' found some stuff
   For Each thing In thingy
      name = thing.Layer
      If name = "00-X-large" Or name = "00-Small" Or name = "00-medium" Or name = "00-large" Then Exit For

   Next thing
   End If
   thingy.Delete
   If name = "" Then gethatchvalue = "ERROR" Else gethatchvalue = name

End Function
Function makeregion(poly As AcadLWPolyline) As AcadRegion
  Dim thepoly(0) As AcadEntity 'thing to use in addregion
  Dim boundary As Variant 'assign with addregion
  Dim boundy() As AcadRegion 'thing you redim
```

```
   Set thepoly(0) = poly  'poly is the polygon
   boundary = ThisDrawing.ModelSpace.AddRegion(thepoly)

   ReDim boundy(UBound(boundary)) As AcadRegion

   Set makeregion = boundary(0)


End Function
Sub checkforboundary(dum As Integer)
Dim thelayers As AcadLayers, alayer As AcadLayer, found As Boolean
  found = False
  Set thelayers = ThisDrawing.Layers
    If thelayers.count > 0 Then
        For Each alayer In thelayers
        If alayer.name = "boundary" Then found = True
        Next
    End If

  If Not found Then Set alayer = ThisDrawing.Layers.add("boundary")

End Sub
Function checkforlayer(aname As String) As AcadLayer
Dim thelayers As AcadLayers, alayer As AcadLayer, found As Boolean
  found = False
  Set thelayers = ThisDrawing.Layers
    If thelayers.count > 0 Then
        For Each alayer In thelayers
        If alayer.name = aname Then found = True
        Next
    End If

  If Not found Then
  Set checkforlayer = ThisDrawing.Layers.add(aname)
  Else
  Set checkforlayer = ThisDrawing.Layers.item(aname)
  End If

End Function

Sub makeboundaryregion(dd As Integer)
  Dim ahole As AcadRegion, boundary As AcadRegion
   Dim totpol As Integer
    checkforboundary 0
 ' totpol = UBound(polys)

      Set boundary = makeregion(boundarypoly) 'globalvar boundarypoly
    ' For i = 0 To UBound(polys)                'globalvar polys()
      '    Set ahole = makeregion(polys(i))
      '    boundary.Boolean acSubtraction, ahole
      '    Set bound = boundary           'global var bound
      '    bound.Layer = "boundary"
     'Next

      Set bound = boundary           'global var bound
      bound.Layer = "boundary"

End Sub
```

Module brownian

```
'20 jan 2010
'adds back in visitor percentage

'12 jan 2010
```

```
'what needed to run the program: base poly.dwg turn off boundary layer before the run, change background MENU
TOOLS/OPTIONS/DISPLAY/COLORS
'a folder called stuffs on C:, relationships.txt. data.txt
'has to save and naming file to get a display of coloured cells when simulation runs
'comments cleaned. bug unsolved yet: dwg and bmp can't capture coloured cells. could well be plot and render bug in autocad.

'dwg/bmp got coloured using visualstyles solved 16 feb 2011 -- choesnah
'for purpose of recording/showing ABM movement, turn off  ThisDrawing.SendCommand ("_vscurrent" & vbCr & "R" & vbCr) -- choesnah
16 feb 2011

' ADD TEXTOUT for analysis 27 jan 2007 -choesnah
' basic skeleton for moving circlees with simple agents who are circlees
' most of the bits not needed are commented out or missing
' included steplength and diameter 5/8/03

' 16th april 2004 simplified clustering with only colours, and integrated voronoi
' with area and perimeter calculations
' to do - aggregate smallest cells and redo voronoi as larger cells
'startdiam added used in reset++

'9th dec get up relations and transfer chum to function rather than data type

Const universe = 200
Const wobble = 45
Const pi = 3.14159
Public pathname As String

Public Type agent
   circleid As Long
   heading As Double
   colour As Integer
   diameter As Double 'formerly known as steplength ,
   steplength As Double
   begin As mypoint
   finish As mypoint
   plus As Double
   minus As Double
   forward As Double
   stuck As Integer
   neighbour As Long 'object id of the friend
   groupnumber As Integer
   spacetype As Integer
   jump As Boolean
   stopped As Integer 'counts up how many consecutive times its been stationary
End Type

Public Const globaldatapathname = "c:\stuffs\data.txt"
Public savestuff As Boolean
Public alllayers As AcadLayers
Public thelay As AcadLayer, boundarylayer As AcadLayer
Public boundary As Acad3DSolid
Public maxgoes As Integer
Public thecircles() As agent 'we dont know how many circles to draw, thus not to put "1 to pts" inside brackets
Public limbocircles() As agent
Public relations() As Boolean
Public startdiam As Double
Public groups As Integer
Public cols() As Integer

Sub dobrownian(tt As Integer)
   Dim walker As agent, c As Integer
   Dim topleft As mypoint, bottomright As mypoint
   Dim acircle As AcadObject, randpoint(2) As Double
   Dim t As Long
   Dim origin(2) As Double
   Dim rellies As String

   startdiam = 2

   Set thelay = checkforlayer("circle_layer")
   Set boundarylayer = checkforlayer("boundary") 'dont erase boundary as well as circles
```

```
  seed = val(InputBox("seed", "any numbers", 978345210))
  ticks = val(InputBox("how may times round the block ", "steps", 500)) 'this is how many steps to run program which is proportional to
mod n =0, so adjust as necessary

  Rnd (-1)
  Randomize (seed)
  savestuff = (InputBox("want to save stuff", "y, folder c:\stuffs") = "y")

  If savestuff Then
     pathname = "c:\stuffs\" + InputBox("type name for saved drawings", "naming the drawings")
     Open globaldatapathname For Append As #1
        Write #1, "counter ¦ number of occupied cells ¦ total area of occupied cells"
     Close #1
   End If

  'ThisDrawing.SendCommand "_erase" & vbCr & "all" & vbCr
  'ThisDrawing.SendCommand vbCr

  thelay.Lock = False
  boundarylayer.Lock = False

  groups = InputBox("how many groups ?", "different groups represented by different colours", 6)
  pts = InputBox("how many people?", "size of population", 70)

  ReDim originalpoints(1 To pts) As mypoint
  ReDim thecircles(0 To pts) As agent     'fills in the array thecircles with the randomly scattered circles
  ReDim limbocircles(1 To pts) As agent
  ReDim relations(1 To groups, 1 To groups) As Boolean              '2d array of compatible agents
  ReDim cols(1 To groups) As Integer
  Dim rel As Integer

  rellies = "c:\stuffs\relationships.txt"

  Open rellies For Input As 2#

  For i = 1 To groups '(n is how many type of agents/colours)

       For j = 1 To groups
             Input #2, rel
             relations(i, j) = (rel = 1)
       Next j
  Next i

  Close #2



For c = 1 To pts 'nd means unbiased sample


  randpoint(0) = random(-universe * 0.8, universe * 0.8)      'choose random x y z the array is autocads way of holding a point
  randpoint(1) = random(-universe * 0.8, universe * 0.8)
  randpoint(2) = 0

  walker.diameter = startdiam 'sizs(Int(random(1, nsize)))
  walker.steplength = walker.diameter
  walker.heading = random(0, 360)

  Set acircle = ThisDrawing.ModelSpace.AddCircle(randpoint, walker.diameter / 2)
  'walker is represented as circle

  walker.circleid = acircle.ObjectID

  If Rnd > 0.2 Then 'change here to control how many percentage of visitors
  walker.colour = Int(random(1, CDbl(groups)))
  Else
  walker.colour = 256
  End If

  acircle.color = walker.colour 'colour the circle by groupnumber
```

```
   acircle.Layer = "circle_layer"
   acircle.Update

   walker.begin.x = randpoint(0)       ' set walker's position to be that same as the circle
   walker.begin.y = randpoint(1)       ' (using my preferred way of defining a point
   walker.begin.z = 0

   thecircles(c) = walker '

Next c

teatime (0) 'a cup of tea is good example of brownian system


End Sub
Sub gestalt(counter As Integer, ci As Integer, cj As Integer)
Dim numcols() As Integer
Dim jj As Integer
ReDim numcols(1 To groups) As Integer


For c = 1 To pts
  originalpoints(c).x = thecircles(c).begin.x + random(0, 0.001)
  originalpoints(c).y = thecircles(c).begin.y + random(0, 0.001)
  If thecircles(c).spacetype = 1 Then
  dd = 0
  End If

  originalpoints(c).spacetype = thecircles(c).spacetype
  originalpoints(c).kuller = thecircles(c).colour
Next c

 voronoi (0)

 Open globaldatapathname For Append As #1

totalareas = 0
totalspacetypes = 0
   For jj = 1 To groups
      numcols(jj) = 0
   Next jj

  For i = 1 To pts
      drawpoly cells(i)
      If cells(i).spacetype = 1 Then
      For jj = 1 To groups
         If cells(i).kuller = cols(jj) Then numcols(jj) = numcols(jj) + 1 'counting up all the polygons that spacetype 1 (occupied) = we know how
many agents clump
         'Write #1, "how many cells occupied now?"; numcols(jj)
      Next jj

      totalspacetypes = totalspacetypes + 1
      totalareas = totalareas + cells(i).area
      End If

 Next i
 erasepolylines (0)
 ThisDrawing.Regen acActiveViewport

 Write #1, counter, totalspacetypes, totalareas ' counter = at what step, totalspacetypes = ??? and totalareas = total area rendered
  For jj = 1 To groups
  'Write #1, numcols(jj);
  Next jj
 Close #1

 ca (0)


For c = 1 To pts
thecircles(c).jump = cells(c).jump
If thecircles(c).begin.x > universe Then
```

```
  dd = 0
  End If
'thecircles(c).spacetype = 0  'this will hapen in reset
Next c

End Sub

Function hitsomething(myself As agent) As Boolean
Dim circ As AcadCircle, pt As Variant, intersect As Boolean

Set circ = ThisDrawing.ObjectIdToObject(myself.circleid)
circ.Radius = myself.diameter * 4
'circ.Update

intersect = False

 'Find the intersection points between thecircles(i) and boundary
      pt = circ.IntersectWith(boundarypoly, acExtendNone)
   If VarType(pt) <> vbEmpty Then
      If UBound(pt) > -1 Then
          intersect = True
      Else
         intersect = False
      End If
   Else
      intersect = False
   End If

hitsomething = intersect 'withboundary(myself)
If Not hitsomething And outside(myself) Then
dd = 0
End If

circ.Radius = myself.diameter / 2 'set back
'circ.Update
End Function

Function intersectwithboundary(myself As agent) As Integer

Dim howfar As Double
howfar = universe * 0.8

If myself.begin.x <= -howfar Or myself.begin.x >= howfar Or myself.begin.y <= -howfar Or myself.begin.y >= howfar Then
   intersectwithboundary = True
Else
   intersectwithboundary = False
End If
End Function

Sub teatime(dummy As Integer)
Dim i As Integer, j As Integer, d As Double, towards As Double, away As Double
Dim w As agent, therad As Double
Dim near As Double, thej As Integer
Dim needed As Double, altered As Integer
 Dim allpolys As AcadSelectionSet

' works by running through all circles and
' 1 if too close then back off one diameter
' 2 if nearest bloke is chum then adopt heading after backoff
'
' searching is done on thecircles array and changes to position and heading are made
' to limbocircles which are copied back to the circles at the end of each generation
' seem to have disabled jump reset just relocates randomly now inside universe * 0.8 11/ 4/06



Dim ci As Integer, cj As Integer

Dim stopped As Integer, muststop As Integer, counter As Integer
counter = 0
```

```
Do
   counter = counter + 1
   ''-----------------------store current circles

   ' check limbo and circles relations
   ' try jumping (first time only) and go home on incoming circles

   For i = 1 To pts
      With thecircles(i)
            If .steplength < 0.001 Then
                  .stopped = .stopped + 1
               Else
                  .stopped = 0
             End If

            If .jump Then
            reset thecircles(i) 'dont bother, just keep walking

             .jump = False
            End If
      End With


      limbocircles(i) = thecircles(i) 'SET LIMBO TO CURRENT STATE
   Next i


   '''----------------------- check the circles and alter heading and/or steplength
   For i = 1 To pts


      For j = 1 To pts


       If i <> j Then 'not myself
            d = distance(thecircles(i).begin, thecircles(j).begin)

           needed = thecircles(i).diameter * 5 '================= trying ratio but from large nbhood not everywhere/ 2 +
thecircles(j).diameter / 2

            ' what is happening here is that we have widened the distance the agent is affected by a chum
            ' so it moves towards things proportionally slowly in the feild of d * 5 or whatever
            If d < needed Then                                   ' very close
               If chums(thecircles(i).colour, thecircles(j).colour) Then
                  slowdown limbocircles(i), d, thecircles(i).diameter * 5 'slowdown when distance = 5* dia
                     ci = i
                     cj = j
                  towards = getangle(thecircles(i).begin, thecircles(j).begin)
                  limbocircles(i).heading = towards  'this one move towards chum

                Else


         '-----still close but not chums  --------------------
               towards = getangle(thecircles(i).begin, thecircles(j).begin)
               away = (towards + 90) Mod 360
               limbocircles(i).heading = away

            End If 'chums
          End If 'close

            If intersectwithboundary(thecircles(i)) Then 'just cheapo to speed up CHANGED TO THE CIRCLES
                  limbocircles(i).heading = (limbocircles(i).heading + 180 + random(-2, 2)) Mod 360
            End If

         End If 'i<> j
      Next j
   Next i

   For i = 1 To pts
     thecircles(i) = limbocircles(i)
```

```
    carefullypush thecircles(i), thecircles(i).steplength, False

    If outside(thecircles(i)) Then 'double check in case carefullypush didnt work
    vv = 0
    End If

  Next i
  If counter Mod 60 = 0 Then 'Mod 50 = 0 means every 50-th to get gestalt incl. graphics out and print chum
                     'HOW to show/record ABM movement  is it no gestalt???? is it big number of Mod???
    gestalt counter, ci, cj

    If savestuff Then


    'ThisDrawing.SendCommand "shademode" + vbCr + "_l" + vbCr    this line does not work 16 feb 2011
    ThisDrawing.SendCommand ("_vscurrent" & vbCr & "R" & vbCr)  'get coloured cells graphic output - 16 feb 2011

    Set allpolys = ThisDrawing.SelectionSets.add("allofit")
       allpolys.Select acSelectionSetAll
       ThisDrawing.Regen acActiveViewport
       ThisDrawing.SaveAs (pathname + Str$(counter))
       ThisDrawing.Export (pathname + Str$(counter)), "BMP", allpolys

       allpolys.Delete
    End If

    'unshade
    'ThisDrawing.SendCommand "shademode" + vbCr + "_2" + vbCr    this line does not work 16 feb 2011


    thelay.Lock = True
    boundarylayer.Lock = True
    ThisDrawing.SendCommand "_erase" & vbCr & "all" & vbCr
    ThisDrawing.SendCommand vbCr
    thelay.Lock = False
    boundarylayer.Lock = False


    ThisDrawing.Regen acActiveViewport 'screen view not updating, can not create movie or display movement

  End If
  check_for_stationary (counter)
Loop Until counter > ticks 'saving 5 images

End Sub
Sub check_for_stationary(counter As Integer)

'if an agent has been stationary for three ticks then increase stopped counter by one
' if the stopped counter is more than zero check that any are already stopped
'if so add up counter
Dim numstat As Integer
numstat = 0
Dim i As Integer
For i = 1 To pts
If thecircles(i).stopped > 2 Then numstat = numstat + 1
Next i
End Sub


Sub gohome(a As agent)
a.begin.x = random(-universe * 0.8, universe * 0.8)
a.begin.y = random(-universe * 0.8, universe * 0.8)
'sorry

End Sub
Sub jump(myself As agent)
' get boundary poly's points
Dim sortdists() As pair
Dim howmanycoords As Variant
Dim howmanypoints As Integer
```

```
Dim bounder As AcadLWPolyline, apoint As Variant, dist As Double, ppt As mypoint
Dim jumppoint As mypoint
Set bounder = boundarypoly
howmanycoords = bounder.Coordinates
howmanypoints = (UBound(howmanycoords) + 1) / 2
ReDim sortdists(1 To howmanypoints) As pair

For i = 0 To howmanypoints - 1
   apoint = bounder.Coordinate(i)
   ppt.x = apoint(0): ppt.y = apoint(1)
   dist = distance(myself.begin, ppt)
   sortdists(i + 1).value = dist
   sortdists(i + 1).index = i + 1

Next i

bubblesort sortdists, howmanypoints
apoint = bounder.Coordinate(sortdists(howmanypoints).index - 1)
dist = sortdists(howmanypoints).value
myself.finish.x = apoint(0)
myself.finish.y = apoint(1)
myself.heading = getangle(myself.begin, myself.finish)
If outside(myself) Then
d = 0
End If
myself.jump = False
End Sub

Sub reset(myself As agent)
 Dim dist As Double

myself.finish.x = random(-universe * 0.8, universe * 0.8)
myself.finish.y = random(-universe * 0.8, universe * 0.8)
dist = distance(myself.begin, myself.finish)
myself.heading = getangle(myself.begin, myself.finish)
myself.diameter = startdiam 'sizs(Int(random(1, nsize)))
myself.steplength = myself.diameter
myself.spacetype = 0
If outside(myself) Then
d = 0
End If
myself.heading = random(0, 360)
End Sub
Function outsidebegin(a As agent) As Boolean
outsidebegin = a.begin.x < -198 Or a.begin.y < -198 Or a.begin.x > 198 Or a.begin.y > 198
End Function
Function outsidefinish(a As agent) As Boolean
outsidefinish = a.finish.x < -198 Or a.finish.y < -198 Or a.finish.x > 198 Or a.finish.y > 198
End Function
Function outside(a As agent) As Boolean
outside = a.begin.x < -198 Or a.begin.y < -198 Or a.begin.x > 198 Or a.begin.y > 198 Or a.finish.x < -198 Or a.finish.y < -198 Or a.finish.x >
198 Or a.finish.y > 198
End Function
Sub slowdown(myself As agent, dist As Double, maxdist As Double)
' if you are near a compatible object then reduce steplength proportionally to the distance
'between you and the other guy (only called for agents withing maxdist of each other added 11 april 06
Dim ratio As Double 'proportion of dist represented by diameter

ratio = dist / maxdist
 If ratio > 0 Then

 myself.steplength = myself.steplength * ratio
      If myself.steplength < 0.001 Then  'was veryslow
            myself.spacetype = 1
      Else
            myself.spacetype = 0
      End If
  End If


End Sub
```

```
Sub slowdown2(myself As agent, dist As Double)
' if you are near a compatible object then reduce steplength proportionally to the distance
'between you and the other guy
Dim ratio As Double 'proportion of dist represented by diameter
  If dist > 0 Then

          myself.steplength = myself.steplength * 0.5
             If myself.steplength < 0.1 Then  'was veryslow
                  myself.spacetype = 1
                Else
                  myself.spacetype = 0
             End If

   End If
End Sub


Function polar(here As mypoint, length As Double, angle As Double) As mypoint
Dim therad As Double

therad = (angle / 180 * pi) 'look left , therad = the radian
polar.x = here.x + length * Cos(therad)
polar.y = here.y + length * Sin(therad)
polar.z = 0
End Function


Function chums(i As Integer, j As Integer) As Boolean
' to see if agent i should follow agent j
chums = False

If i >= 1 And i <= groups And j >= 1 And j <= groups Then
  chums = relations(i, j)

End If
    ci = i
    cj = j
End Function

' push gets the circle id being carried by the agent and convert the objectid
' into an object. then it can move it

Sub carefullypush(myself As agent, distance As Double, jumping As Boolean)
Dim thecircle As AcadCircle
Dim start(0 To 2) As Double, finish(0 To 2) As Double
   myself.finish = polar(myself.begin, distance, myself.heading)
   myself.finish.z = 0
   If outsidebegin(myself) Then
   d% = 0
   End If
   If outsidefinish(myself) Then
   d% = 0
   End If
 Dim home As Boolean
   If outside(myself) Then ' see if this new point is outside the universe
      gohome myself 'jump agent to 0 0 0
      home = True
      myself.finish = polar(myself.begin, distance, myself.heading)

   End If

   convert myself.begin, myself.finish, start, finish
   Set thecircle = ThisDrawing.ObjectIdToObject(myself.circleid)  ' convert to object
   thecircle.center = start
   thecircle.Move start, finish                       ' move it
   myself.finish.z = 0

   thecircle.color = myself.colour
   thecircle.Update
```

```
   myself.begin = myself.finish                         ' move myself

    If outside(myself) Then
    ff% = 0
    End If
End Sub

Sub convert(b As mypoint, f As mypoint, start() As Double, finish() As Double)
start(0) = b.x
start(1) = b.y
start(2) = b.z
finish(0) = f.x
finish(1) = f.y
finish(2) = f.z
End Sub
```

Module friends

```
'**********************************general sub-routines and functions*************************
'
'friendly subs and functions include:      description:
'
'      distance2d()                  distance between 2 points in 2d
'      distance3d()                  distance between 2 points in 3d
'      random()                      calculates random number between two limits
'      askpoint()                    prompts user for input points on screen
'      howmany()                     creates dialog box and prompts user to input number
'      bubblesort()                  sorts an array hierachically according to some criteria
'      findpoint()                   calculates point with a given angle and distance
'      findpointZ()                  calculates z-value of point given height and distance
'      getangle2d()                  calculates angle between two points in 2d
'      copypt()                      copies one array into another
'      wipe()                        erases everything in the drawing
'      load_table()                  reads a txt file in and stores them in a table
'      hexdec()                      converts a hexidecimal string to a decimal number
'      bit_swtich()                  switches a bit on in a byte (translation from 'C')
'      snap_off()                    toggles the snap mode of the current viewport
'      open_dwg()                    opens a drawing
'      sel_set_del()                 gathers existing selection sets and delets them
'      flipcoin()                    50/50 % chance to get either 1 or -1
'      wipe_layer()                  erase objects on a specific layer
'
'***********************************************************************************************

Public Const pi = 3.14159

Public Type point
   x As Double
   y As Double
   z As Double
End Type

Public Function distance2d(here As point, there As point)

   Dim dx As Double, dy As Double

   dx = (here.x - there.x) ^ 2
   dy = (here.y - there.y) ^ 2

   distance2d = Sqr(dx + dy)

End Function

Public Function distance3d(here As point, there As point)

   Dim dx As Double, dy As Double

   dx = (here.x - there.x) ^ 2
```

```
    dy = (here.y - there.y) ^ 2
    dz = (here.z - there.z) ^ 2

    distance3d = Sqr(dx + dy + dz)

End Function



Public Sub askpoint(apoint() As Double)

    Dim token As Variant

    token = ThisDrawing.Utility.GetPoint(, "Enter a point: ")   'has to work with variants and no arrays
    apoint(0) = token(0): apoint(1) = token(1): apoint(2) = 0

End Sub

Public Function howmany(what As Integer) As Integer

    Dim message As Variant, title As Variant, default As Variant

    If (what = 0) Then

        message = "what gridsize"
        title = "grid"
        default = "3"

    Else

        message = "number of effectors"
        title = "effectors"
        default = "3"

    End If

    ' Display message, title, and default value.
    howmany = InputBox(message, title, default)

End Function


Public Sub findpoint(here As point, angle As Double, length As Double, there As point)

    Dim radianang As Double

    radianang = (angle / 180) * pi
    there.x = here.x + (length * Cos(radianang))
    there.y = here.y + (length * Sin(radianang))
    there.z = here.z

End Sub

Public Function findpointZ(height As Double, distance_to_xy_coor As Double) As Double

    findpointZ = Tan(height / 180 * pi) * distance_to_xy_coor       'take distance2d()

End Function

Public Function getangle2d(st As point, fin As point) As Double

    Dim q As Integer, head As Double, add As Double
    Dim xd As Double, yd As Double, r As Double

    ' calculate quadrant
    If fin.x > st.x Then
        If fin.y > st.y Then
            q = 1
        Else
            q = 2
        End If
```

```
  Else
     If fin.y < st.y Then
        q = 3
     Else
        q = 4
     End If
  End If

  Select Case q

     Case 1
        xd = fin.x - st.x
        yd = fin.y - st.y
        If xd = 0 Then
           r = pi / 2
        Else
           r = yd / xd
        End If
        add = 0

     Case 2
        yd = st.y - fin.y
        xd = fin.x - st.x
        add = 270
        If yd = 0 Then
           r = pi / 2
        Else
           r = xd / yd
        End If

     Case 3
        xd = st.x - fin.x
        yd = st.y - fin.y
        If xd = 0 Then
           r = pi / 2
        Else
           r = yd / xd
        End If
        add = 180

     Case 4
        xd = st.x - fin.x
        yd = fin.y - st.y
        If yd = 0 Then
           r = pi / 2
        Else
           r = xd / yd
        End If
        add = 90

  End Select

  If xd = 0 Then
     getangle2d = 90 + add
  Else
     getangle2d = ((Atn(r) / pi) * 180) + add
  End If

End Function

Public Sub copy(a As point, b() As Double)

  b(0) = a.x
  b(1) = a.y
  b(2) = a.z

End Sub

Public Sub wipe(token As Integer)

  ThisDrawing.SendCommand "erase" & vbCr & "all" & vbCr & vbCr
```

```
End Sub

Public Sub load_table(token As Integer)

   Dim table(256, 16) As Integer   'change to whatever table you want to create
   Dim name As String

   '---make sure you define the whole path to the file & !!! put '-1' at the end of the string to be read which means that the end of line is
reached
   name = "Table3D.TXT"           'exchange the name of the table with full path

   Open name For Input As 1
      f = CStr(Input$(LOF(1), #1))
   Close

   '---store the values in an array called table
   t1 = Split(f, "{")             'change the symbol of the delimiter accordingly
   For i = 0 To UBound(t1)
      t2 = Split(t1(i), ",")      'change the symbol of the delimiter accordingly
      For j = 0 To UBound(t2)
         table(i, j) = val(t2(j))
      Next
   Next

End Sub

'***********************************************************************************************************
'****************************converts a hexadecimal string to a decimal number*************************
'***********************************************************************************************************

Public Function hexdec(no As String) As Long

   Dim sel As Boolean
   Dim leng As Integer
   Dim temp As Long, total As Long
   Dim l As String, r As String

   no = Trim(no)          'cuts the empty spaces from the string

   leng = Len(no)

   For i = 1 To leng

      r = Right(no, i)
      l = Left(r, 1)

      sel = False

      Select Case l
         Case "a"
            lef = 10
            sel = True
         Case "b"
            lef = 11
            sel = True
         Case "c"
            lef = 12
            sel = True
         Case "d"
            lef = 13
            sel = True
         Case "e"
            lef = 14
            sel = True
         Case "f"
            lef = 15
            sel = True
         Case " "
            lef = 0
            sel = True
```

```
    End Select

    If (Not sel) Then
      lef = val(l)
    End If

    temp = lef * (16 ^ (i - 1))
    total = total + temp

  Next i

  hexdec = total

End Function

Public Function bit_switch(bit As Integer) As Integer

'---in 'C' one can switch on a bit of a byte separately

  Dim bit_con As Integer

  Select Case bit

    Case 1
      bit_con = 1     '0000000I
    Case 2
      bit_con = 3     '000000II
    Case 4
      bit_con = 7     '00000III
    Case 8
      bit_con = 15    '0000IIII
    Case 16
      bit_con = 31    '000IIIII
    Case 32
      bit_con = 63    '00IIIIII
    Case 64
      bit_con = 127   '0IIIIIII
    Case 128
      bit_con = 255   'IIIIIIII
    Case 256
      bit_con = 512

  End Select

  bit_switch = bit_con

End Function

Public Sub snap_off(token As Integer)

  Dim viewportObj As AcadViewport

  ' Set the viewportObj variable to the activeviewport
  Set viewportObj = ThisDrawing.ActiveViewport

  ' Toggle the setting of SnapOn
  viewportObj.SnapOn = Not (viewportObj.SnapOn)

  ' Reset the active viewport to see the change on the AutoCAD status bar
  ThisDrawing.ActiveViewport = viewportObj

End Sub

Public Sub open_dwg(token As Integer)

  Dim path As String
  ' The following example opens "C:\AutoCAD\Sample\downtown.dwg" file.
  ' This drawing may not exist on your system. Change the drawing
  ' path and name to reflect a valid AutoCAD drawing on your system.
  path = "C:\Documents and Settings\bier\theke\MSc\studens 2002-2003\peter keenan\mesh 2"
```

```
    ThisDrawing.Application.Documents.Open (path)

End Sub

Public Sub sel_set_del(token As Integer)

    Dim selset As AcadSelectionSet

    If (ThisDrawing.SelectionSets.count > 0) Then
        ThisDrawing.SelectionSets.item(0).Delete
    End If

End Sub

'Public Function flipcoin() As Integer

'    flipcoin = IIf((random(0, 10) > 5), 1, -1)

'End Function

Public Sub wipe_layer(name As Variant)

    Dim ss As AcadSelectionSet
    Dim ft As Variant, fd As Variant
    Dim gp(0) As Integer
    Dim dv(0) As Variant

    gp(0) = 8
    dv(0) = name
    ft = gp
    fd = dv

    Set ss = ThisDrawing.SelectionSets.add("it")

    ss.Select acSelectionSetAll, , , ft, fd
    For i = 0 To ss.count - 1
        ss.item(i).Delete
    Next i

    ss.Delete

End Sub

Sub record(name As String, no As Integer)

    ThisDrawing.SendCommand "render" & vbCr & name & vbCr & vbCr
    'ThisDrawing.SendCommand "name" & vbCr

End Sub
```

Module voronoibits

```
   '-------------------------------------- changing datastructure to hold indeces into originalpoints
   '-------------------------------------- rather than points 11.6.03---------------------
    ' defining the cells of the voronoi diagram
    ' working 26 june 03

    Const pi = 3.1415926535
    Const yspace = 0
    Const xspace = 1


    Type pointedge
    pos As point       'position of intersection
    Bedge(2) As Integer 'indeces into  boundary array where intersection occurs
    End Type
```

```
Type intersectStuff
outnode As point
outnodeid As Integer  'index into vertex array for voronoi cell
beforeinter As pointedge
afterinter As pointedge
End Type


Const VERYSLOW = 0.7
Type mypoint
   x As Double
   y As Double
   z As Double
   spacetype As Integer
   kuller As Integer
End Type

Type pair 'to tie the triangle nos to the sorted angles
   value As Double
   index As Integer
End Type

Type delaunay
   p1 As Integer
   p2 As Integer
   p3 As Integer
   circcentre As mypoint      ' the coordinates of the centre of the circle by 3 pts constructed by this point
   circrad As Double          ' the radius of this circle
End Type

Type cell
   item() As Integer
   tot As Integer
   area As Double
   id As Long
   spacetype As Integer
   jump As Boolean
   kuller As Integer
End Type

Public pts As Integer
Public numtriangles As Integer
Public originalpoints() As mypoint
Public triangles() As delaunay
Public cells() As cell
Public neighbour() As cell

Public cyclesmax As Long
Public cycles As Long


Sub voronoi(d As Integer)
 ReDim cells(1 To pts) As cell
 ReDim neighbour(1 To pts) As cell
 Dim i As Integer, j As Integer, k As Integer


 For i = 1 To pts
 cells(i).spacetype = originalpoints(i).spacetype
 cells(i).kuller = originalpoints(i).kuller
 Next i


 cycles = 0
 numtriangles = 0
 'cyclesmax = pts ^ 3

 For i = 1 To pts
    For j = i + 1 To pts
       For k = j + 1 To pts
          ' the triangles array is populated in the sub drawcircle - sorry !!
```

```
            drawcircle_ifnone_inside i, j, k, pts
            cycles = cycles + 1
            'counterform.count_Click
          Next k
        Next j
    Next i

    collectcells (0)      'define data for all voronoi cells
    neighcells (0)       'define

  End Sub
    Sub collectcells(d As Integer) ' populates array cells with lists of all the vertex incident triangles of a point
    Dim v As Integer, N As Integer, t As Integer

    For v = 1 To pts ' go through all the original points
        N = 0
        ReDim cells(v).item(1 To 1)
        ' drawpoint originalpoints(V), acGreen, 2
        ' ThisDrawing.Regen acAllViewports

          For t = 1 To numtriangles 'go through all triangles
            If triangles(t).p1 = v Or triangles(t).p2 = v Or triangles(t).p3 = v Then
              N = N + 1 '' T is index into a tri sharing a vertex with originalcells(V)
              ReDim Preserve cells(v).item(1 To N)
              cells(v).item(N) = t
              cells(v).tot = N
            End If
          Next t
          sortbyangle v, cells(v)
    Next v
    End Sub
    Function centre_gravity(this As delaunay) As mypoint
    Dim tx As Double, ty As Double, tz As Double
    tx = (originalpoints(this.p1).x + originalpoints(this.p2).x + originalpoints(this.p3).x) / 3
    ty = (originalpoints(this.p1).y + originalpoints(this.p2).y + originalpoints(this.p3).y) / 3
    tz = 0

    centre_gravity.x = tx
    centre_gravity.y = ty
    centre_gravity.z = tz

    End Function

    Sub sortbyangle(index As Integer, this As cell)
    Dim angles() As pair, i As Integer, O As mypoint, CG As mypoint
    ReDim angles(1 To this.tot) As pair
    O = originalpoints(index)
        For i = 1 To this.tot
            CG = centre_gravity(triangles(this.item(i)))
            angles(i).value = getangle(O, CG)
            angles(i).index = this.item(i)
        Next i
    bubblesort angles, this.tot
        For i = 1 To this.tot
            this.item(i) = angles(i).index
        Next i

    End Sub
    Sub bubblesort(s() As pair, N As Integer)
     Dim index As Integer, c As Integer, swap As Integer, temp As pair

Do
    swap = False
    For c = 1 To N - 1

        If s(c).value > s(c + 1).value Then
            temp = s(c)
            s(c) = s(c + 1)
            s(c + 1) = temp
            swap = True
        End If
```

```
   Next c
Loop Until (swap = False)

End Sub
Function getangle(st As mypoint, fin As mypoint) As Double

Dim q As Integer, head As Double, add As Double
Dim xd As Double, yd As Double, r As Double
' calculate quadrant
If fin.x > st.x Then
 If fin.y > st.y Then
   q = 1
  Else
    q = 2
 End If
  Else
  If fin.y < st.y Then
    q = 3
  Else
    q = 4
  End If
 End If


 Select Case q

 Case 1
   xd = fin.x - st.x
   yd = fin.y - st.y
   If xd = 0 Then
      r = pi / 2
   Else
      r = yd / xd
   End If
   add = 0
 Case 2
   yd = st.y - fin.y
   xd = fin.x - st.x
   add = 270
   If yd = 0 Then
      r = pi / 2
   Else
      r = xd / yd
   End If
 Case 3

   xd = st.x - fin.x
   yd = st.y - fin.y
   If xd = 0 Then
      r = pi / 2
   Else
      r = yd / xd
   End If
   add = 180
 Case 4
   xd = st.x - fin.x
   yd = fin.y - st.y
   If yd = 0 Then
      r = pi / 2
   Else
      r = xd / yd
  End If
   add = 90
 End Select

 If xd = 0 Then
   getangle = 90 + add
 Else
   getangle = ((Atn(r) / pi) * 180) + add
 End If
```

```
End Function
   Sub neighcells(d As Integer)

   Dim v As Integer, N As Integer, nbs As Integer, cp As Integer

   For v = 1 To pts
   nbs = 0                             'go through the item list for this cell (based on vertex V)
     For cp = 1 To cells(v).tot - 1       'the indeces into array cells
         N = matchupcells(cells(v).item(cp), cells(v).item(cp + 1), v) 'two points on the voronoi region
         If N > 0 Then
            nbs = nbs + 1
            ReDim Preserve neighbour(v).item(1 To nbs)
             neighbour(v).item(nbs) = N
             neighbour(v).tot = nbs
        End If
     Next cp
   Next v
   End Sub

Function matchupcells(p1 As Integer, p2 As Integer, current As Integer) As Integer

' find a cell (in array cells)which shares an edge p1 - p2 with this cell (current)
Dim m As Integer, v As Integer, cp As Integer

matchupcells = 0

      For v = 1 To pts
       If v <> current Then 'dont look at you own list
         m = 0

         'a voronoi region can only share two verteces ( one edge) with any other
         'but since the edges are organised anti clockwise, the neighbouring cell
         'will be going the other way. so here we just look for two matches hope thats ok?
            For cp = 1 To cells(v).tot 'run through vertex list for this cell
               If cells(v).item(cp) = p1 Then m = m + 1
               If cells(v).item(cp) = p2 Then m = m + 1
            Next cp
            If m = 2 Then
               matchupcells = v
               Exit For       'dont go on looking once found a match
            End If
         End If
       Next v
   End Function

   Sub drawcircle_ifnone_inside(i As Integer, j As Integer, k As Integer, pts As Integer)
   Dim testcircle As delaunay

   testcircle.p1 = i
   testcircle.p2 = j
   testcircle.p3 = k
   circbythreepts testcircle
   If Not inside(testcircle, pts) Then
      'drawpoint testcircle.circcentre, acYellow, testcircle.circrad
      numtriangles = numtriangles + 1
      ReDim Preserve triangles(1 To numtriangles)
      triangles(numtriangles) = testcircle
   End If

   End Sub

   Function inside(this As delaunay, pts As Integer) As Integer
   ' are there any points closer to the centre of this circle than the radius

   inside = False
   Dim i As Integer, dd As Double, cr As Double
   For i = 1 To pts
   'ignore points that are on this circle
        If i <> this.p1 And i <> this.p2 And i <> this.p3 Then
```

```
        dd = distance(this.circcentre, originalpoints(i))
        cr = this.circrad
           If (dd < cr) Then
              inside = True
              Exit For
           End If
      End If
 Next i
End Function
Sub circbythreepts(this As delaunay)

Dim a As Double, b As Double, c As Double, k As Double, h As Double, r As Double, d As Double, e As Double, f As Double
Dim pos As mypoint
Dim k1 As Double, k2 As Double, h1 As Double, h2 As Double

a = originalpoints(this.p1).x: b = originalpoints(this.p1).y
c = originalpoints(this.p2).x: d = originalpoints(this.p2).y
e = originalpoints(this.p3).x: f = originalpoints(this.p3).y


'three points (a,b), (c,d), (e,f)
'k = ((a²+b²)(e-c) + (c²+d²)(a-e) + (e²+f²)(c-a)) / (2(b(e-c)+d(a-e)+f(c-a)))
k1 = (((a ^ 2) + (b ^ 2)) * (e - c)) + (((c ^ 2) + (d ^ 2)) * (a - e)) + (((e ^ 2) + (f ^ 2)) * (c - a))
k2 = (2 * ((b * (e - c)) + (d * (a - e)) + (f * (c - a))))

k = k1 / k2

'h = ((a²+b²)(f-d) + (c²+d²)(b-f) + (e²+f²)(d-b)) / (2(a(f-d)+c(b-f)+e(d-b)))
h1 = (((a ^ 2) + (b ^ 2)) * (f - d)) + (((c ^ 2) + (d ^ 2)) * (b - f)) + (((e ^ 2) + (f ^ 2)) * (d - b))
h2 = (2 * (((a * (f - d)) + (c * (b - f)) + (e * (d - b)))))
h = h1 / h2

'the circle center is (h,k) with radius; r² = (a-h)² + (b-k)²
r = Sqr((a - h) ^ 2 + (b - k) ^ 2)

pos.x = h: pos.y = k: pos.z = 0
''drawpoint pos, acYellow, r
this.circcentre = pos
this.circrad = r

End Sub

Sub convert(b As mypoint, f As mypoint, start() As Double, finish() As Double)

start(0) = b.x
start(1) = b.y
start(2) = b.z
finish(0) = f.x
finish(1) = f.y
finish(2) = f.z
End Sub

Function findcenter(pts As Integer) As mypoint
Dim xt As Double, yt As Double

xt = 0
yt = 0

   For i = 1 To pts
      xt = xt + originalpoints(i).x
      yt = yt + originalpoints(i).y
   Next i

   findcenter.x = xt / pts
   findcenter.y = yt / pts
   findcenter.z = 0

End Function


Function findpluto(lots As Integer, center As mypoint) As mypoint
```

```
  Dim i As Integer
  Dim longestend As mypoint, maxdist As Double, thedist As Double

  maxdist = -10000
  For i = 1 To lots
     thedist = distance(center, originalpoints(i))
     If thedist > maxdist Then
        maxdist = thedist
        longestend = originalpoints(i)
     End If
    Next i
    findpluto = longestend
  End Function



Sub Draw_Line(b As mypoint, f As mypoint, c As Integer)
  Dim lineobj As AcadLine
  Dim mLineObj As AcadMLine
  Dim start(0 To 2) As Double, finish(0 To 2) As Double

  convert b, f, start, finish

  Set lineobj = ThisDrawing.ModelSpace.AddLine(start, finish)

  lineobj.color = c
  lineobj.Layer = "delaunay"
  'lineobj.Update


End Sub

Sub drawpoly(this As cell)
  Dim tri As delaunay
  Dim plineObj As AcadLWPolyline
  'changed to lw polyline so only duets of coords not trios
  Dim thepoly(0) As AcadEntity 'thing to use in addregion
  Dim boundary As Variant 'assign with addregion
  Dim boundy() As AcadRegion 'thing you redim
  Dim acell As AcadRegion
  Dim numtri As Integer, thepoints() As Double, TPC As Integer

  numtri = this.tot * 2 - 1
  ReDim thepoints(numtri + 2) As Double
  TPC = 0

  ' loop through all the items getting the coordinates of the circlcentres that are
  ' inside the elements of the thetriangles array

  For i = 1 To this.tot
        thepoints(TPC) = triangles(this.item(i)).circcentre.x
        TPC = TPC + 1
        thepoints(TPC) = triangles(this.item(i)).circcentre.y
        TPC = TPC + 1
       ' thepoints(TPC) = triangles(this.item(i)).circcentre.z
       ' TPC = TPC + 1
  Next i
 thepoints(TPC) = thepoints(0)
  TPC = TPC + 1: thepoints(TPC) = thepoints(1)
 'TPC = TPC + 1: thepoints(TPC) = thepoints(2)

  If TPC > 3 Then
  On Error Resume Next    'got crash on huge poly
      Set plineObj = ThisDrawing.ModelSpace.AddLightWeightPolyline(thepoints)
      If plineObj.area > 0 Then

      Set acell = makeregion(plineObj)

      On Error Resume Next
         acell.Boolean acIntersection, bound
         this.area = acell.area
```

```
        this.id = acell.ObjectID 'changed to acell
        If this.spacetype = 1 Then
        acell.color = this.kuller
        Else
        acell.color = acWhite
        End If

       ' acell.Update
       ' ThisDrawing.Regen acActiveViewport
        makeboundaryregion 0

    End If


End If

End Sub


Sub drawcircle(x As Variant, y As Variant, kuller As Integer, size As Integer)
Dim p(2) As Double, circ As AcadCircle
p(0) = x: p(1) = y: p(2) = 0
Set circ = ThisDrawing.ModelSpace.AddCircle(p, size)
circ.color = kuller
' circ.Update

End Sub
Function random(bn As Double, tn As Double) As Double

    random = ((tn - bn + 1) * Rnd + bn)

End Function


Function distance(startp As mypoint, endp As mypoint) As Double
Dim xd As Double, yd As Double
xd = startp.x - endp.x
yd = startp.y - endp.y
distance = Sqr(xd * xd + yd * yd)
End Function


Sub drawpoint(pos As mypoint, c As Integer, r As Double)
    ' This example creates a point in model space.
    Dim circleObj As AcadCircle
    Dim location(0 To 2) As Double
    location(0) = pos.x
    location(1) = pos.y
    location(2) = pos.z
    ' Create the point
      Set circleObj = ThisDrawing.ModelSpace.AddCircle(location, r)
    circleObj.color = c
    'ZoomAll
End Sub


Sub bigtri(mid As mypoint, longestend As mypoint, pts As Integer)

Dim i As Integer
Dim x(2) As Double, y(2) As Double
Dim startangle As Double
Dim vert As mypoint
r = distance(mid, longestend)
startangle = Atn((longestend.y - mid.y) / (longestend.x - mid.x)) 'define randomly generated start angle
      'get biggest triangle vertices
        For i = 0 To 2
            x(i) = mid.x + (2 * r * Cos(i * pi * 120 / 180) + startangle)
            y(i) = mid.y + (2 * r * Sin(i * pi * 120 / 180) + startangle)

              vert.x = x(i)
              vert.y = y(i)
```

```
            vert.z = 0
           'drawpoint vert, acYellow, 1 'these three vertices are the bounding triangle for the delaunay triangulation
           pts = pts + 1
           ReDim Preserve originalpoints(1 To pts)
           originalpoints(pts) = vert
       Next i
   'ZoomAll

   End Sub



Function askpoint(apoint As mypoint) As Integer
Dim token As Variant
On Error Resume Next

token = ThisDrawing.Utility.GetPoint(, "Enter a point: ") 'has to work with variants and no arrays

If err Then
err.Clear
askpoint = False
Else
apoint.x = token(0): apoint.y = token(1): apoint.z = 0
askpoint = True
End If

End Function


Public Sub load_table(d As Integer)

   'Dim table(256, 16) As Integer   'change to whatever table you want to create
   Dim name As String, t1 As Variant, t2 As Variant

   '---make sure you define the whole path to the file & !!! put '-1' at the end of the string to be read which means that the end of line is
reached
   name = "c:\voronoi textfiles\experiment.TXT"          'exchange the name of the table with full path

   Open name For Input As 1
      f = CStr(Input$(LOF(1), #1))
   Close
   t1 = Split(f, Chr$(13))
   pts = UBound(t1)
   ReDim originalpoints(1 To pts) As mypoint
   '---store the values in an array called table
           'change the symbol of the delimiter accordingly
   For i = 1 To pts - 1
      t2 = Split(t1(i), ",")      'change the symbol of the delimiter accordingly

         originalpoints(i).x = val(t2(0))
         originalpoints(i).y = val(t2(1))
         originalpoints(i).z = 0
   Next

End Sub
```

Appendix 5 CD contents

Thesis2011_Choesnah_Idarti.docx
Thesis2011_Choesnah_Idarti.pdf

Program Folder which contains
- Data.txt
- Relationships.txt
- Readme_program.txt
- Base poly.dwg
- Spatial_languaging.dvb