AC 2007-934: BR: AN INTERACTIVE SOFTWARE-PROTOTYPE FOR 3D LAYOUT

Henriette Bier, TU Delft

After graduating in architecture [1998] from the University of Karlsruhe in Germany, H. Bier has worked with Morphosis [1999-2001] on internationally relevant projects in the US and Europe. She has taught computer-based architectural design [2002-2003] at Universities in Austria, Germany and the Netherlands and started a doctoral research at TU Delft [2004]. Her research focuses not only on analysis and critical assessment of digital technologies in architecture, but also reflects evaluation and classification of digitally-driven architectures through procedural- and object-oriented studies. It defines methodologies of digital design, which incorporate [Intelligent] Computer Based Systems proposing development of prototypical tools to support the design process. Results of her research have been published in books, journals and conference proceedings. She regularly leads workshops at universities in Germany, Italy and Belgium, and teaches design studios within Hyperbody Research Group and SpaceLab at TU Delft. Currently, she is project coordinator of the workshop and lecture series on Digital Design and Fabrication within DSD [Delft School for Design] with invited guests from MIT [Massachusetts Institute of Technology] and ETHZ [Eidgenoessische Technische Hochschule Zuerich].

Dave Hoffers, TU Delft

D. Hoffers is student at Delft University of Technology. He participated 2006 in the BuildingRelations project and developed the sub-tool: SizeDefiner.

Matthijs Frederiks, TU Delft

M. Frederiks is student at Delft University of Technology. He participated 2006 in the BuildingRelations project and developed the sub-tool: FunctionDistributor.

Sander Korebritz, TU Delft

S. Korebritz is student at Delft University of Technology. He participated 2006 in the BuildingRelations project and developed the sub-tool: BoundingBox.

BR: An Interactive Software-Prototype for 3D Layout

Abstract

As a research project implemented with graduate students from TU_____, BuildingRelations [BR] deals with the development of an interactive software prototype to support the design process: BR employs bottom-up principles of organization to generate functional layouts exhaustively enabling development of more alternatives than by means of conventional sketching methods mainly because architectural space planning is highly combinatorial, and therefore, difficult to conceive exhaustively by human search means.

Content

Focusing on the development of an interactive design tool which allows simulation of complex design processes, the project proposes an alternative design method based on Swarm Intelligence [SI]. SI is, basically, an Artificial Intelligence [AI] method consisting of agents interacting locally with one another and with their environment similarly to the way fish interact in a swarm and birds in a flock.

In the absence of top-down control dictating how individual agents should behave, local interactions between agents lead to the bottom-up emergence of global behavior. The rules according to which agents interact are simple: C. Reynolds' flocking simulation, for instance, is based on three rules according to which digital birds flock – [1] maintain a minimum distance from neighbors, [2] match velocity with neighbors and [3] move towards the center of the swarm. While these rules are local, establishing the behavior of one agent in relationship to its neighbor, the flock behaves as a whole coherently.

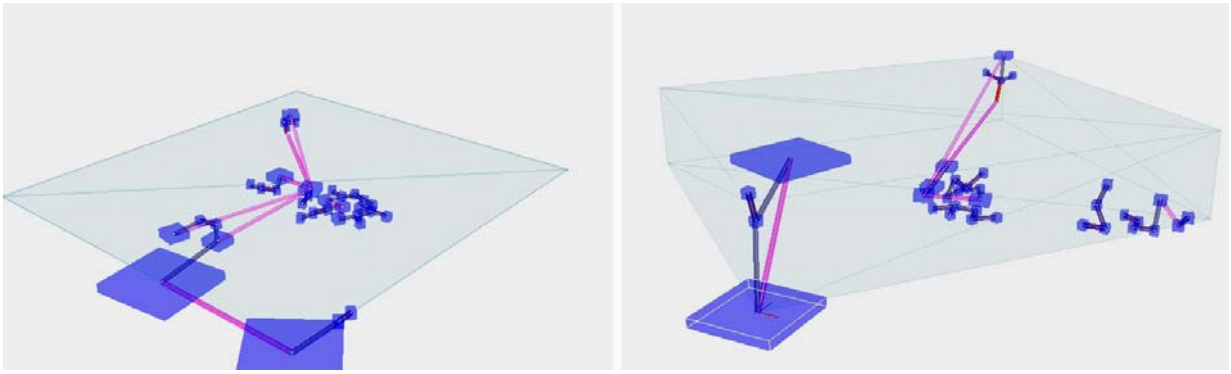

Figure 1: Functional units swarm in the 3D space towards local optimal configurations. Functional units are represented in this example as blue boxes; spatial relations between functional units are represented as magenta connecting-lines.

Similarly, all functional units/objects pertaining to a building can be seen as flocking agents striving to achieve an optimal layout. Spatial relations between functional units can be described as sets of rules, according to which all units organize themselves into specific configurations. This approach is particularly suitable for the functional layouting of large and complex structures: While the architect might find it difficult to have an overview of all functions and their attributed volume and preferential location, the functional units can easily swarm towards local optimal configurations.

Attempts to automate the layout process incorporate approaches to spatial allocation by defining the occupiable space as an orthogonal grid and using an algorithm to allocate each rectangle of the grid to a particular function[2]. However, these approaches limit the design to orthogonal spatial layouts.

Other strategies break down the problem into parts such as topology and geometry: While topology refers to logical relationships between layout components, geometry refers to the position and size of each component of the layout. A topological decision, for instance, that a functional unit is adjacent to another specific functional unit restricts the geometric coordinates of a functional unit relative to another[3].

Based on a similar strategy, BR generates orthogonal as well as non-orthogonal spatial layouts for large, complex layouting problems in an interactive design process. Furthermore, it operates in 3D space and therefore represents an innovative approach to semi-automated design processes.

Implementation

Implemented in a research project with eight graduate students from TU _____, BR is being continued at the time being within the framework of a PhD research.

The one semester long project addressed issues related to concepts and tools development for interactive design as part of the Interactive Architecture [IA] program. The main focus has been the development of methodologies for architectural computer-based applications, which are implemented in interactive 3D simulations and visualizations. Within one semester students developed an understanding for HOW interactive media influences architectural design through object-oriented and procedural studies and were introduced to Virtools as a platform to generate interactively not only designs but also design tools.

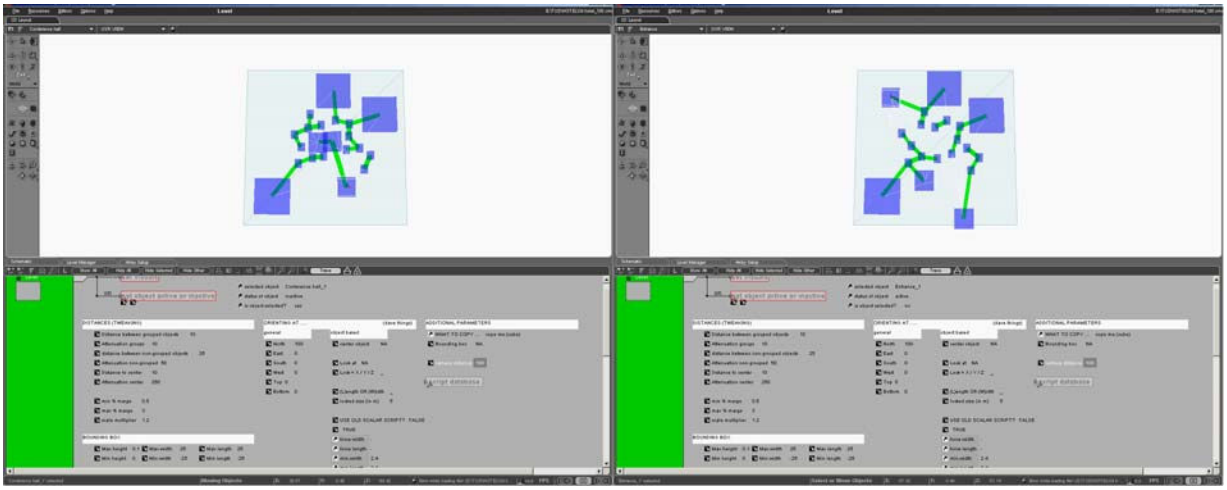

Figure 2: BR-Interface - top view - showing functional units swarming towards local optimal configurations.

Assuming that a design tool is connected to design and design thinking in a fundamental way, students developed a conceptual framework for the interactive design development of 3D hotel layouts. The tool has been divided in sub-tools such as SizeDefiner, FunctionsDistributor, and BoundingBox, which have been developed by individual students.

SizeDefiner is a sub-tool which interactively establishes dimensional relationships and constraints for building components. It is based on data originating from building regulations: These define rules and restrictions regarding minimum floor areas per person - see Dutch Building Regulations 2003.

In its first version SizeDefiner receives the input from the building regulations database and the number of people occupying the space from the user/designer; SizeDefiner then generates the space and scales it to fit the minimum size required by those regulations. A more advanced version incorporates additional functions enabling the user/designer to adjust the number of floors, adjust floor heights and set the width and length of the space by overruling regulation constraints when necessary.

The autonomous working of the script requires Artificial Intelligence: Spatial units/building components establish relationships with other spatial units by determining their distance and automatically adjusting their width, length and height in order to prevent overlaps/collisions. Spatial units, therefore, adjust themselves to their surroundings.

Functional spaces are linked to other functional spaces creating spatial relations defined and simulated with another sub-tool: FunctionsDistributor.

FunctionsDistributor takes, basically, a program of requirements - number of specific spaces, their occupancy numbers, building regulation classes, etc - and translates it in an optimally ordered spatial layout. The optimization is achieved by defining distances between objects of the same type and objects of different type, meaning that objects of the same type cluster while objects of unrelated types disperse.

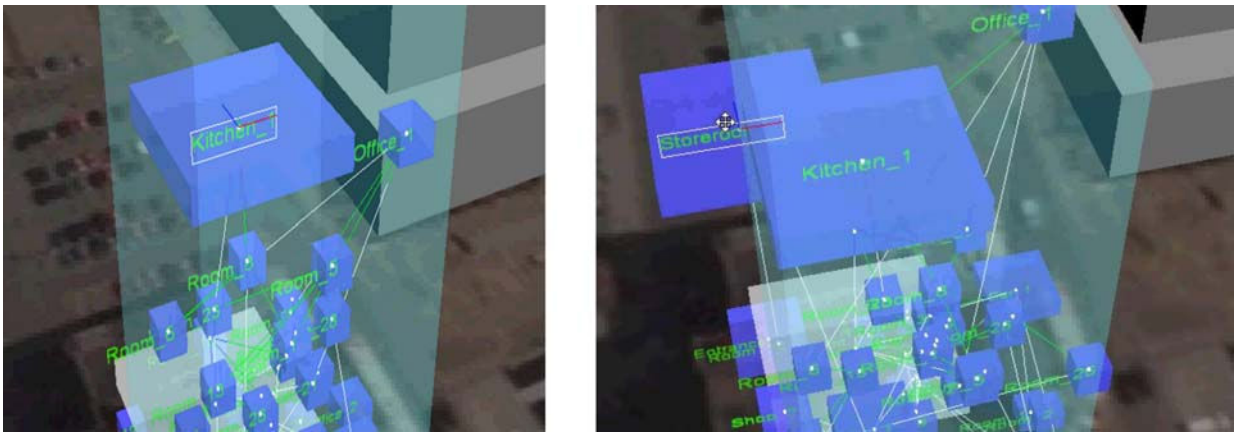

Figure 3: BR-Interface shows 20-30 Functional Objects swarming within the BoundingBox for a hotel in Rotterdam.

The development of this sub-tool involved several steps pertaining 3D spatial configuration, structure and representation:

[1] Organization: Functional objects have configurable distances from each other enabling them to group together or to spread out.

[2] Representation: Relations between objects are represented as magenta lines when relations are established between grouped objects. Green lines indicate nearest object relations. This representation enables reading and evaluation of the dynamic spatial diagram.

[3] Structure: Objects are square or spherical and their orientation is in a grid-like or radial structure respectively. This enables development of specific layout designs.

[4] Scripting and databases/arrays: FunctionsDistributor uses three arrays pertaining to the program requirements, the building regulations, which define the dimensions of the objects, and the master array, which is a dynamic array containing all data about the functional objects.

[5] Min-max component number: Several tests have been implemented in order to define a min-max number of building components. In the end the maximum number of elements has dropped from 100-150 to 50-100 due to increasing complexity of the script.

Developed in Virtools, which is an interactive 3D application, the described software prototype uses a specific strategy to determine/recognize nearest objects: It 'memorizes' and therefore 'remembers' objects involved in object collisions by storing those objects' 3D data in a database. The acquired data can then be used to adjust the width, length and height of the '3D functional objects' automatically in order to prevent collisions with surrounding spaces.

The placement of all functions in 3D space is controlled by the FunctionsDistributor script. The SizeDefiner script generates functions depending on their internal values as well as their external placement defined by the FunctionsDistributor script or by the user.

This system enables adjustment of spaces to their surrounding spaces, whereas distances to other objects are defined by:

[1] The bounding box in which all functional objects are expected to fit – This has been developed as a sub-tool.

[2] The preferred distance to the nearest functional object, which is defined according to the elastic cord principle: the bigger the distance, the harder the object tries to get to the preferred distance.

[3] The preferred distance to the nearest object within the same functional group, which is defined according to the elastic cord principle: the bigger the distance, the harder the object tries to get to a preferred distance.

[4] The preferred distance to a ‘center’ and inside/outside boundaries, which enables functional objects to cluster round a common point according to the elastic cord principle.

In order to avoid overlaps, at the moment one object touches another object collision detection enforces that they both move for 0.1 second in the opposite vectorial direction.

These self-organization mechanisms are complemented by interactivity: The layouting process does not take place exclusively outside the influence of the user/designer. The user can select objects and move them to other places; the model then readjusts to the new configuration. By clicking on an object, the user can 'free' the object from a specific position enabling it to participate in the simulation all over again. In this interactive process, the system and the user/designer search complementarily for an optimum layout of functions.

A third sub-tool, already mentioned, is the BoundingBox, which establishes the boundaries within which functional objects position themselves. It contains real-time editing features which enable form-finding processes pertaining to surface definitions such as NURBS and triangulated meshes.

This sub-tool converts the data defining min-max areas, min-max floor heights, etc. into a geometric model by creating a shape according to the required square meters. By dragging control points the model is recalculated to stay within the pre-determined boundaries, while changing shape.

All data is received and sent continuously from the BR-database to all active sub-tools. The BR-database contains all the information regarding which group functional objects belong to, which objects/groups they may relate to, function type, etc. In addition it establishes connectivities between different software and functions as a parameter pool containing geometric and global data. For instance, a 3D model developed with the BoundingBox sub-tool could be saved to an online database, from which FunctionsDistributor would take data to generate a functional model within the parameters defined by the 3D model.

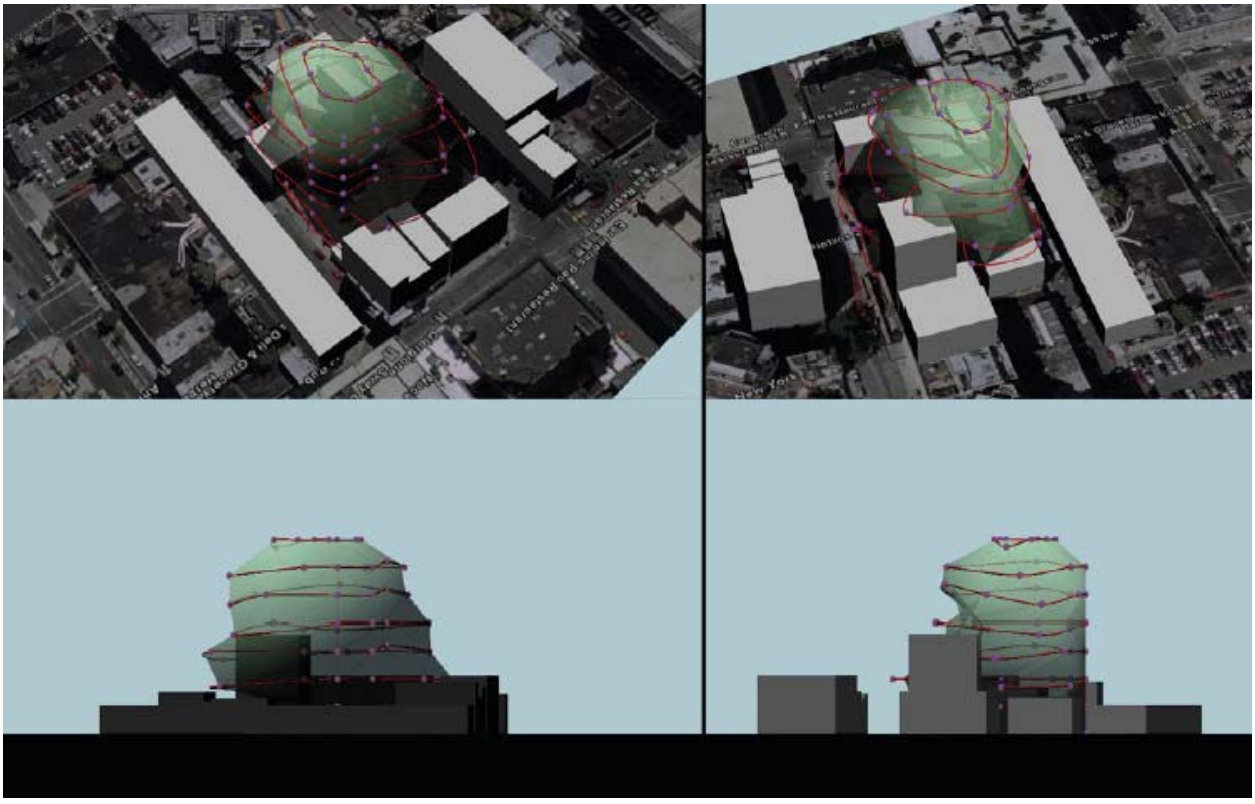

Figure 4: The free-formed BoundingBox establishes boundaries within which functional objects position themselves.

Evaluation

Within one-semester students developed - without previous knowledge of the visual programming language employed in Virtools - a software prototype for design development and concluded that:

[1] BR generates functional layouts exhaustively and enables the designer to develop and consider more alternatives than by means of conventional sketching methods mainly because architectural space-planning is highly combinatorial, and therefore, difficult to conceive exhaustively by human search-means.

[2] Instead of one, BR generates multiple designs and enables informed choices by departing from a singular-design principle, which represents a potentially prejudiced position of the *master*-designer. BR, therefore, does not generate the *ultimate* design but instead offers alternative designs within the spectrum of a relevant solutions-field.

[3] Used interactively and in combination with other software, in order to achieve non-deterministically an optimal design, BR is a design support system, since it supports the designer in the functional layouting process rather than prescribes a solution.

This software prototype has been employed, inter alia, for developing functional layouts for a hotel in Rotterdam incorporating 20-50 rooms. Its current version, even though diagrammatic, demonstrates an obvious capability to support functional layouting of large and complex buildings based on swarm principles, on condition that global optimization mechanisms are incorporated into it - which is the next step in the development of this software.

The inability of this software prototype to generate a global optimum goes back, in part, to the definition of local optima: The software generates endlessly many local optima. It does not, though, generate a global optimum. The further development of this tool aims, therefore, to incorporate global optimization mechanisms; it also aims to address issues related not only to the functional organization of space but also to issues of form generation and spatial coherency.

With respect of education, the intertwining of CAD-issues with design thinking has become relevant on the level where software is design task related and computer skills are not taught outside design projects. Generic CAD-software teaching - as often implemented in architectural education - is similar to teaching material properties and assembly principles unrelated to the respective design tasks: Knowing that both influence each other in a fundamental way, they have to be taught in a way that this connectivity is emphasized and not disregarded.

As a multi-facetted project BR proves that not only intertwining CAD-software teaching with the design studio but also the further development of the CAD-software is beneficial for students: On the one hand scripting empowers them to develop software prototypes able to implement desired design tasks, on the other hand the mechanics of algorithmic definitions forces them to break down design problems into search and solution spaces and establish search methodologies in order to find possible solutions.

This implies development and use of not only design knowledge but also knowledge stemming from a domain outside architecture, namely, computer science raising the question of the necessity to incorporate applied computer science in the architectural curriculum in a similar way structural or material studies have been incorporated. The premise is that students have to acquire literacy in computing and algorithmic definitions applied to architectural tasks: These enable them to develop and appropriate design tools in such a way that these tools not only support but also enhance the design process by incorporating machinic intelligence, and therefore, compensating where human decision-making might be limited or overextended.

Bibliography

1. Reynolds, C.: 1987, Flocks, Herds, and Schools: A Distributed Behavioral Model, Computer Graphics, SIGGRAPH '87 Conference Proceedings.
2. Choudhary, R. and J. Michalek: 2005, Design optimization in computer aided architectural design, International Conference of the Association for Computer Aided Architectural Design Research in Asia.
3. Michalek, J.J., R. Choudhary and P.Y. Papalambros: 2002, Architectural layout design optimization, Engineering Optimization, 34/5:461-484.