Turning the design process downside-up

Self-organization in real-world architecture

SCHEURER, Fabian
Chair of CAAD, ETH Zurich

Keywords: Self-organization, emergent design, spatial structure, dynamic simulation

Abstract: This paper describes the latest results of an ongoing research project that aims at using the power of self-organization for the design and optimisation of irregular spatial structures in real-world applications. An example is presented, which uses a growing swarm to define the configuration of randomly positioned columns in a large concrete structure. This agent based simulation, developed in cooperation with the building's architects and engineers, was successfully used in the final design stage of a project in the Netherlands to resolve the conflicting structural and functional requirements arising from the initial design.

1 INTRODUCTION

Irregular spatial structures seem to be a rising phenomenon in contemporary architecture. Recent examples include the "bird's nest" hull design for the Chinese National Stadium by Herzog & de Meuron (ARUP 2004a) the "foam" structure of the Chinese National Swimming Centre by PTW Architects (ARUP 2004b; Bull 2004) and the "fractal" façade of the Federation Square complex in Melbourne by Lab Architecture Studio (Lab-Architecture 2001). As the descriptive names suggest, the form of those structures is often inspired by natural processes. The design is based on a metaphorical idea and a limited number of components: welded nodes and steel beams in the case of the Beijing stadiums, glass and stone panels at Melbourne's Federation Square. However, they are not – or not at first sight – repetitive in the sense of classical machine production but consist of highly individual parametric parts and pose immense challenges to realization.

Thanks to the advances in computer aided manufacturing (CAM) it is possible to produce large numbers of parametrically shaped parts at reasonable cost. But CAM requires exact production data, and defining the exact location, orientation and geometry of structurally and functionally interdependent components has become a big challenge. This is especially true when the arrangement of those components is important equally for the aesthetic impression and the structural integrity of the building. The complexity of the building process shifts from the production phase towards the design phase.

One way to handle this is to reduce complexity by using geometries that are repetitive on a level that is not recognizable at first sight, like in the mentioned examples by Lab and PTW. This paper shows – by example of a real world project – how a design idea based on a naturalistic metaphor, can be realized by using an agent-based approach to deliver exact data of a *non-repetitive* structure.

2 GROWING A FOREST OF COLUMNS

In early 2003 KCAP, the Rotterdam office of architect Kees Christiaanse, approached the chair of CAAD at the ETH Zurich with a request to help them growing a "forest of columns". KCAP had been awarded the contract to design a large semi-underground bicycle parking facility with a pedestrian area on top for the main railway station in the Dutch city of Groningen. The proposed design consisted of a large concrete slab with a number of openings and incisions for ramps and stairs, which was held up by some 150 slender columns. To increase the notion of lightness those columns should not be aligned in a regular pattern but stand in random positions, be of three different diameters and gently lean in arbitrary directions – just like trees in forest (see Fig. 1). Since the form of the slab, the position of stairs and ramps, the paths for cycling and walking and the locations of the bike stands had already been more or less defined due to outside conditions, the challenge was now to find the optimal arrangement for the columns.

Fig. 1 Design sketches (by permission of KCAP)

2.1 Defining the task

The architects at KCAP were facing a complex puzzle: they had to exactly define the position, inclination and diameter for over one hundred columns so that the following functional and constructive requirements were fulfilled:

- No column should obstruct a predefined walking or cycling path.
- The diameter (and resulting bearing capacity) of each column had to be sufficient for the portion of the slab it was holding up – depending on the distance to its neighbouring columns.
- The distance of the columns had to be aligned with the spanning and

cantilevering capacity of the slab, especially at the edges and nearby expansion joints.

- The columns in the centre part of the structure were required to stand closer together because of glass blocks within the concrete slab (reducing its spanning capacity).
- The number of columns and their diameter should be minimised, saving costs.

Not only were the properties of a single column mutually dependent (changing the inclination would change the positions of the column's head and foot) but every change in a column would directly influence its surrounding (the column's bearing capacity influences the optimum distance to its neighbours). Therefore every change in a single column would propagate through the whole structure and eventually influence every other column.

Knowing full well that the slab design was still subject to slight changes, and that every change in the layout would mean to redefine the properties of every single column, KCAP was looking to automate the placement of the columns. KCAP undertook a parametric approach to the design as they already had experience in the design of irregular structures on another field:

In the year 2000 KCAP and the University of Kaiserslautern began the “Kaisersrot” project to develop new methods of urban development based on bottom-up principles. To allow rapid testing of design rules, CAD software tools were programmed which enabled the planners to systematically describe inter-dependencies in urban structures and iteratively generate urban plans according to rules and user interactions. The main focus was on the dispersion of plots within a larger development area following the conflicting demands of the plot owners (Kaisersrot 2004). The similarities to the dispersion of columns under a slab seemed to be obvious.

2.2 Designing a dynamic model

The simulation of complex adaptive systems like cellular automata and swarms has been widely used to explore spatial organisation (Coates and Schmid 1999; Frazer 1995; Krause 1997; Miranda Carranza and Coates 2000). Following the concept of Kaisersrot, the basic idea for the solution of the Groningen project was to create a simulation that takes every column as an autonomous agent that tries to find its place within a habitat, competing with the other agents and governed by the local criteria mentioned above. However there was one important difference: in the Kaisersrot project the number of agents was fixed, here the number of columns should be an outcome of the optimisation process. The agents therefore had to be created "as needed" at run-time of the simulation, demanding for an additional mechanism of birth (and dead). In a first step, an abstract definition of the simulation was defined, closely modelled after the requirements described in 2.1.

2.2.1 The habitat

The habitat is the part of the system that is designed entirely top-down: by the formal and functional requirements of the architects' design:

- The outline of the slab with the incisions for the stairs and ramps.
- The locations and diameters of the holes in the slab.
- The location of two expansion joints cutting through the slab.
- The centre lines of the defined walking and cycling paths.
- The distance between the floor and the slab (the height of the basement)

Also in the habitat the most important global parameters are defined: The spanning and cantilevering capacity in different parts of the slab. The values have been calculated by the engineering partner of the project, the Amsterdam office of Ove Arup and Partners and were translated into two properties:

- The optimum distance of two columns in regard with their diameters.
- The optimum distance of a column to the slab edge or an expansion joint in regard with its diameter.

By translating the constructive real-world requirements into these two "rules of thumb", the state changes of the model could be calculated very quickly and at a sufficient accuracy while also ensuring that whenever it reached a state of equilibrium, it was complying with the constructive needs and therefore "buildable".

2.2.2 The column agents

The state of an agent is defined by the locations of its top and bottom end and its diameter, which can be of three different predefined values. It is moving in the habitat according to the following rules:

- Both ends of each agent can move freely within the horizontal planes defining the floor and the slab, but the agent tries to keep its inclination angle below a given threshold value by vertically aligning its ends.
- Each agent tries to keep clear of the walking and cycling paths by moving its bottom end away from the paths' centre lines. In combination with the last rule this also "drags" the top end away.
- The movement of the top ends is confined by the area of the slab outline and the desired distance to the edge is defined by the cantilevering capacity of the slab. The same principle applies at the expansion joints and the holes.
- Each agent tries to hold its neighbouring agents at an optimal distance in regard to the local spanning capacity of the slab and the bearing capacity of the columns. The distances are calculated by a "rule of thumb" that associates a circular part of the slab to the top end of each column,

according to its bearing capacity.

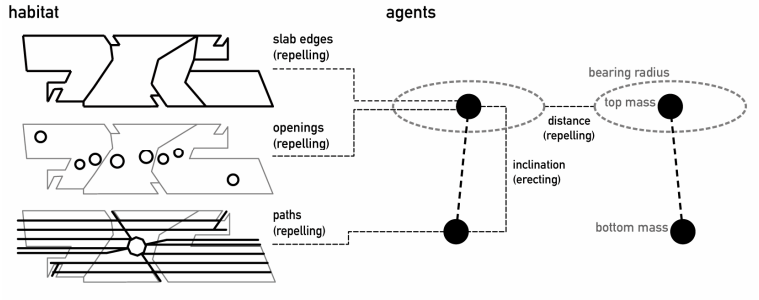

Fig. 2 Relations between agents and environment

In order to adapt to their environment, the agents are able to change not only their location, but also their internal state using the following mechanism:

- The agents measure the “pressure” which is applied to their top ends by close neighbours and the slab’s edges. If it exceeds a certain threshold there are obviously more columns around than needed. It may reduce its diameter and therefore its bearing capacity by one step, allowing its neighbours to come closer (Fig. 3 top-right). Conversely, if an agent encounters too little pressure, there are not enough columns around to hold up the slab. It may increase its diameter by one step and therefore its bearing capacity, trying to bear a larger portion of the slab (Fig. 3 top-left).
- If the pressure does not drop after the agent reached its smallest state, it may eventually "die" and completely remove itself from the habitat to make room for the others, decreasing the number of columns (Fig. 3 bottom-right). If the maximum diameter is reached without generating a sufficient surrounding pressure, the agent "splits" into two agents of the smallest diameter, increasing the number of columns (Fig. 3 bottom-left).

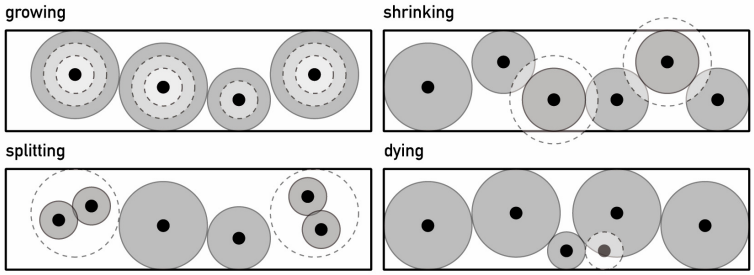

Fig. 3 Growth and death of column agents

2.3 Programming the simulation

This abstract model was then used to develop a computer simulation model. The main requirement for the program was to be interactive, e.g. the designer should be able to directly interfere with the running simulation by changing parameters and positions of single columns. As a result a graphical user interface was required which presented the results of the simulation to the user, preferably in three dimensions, and provided input facilities for the relevant parameters. It also had to be dynamic, and responsive, doing at least 10 simulation steps per second. This was achieved by programming the simulation in Java and using the Java 3D API for real-time rendering.

The definitions for the dynamic behaviour of the agents given in 2.2.2 were translated straightforwardly into a particle simulation with springs, attractors and repellers. Particle dynamics are a simple but powerful way of simulating the behaviour of objects that have a mass but no spatial extent (Witkin and Baraff 1997). By connecting particles with damped springs, complex non-rigid structures can be easily simulated, as has been used for example in real time structural analysis (Martini 2001). In computer graphics, Greg Turk used particle systems to re-tile irregular surfaces by evenly distributing particles on the basis of local repulsion (Turk 1992), here it is used to disperse columns . The top and bottom ends of a column are particles connected by a spring, which tries to align them vertically (This spring resembles the actual column, rendered in the 3D view of the simulation). A circular area around the top end marks the virtual bearing capacity of the column, which is dependent on the column diameter. The column tops are repelling each other by force fields which try to keep the distance so that the circles of neighbouring column tops just touch. The edges of the slab and the expansion joints are linear repellers, pushing the column heads away, the centrelines of the paths do the same for the column feet. The openings in the slab are defined as repeller points. The adaptation of the column diameter and the splitting and dying is triggered by changes in the pressure and agent experiences by neighbours and other repellers The pressure is accounted by simply adding up the pushing forces applied to the top of the column. A similar mechanism to adapt the state and number of particles has been used in computer graphics to sample implicit surfaces (Witkin and Heckbert 1994).

2.4 Letting loose the columns

Since they are able to multiply, it is sufficient to start with a single column that is thrown randomly into the habitat. It will immediately start growing and eventually it will split into two small columns and so on. Just after twenty cycles (approx. 2 seconds) the whole slab is filled by the bearing circles of the columns (Fig. 4). A colour coding in the display of the running simulation is based on the kinetic energy of the column tops, marking the hot spots where still a lot of pushing is happening in red whereas the parts of the slab that already found an equilibrium state are blue. Other colour schemes visualize the distribution of the three column diameters and the inclination angles of the columns to identify problem zones at runtime.

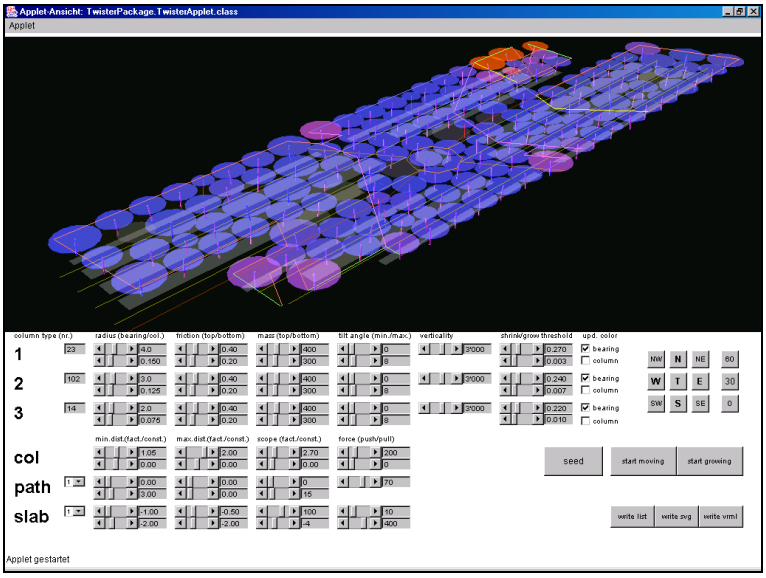

Fig. 4 Screenshot of the running simulation

The model can be influenced by changing the various parameters, for example the strength of the repelling forces, the minimum distances from the path lines and a viscous drag that globally slows down the movement of the columns. It is also possible to pick single columns and drag them to desired locations while the simulation is running.

2.5 From generating to building

The development of the software was done in close cooperation with the architects and engineers and over a number of iterations the model was gradually refined to a final state. Because the designers were familiar with the development versions of the program, they were able to work with the final version almost immediately. It took only about a day to create a few alternative versions of column placements. These parametric designs were then checked by the engineers using finite element simulations to pick out the best performing design. To our relief only a few minor (and local!) corrections had to be made before the design could be finalized. Construction work on the site started in the autumn of 2004 and the building should be finished by mid 2005.

3 RESULTS

Apart from the physical construction that is materialising at the Groningen main station, much knowledge has been gained through this showcase project. The most important issues are briefly described in the following section.

3.1 Design for Emergence

As in any agent based simulation that is aiming at a certain goal, the fundamental design question is: Given the functional requirements, how should the agents be designed so that their emerging behaviour fulfils the given task? (Pfeifer and Scheier 2000) Although there is currently no general methodology for what is called “design for emergence”, it seems in many cases clear how to proceed. Nevertheless, one is easily trapped by preconceived ideas about the desired outcome. The simulation rules should be reduced as far as possible – as Paul Coates put it: “Baroque rule sets are self-defeating, they already specify the majority of the problem solution and anyway, they take too long to program.” An example: In a first version, the lower column ends were attracted by polygonal shapes which KCAP had defined as areas for bike stands and the only places where columns were supposed to stand. This resulted in obvious problems because no bike stands were in the central area of the plot and therefore no chance for columns to survive there. The designers simply had thought one step too far by wanting the columns between the bike stands. The actual requirement was that the columns had to get out of the way and not block the paths. After the model was changed accordingly and the paths were introduced as repellers, the columns neatly arranged themselves between the bike stands but also around the roundabout in the middle without obstructing the ways.

3.2 Calming the system

Since the whole simulation is a non-linear *dynamic* system, but the result should be *static* – a structure that can be built from concrete, steel, glass and other rather inflexible materials – the crucial objective is to bring the system to a stable state. In the theory of dynamic systems this is called a *point attractor* in the phase space of the system, e.g. a state where the system converges towards and finally stops changing unless some external influence disturbs the calm. Abrupt changes can easily bring the system into a state of cyclic, quasi-periodic or even chaotic motion from where it never finds to an equilibrium state anymore. In the Groningen example, the growing and shrinking of the columns is a source of significant disturbance, because it is not continuous but happens in three discrete steps. Every time a column grows or shrinks, it adds a lot of kinetic energy to the system which may amplify itself by triggering growth or shrinkage in neighbouring columns. One way of calming a system in a chaotic state is to increase the "viscosity" of the habitat, which slows down the particles and eventually forces the system into a stable state by simply freezing it. But that might then of course not be an optimum state. The important conclusion here is that changes in the system should be continuous wherever possible.

3.3 Finding the right parameter values

According to Kevin Kelly (Kelly 1997), steering a swarm system resembles the task of a shepherd guiding his herd with gentle interventions and the absence of direct

control could be seen as a disadvantage. Behind the array of sliders that dominate the lower half of the tool's graphical user interface (GUI, see Fig. 4) there are 102 different parameters that influence the behaviour of the system. Thus, finding a set of values that leads to a stable and useful state of the system (see 3.2) needs quite some sensitivity and knowledge about the interdependencies of the parameters. Interestingly this seemed to be no problem for the designers working with the system – they found out very quickly by just probing around and generated a number of valuable alternative configurations within a few hours. An apparent next step would be to switch from the ontogenetic to the phylogenetic timescale and see a model itself as an agent that can be evolved by the use of genetic algorithms (Frazer 1995; Pfeifer and Scheier 2000). The parameters would be encoded into a virtual genome, so that entire populations of models with different genotypes may be created and compared. Selection on the base of a fitness measure – for example the number of columns, their bearing capacity compared to the load of the slab and the change rate – identifies the best performing genome after a specified simulation period and give it the chance to reproduction. This would then of course eliminate the interaction from the optimisation process, since the number of concurrent simulations is too large for detailed observation.

3.4 Programming further simulations

To create a simulation that delivers valuable results the development process requires some knowledge and intuition in the beginning followed by a lot of step by step enhancements based on trial and error. The most time consuming part up to now is the programming of software that simulates the dynamic models and allows for testing and refining of the ideas until the desired results emerge from the system. Implementing the whole simulation in Java has proven to be a realistic approach. To accelerate the development process, the knowledge gained in the Groningen project is currently being used to create a programming toolbox as a base for further simulations. It consists of a number of Java libraries for simulation components (particles, attractors and repellers, springs etc.), GUI components (property editors), renderers (2D and 3D) and import/export interfaces (to and from XML, to VRML etc.). Given the narrow timescales in the architectural process this will be the most important prerequisite for more collaborative work on real-world projects.

CONCLUSION

The Groningen project has proven that the simulation of dynamic adaptive systems is a valuable method for creating and optimising irregular spatial structures within the architectural process, and I am very grateful that KCAP and ARUP believed in the approach and made it happen. However there are some important issues to examine further, as briefly discussed in section 3. New projects are currently being developed at much higher speed by use of the mentioned toolbox, so it will be possible to test different methods to create self-organizing structures more efficiently.

REFERENCES

ARUP. 2004a. Beijing National Stadium, Olympic Green, China. Ove Arup & Partner. Internet. Available from http://www.arup.com/project.cfm?pageid=2184; accessed 11. August 2004.

ARUP. 2004b. The Water Cube - National Swimming Center in Beijing, China. Ove Arup & Partner. Internet. Available from http://www.arup.com/project.cfm?pageid=1250; accessed 11. August 2004.

Bull, Stewart, Downing, Steve. 2004. Beijing Water Cube - the IT challenge. The Structural Engineer, 13.07.2004, 23-26.

Coates, Paul and Claudia Schmid. 1999. Agent Based Modelling. In Architectural Computing from Turing to 2000: 652-661. Liverpool.

Frazer, John. 1995. <<An>> evolutionary architecture. London: Architectural Association.

Kaisersrot. 2004. Build your own neighbourhood. Kaisersrot. Internet. Available from http://www.kaisersrot.com; accessed 02. December 2004.

Kelly, Kevin. 1997. Mehr ist anders - zur Topologie des Schwarms. Arch+, no. 138: 25-32.

Krause, Jeffrey. 1997. Agent Generated Architecture. In ACADIA '97 Conference Proceedings:63-70. Cincinnati, Ohio.

Lab-Architecture. 2001. Federation Square. Lab Architecture. Corporate Website. Available from http://www.labarchitecture.com/; accessed 2004.

Martini, Kirk. 2001. Non-linear Structural Analysis as Real-Time Animation - Borrowing from the Arcade. In CAAD Futures 2001 - Proceedings of the ninth international conference held at the Eindhoven University of Technology. Dordrecht, Boston, London: Kluwer Academic Publishers.

Miranda Carranza, Pablo and Paul Coates. 2000. Swarm modelling - The use of Swarm Intelligence to generate architectural form. In Generative Art - Proceedings of the 3rd international conference, ed. Celestino Soddu. Milano.

Pfeifer, Rolf and Christian Scheier. 2000. Understanding intelligence. Cambridge, MA: MIT Press.

Turk, Greg. 1992. Re-tiling polygonal surfaces. In Proceedings of the 19th annual conference on Computer graphics and interactive techniques, 26:55-64.

Witkin, Andrew P. and David Baraff. 1997. Physically Based Modeling: Principles and Practice. SIGGRAPH 1997 course notes. Available from http://www-2.cs.cmu.edu/~baraff/sigcourse/; accessed 2003.

Witkin, Andrew P. and Paul S. Heckbert. 1994. Using particles to sample and control implicit surfaces. In Proceedings of the 21st annual conference on Computer graphics and interactive techniques: 269-277: ACM Press.