VISUALISATION AND RESEARCH STRATEGY FOR COMPUTATIONAL SPATIAL AND STRUCTURAL DESIGN INTERACTION

Dennis Peeten, Ph.D. Student, d.peeten@tue.nl
Herm Hofmeyer, Associate Professor and Vice-Chairman Unit, h.hofmeyer@tue.nl
Unit Structural Design, Department of Architecture, Building, and Planning, Eindhoven Univ. of Tech., The Netherlands

ABSTRACT

A research engine is under development for studying the interaction of spatial and structural design processes. The design processes are being implemented as two separate configurable transformation steps; a conversion step and an optimisation step. A significant part of the spatial-to-structural conversion process together with a first version of a visualisation tool have been finalized. Because the interesting parts of spatial and structural designs are typically situated also on the inside of the design, the main goal of the visualisation tool is to make the innards visible from the outside. This has been realised by rendering translucent polygons with opaque outlines. However, care has to be taken in the order in which the translucent polygons are rendered. For an arbitrary set of convex polygons, finding a back to front order may not be as trivial as sorting a set of polygons on the distance to the viewpoint. Also, sets of polygons exist that do not have an explicit depth order. As a solution to this problem, a render engine has been developed that builds a binary space partition tree (BSP) to store the polygons. Another benefit of this BSP variant is that during the construction of the tree, difficult polygon arrangements are dealt with in an elegant matter. Further research involves the development of a complete version of the research engine. The performance of this first version will be compared to case studies. Based on these results, adjustments and additions to the research engine transformations will be made. The final version of the research engine will also be used to experiment on academic designs in order to develop insights in the fundamental relation between space and structure.

Keywords: spatial design, structural design, computational design, visualisation

1 INTRODUCTION

Design is a multidisciplinary, iterative process. Among others, two important disciplines involved are those of the spatial designer and structural engineer. In a typical design process, the spatial designer first creates a spatial design based on the customer’s requirements. The structural engineer then designs a structure to enable the designer’s spatial design. In practice, it may occur that the structural engineer requires a slight change in the spatial design in order to be able to design a more efficient structure. This feedback starts a new iteration in the design process. Although the spatial designer is mainly concerned with space, functionality, appearance and user comfort, most designers take the structural design possibilities (subconsciously) into account when creating a spatial design. Therefore in reality, the number of feedback iterations and changes made during such spatial designs are generally small.

Scientific research on the disciplines of spatial and structural design can be divided into three groups. There is one group that performs descriptive research which focuses on developing data models to formalise data and their relationships regarding specific aspects of the design process. Data models have been developed for the aspects of the individual spatial and structural design processes (Björk 1992; Weise et.al 2000). However, more important in the context of this project, is the research that focuses on data models that relate spatial and structural design aspects (Martini and Powell 1990; Sause et.al 1992; Nguyen et.al 1996; Khemlani et.al 1998; Matthews et.al 1998; Eastman and Jeng 1999; Rivard and Fenves 2000; Scherer and Gehre 2000).

The second group performs generative research and aims at formalising the design processes by developing procedures and algorithms to automate design tasks. The oldest –but still active– field in this group researches the subject of space allocation, where design requirements are transformed into spatial designs (Oxman 1997; Kotsopoulos 2006; Reffat 2006; Keatruangkamala and Nilkaew 2006). For structural design automation, a distinction should be made between research in methods that optimise an existing structural design (Fuyama et.al 1997; Bletzinger and Ramm 2001; Kicinger et.al 2004; Yulin and Xiaoming 2004; Bletzinger et.al 2005; Mullins et.al 2005; Steirteghem et.al 2005) and methods that transform a given spatial design into a structural design (Maher 1985; Rafiq and Macleod 1988; Sause and Powel 1991; Borkowski et.al 1991; Harty and Danaher 1997; Sacks and Warszawski 1997; Fenves et.al 2000; Markov and Gabriel 2001). Most of the descriptive and generative research assumes a linear design process in which there is a one way path from spatial to structural design. However, the design process can also be modelled as a cyclic process. This model is supported by research in multidisciplinary design from the third group (Haymaker et.al 2004).

The cyclic design process starts with a spatial design for which a structural design is created, the structural design may be optimised or changed in any way that might consequently impact the original spatial design. The optimised structural design is thus transformed back to a spatial design which, in turn, may be improved by the spatial designer. The new spatial design initiates a new design cycle. In the course of the cyclic design process, each design cycle may gradually improve the spatial and structural design. Recent research (Hofmeyer 2007) has shown that fundamental knowledge on both interaction between spatial and structural design and the underlying design process can be acquired using the aforementioned strategy and a scale to evaluate the design characteristics.

This paper will present research that investigates the spatial and structural design process using the cyclic design model. To this end, a research engine is under development that performs the cyclic design transformations. Certain aspects of the design instances as produced by the research engine are evaluated using appropriate units of measurement. The goal of the research engine is threefold. The first goal is to develop an understanding of the interactions between the spatial and structural design processes, that is to say, the effects of specifically chosen transformation methods on the design measures. Secondly, this research project aims at developing fundamental knowledge on the relationship between structure and space by experimenting with specially crafted academic designs. Finally, the design instances as developed by the research engine may be used as design variants in AEC design processes (generative design) and techniques used in the engine like structural grammars, 3D pattern recognition, etc. can be used as add-on to currently used AEC computer tools.

In the following section 2, an overview of the current state of the research engine and the visualisation toolkit will be given. This is followed by section 3 that describes the implementation of each in more detail. Thereafter, the research to be carried out will be presented in section 4. Finally, in section 5, conclusions can be found.

2 RESEARCH ENGINE

As mentioned in the introduction already, the research engine uses a cyclic design process to investigate the interactions between spatial and structural design processes. The processes are modelled as four sequential configurable transformation procedures as depicted in figure 1. On the left, for illustrative purposes, the transformation results can be seen of several iterations of the cyclic process as a spiral. The diagram on the right resembles the processes involved in one iteration of the research engine. For each iteration n, the research engine takes one spatial design as input and incrementally transforms this design into a structural design, an optimised structural design, a new spatial design and finally a modified spatial design. The spatial and structural designs are numbered sequentially as a function of the iteration number n. The following four transformations are performed in the order specified:

1. The first transformation converts a given spatial design $2n$-1 into a kinematically determined structural design $2n$-1.

2. The second transformation takes structural design 2*n-1* and creates an optimised version of the structural design, resulting in structural design *2n*.
3. The third step transforms the optimised structural design *2n* into spatial design *2n*.
4. Finally, the last transformation takes spatial design *2n* from the previous transformation and transforms it into a modified spatial design 2*n*-1 (with *n* increased by 1) which is meant to resemble the original design more closely.

The cyclic process continues with iteration *n* (with *n* increased by 1) by using spatial design *2n-1* as input. The second, third and fourth transformation steps are, in reality, not likely to be performed explicitly, but may model the implicit thought processes performed by the architect and structural engineer.

One of the aims of the research is to develop multiple (configurable) strategies per transformation. The structural optimisation transformation could for example optimise the structure for building costs or optimise the load bearing capacity of the structure, both by means of different (mathematical) optimisation techniques.
In order to quantify the interaction between the two design processes, adequate units of measurement have to be developed to gauge certain properties of the designs. The idea is to use these units of measurement to evaluate each design that results from a transformation step. The measurements can be studied as a function of the transformation strategies chosen and their parameters, in order to get insight in how the designs evolve.

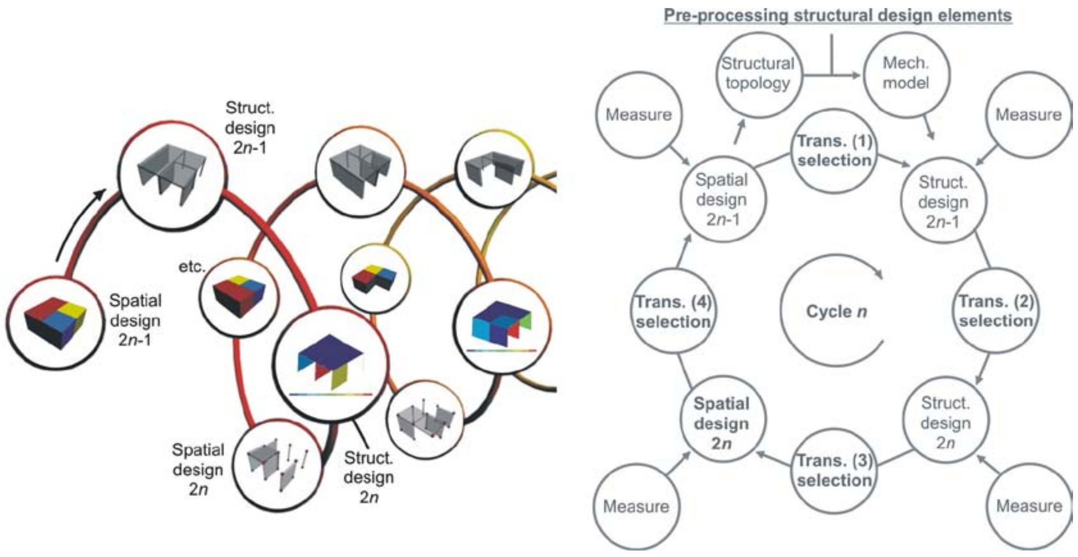

Figure 1: Research engine

The research engine under development contains a visualisation tool that displays the designs as interactive 3D models. In addition to the measurements, these visualisations aid in understanding the interactions between the design processes. Another benefit that the visualisation tool offers is the ability to visually debug the intermediate results of the transformations during the development of the research engine. Verifying the results of a design transformation is easier and less prone to interpretation errors if the design is visualised as an interactive 3D model in contrast of having to interpret the raw data by hand and build mental representations of the design.

3 IMPLEMENTATION

3.1 Research Engine

The implementation of the first transformation step, from spatial design *2n-1* to structural design *2n-1* (see Figure 1), has nearly been finished. The procedure accepts a spatial design consisting of axis parallel rectangular prisms as input and produces a structural design consisting of axis parallel rectangular walls and floors, vertical columns and diagonal trusses. The spatial to structural design transformation procedure performs the following sub-steps in order to arrive at a structural design:

1a. The grouping of the rectangular prisms into larger rectangular prisms, referred to as zones, which play a useful role in the simulated structural design process (Hofmeyer and Kerstens 2006).
1b. By using a structural grammar, the addition to each zone of a set of structural elements like columns, beams, plates, and shear walls (Hofmeyer and Bakker 2008).
1c. Pre-processing the structure's geometry, such that no vertex or edge intersects the interior of another edge or face (Hofmeyer et.al 2010).
1d. Analysis of the pre-processed structure in order to determine the kinematical mechanisms in the structure (Hofmeyer and Russell 2009).
1e. Stabilisation of the structure by intelligently adding structural elements and constraints until no mechanisms remain in the structure (finished, but not yet incorporated).
1f. The transformation of the stabilised structural design into a finite element model and its simulation. This includes the determination of wind and gravity loading and the detailed meshing of the model (under development).

The current implementation performs steps *1a* through *1d*. An automated method for *1e* has been developed, but not yet incorporated into the research engine. The output of steps *1a* to *1d* are shown in Figure 2.

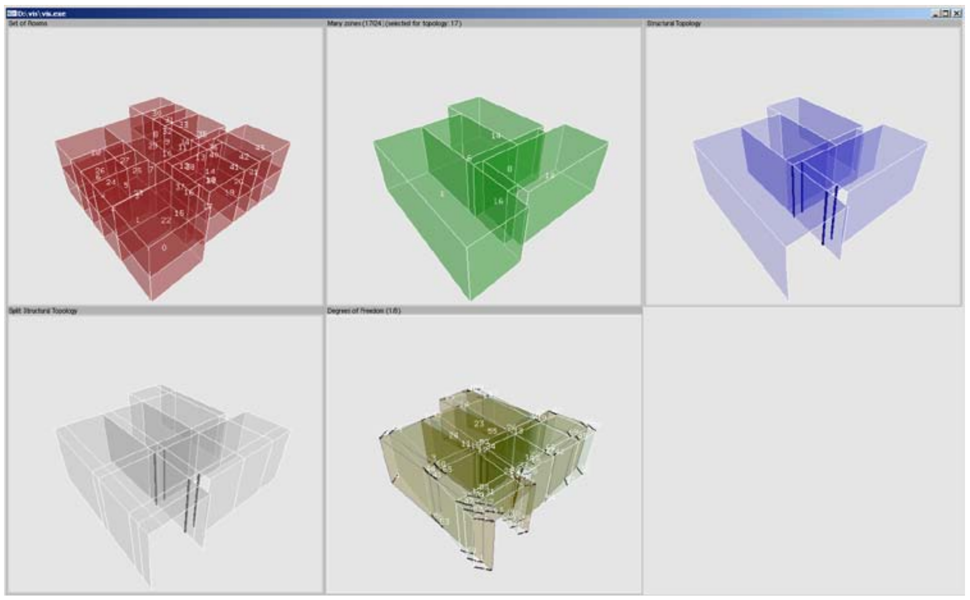
D:\vis\vis.exe
Set of Rooms
Many zones (1704) (selected for topology: 17)
Structural Topology
Split Structural Topology
Degrees of Freedom (18)

Figure 2: Spatial to structural transformation procedure: l.r.t.b: Spatial design (input), Zoned design (step *1a*), Structural design (step *1b*), Pre-processed Structural design (step *1c*), Kinematic mechanism shown as node displacements (step *1d*)

The first step (*1a*) in the spatial to structural design transformation takes the rectangular prisms of the spatial design as input and groups them together into larger rectangular prisms referred to as zones. The resulting sets of zones occupy the same space as the original spatial design. Depending on the spatial design, multiple zone configurations exist that fulfil this requirement. The research engine computes all possible zone configurations, referred to as zoned designs, helped by a technique termed as a "geometrical related reducer" and discussed in (Hofmeyer and Kerstens 2006).

In step *1b*, a rule based system (a structural grammar) adds structural elements to a zoned design, like columns, beams, plates (slabs) and shear walls. Each rule of the structural grammar associates a set of structural elements with a zone of particular dimensions. Depending on the dimensions of a zone, different structures can be generated to accommodate the space required by the zone. The shear walls and slabs are defined by their four corner points and columns are defined by their two endpoints. For now, the zoned design that contains the least number of zones for which at least one rule can be matched for each zone is selected by default for further processing.

The resulting structure may not be stable because the structural grammar treats every zone in isolation. However, in order to be able to determine stresses and strains using the finite element method, the structure is required to be kinematically determined, i.e. fixed in space. The method used in step *1d* to determine the kinematical mechanisms in the structure is supported by a recently developed method presented in (Hofmeyer and Russel 2009), which uses the null space of the structure stiffness matrix to predict these kinematic mechanisms. This work was inspired by (Hofmeyer and Bakker 2008) that demonstrates that when a structure stiffness matrix is singular, one or more mechanisms exists. The locations of these mechanisms can subsequently be computed by finding the null space of the stiffness matrix.

During the development of the null space procedures, it was realised that the structure generated from a zoned design is not necessarily connected, i.e. adjacent structural elements do not share endpoints. As a result, these neighbouring structural elements are able to move independently from each other. When implemented in the finite element model, though, they are expected to be connected.

For this reason, two splitting algorithms have been developed, which are described in (Hofmeyer et.al 2010), one of which is implemented as step *1c* in the spatial to structural transformation procedure. Using logic and partly brute force calculations, all structural elements are checked for intersections and coincident points. Where relevant, areas and lines are then split and redefined such that for the resulting structure all adjacent elements are correctly connected.

When the automatic stabilising procedures have been incorporated, to get to the finite element model (the end result of the first transformation procedure) loads have to be added and the structural system should be provided with a finer mesh than used during the analyses of free mechanisms.

3.2 Visualisation

The intent for the visualisation part of the research engine is to visualise the different designs as interactive 3D models in a way that is clear and comprehensible at a first glance. Because the interesting parts of spatial and structural designs are typically also situated on the inside of the design, the main goal of the visualisation tool is to make the innards visible from the outside without having to navigate the design in first person view. This goal has been realised by rendering translucent surfaces with opaque outlines. The translucency makes it possible to look through elements and even infer relative depth order between different elements while the outlines make it easier to identify the individual elements.

The current spatial designs that are accepted and produced by the research engine consist of axis aligned elements, which can be straightforwardly rendered using polygons. Although there are numerous scientific visualisation toolkits available, it was decided to implement a custom made render engine using the OpenGL graphics library (Shreiner and Group 2009). Considering that many of the available visualisation toolkits have an overkill in features and are needlessly complex compared to the relatively limited requirements for the visualisations of design instances, it was believed that the time needed to implement a custom render engine would not exceed the time needed to research and select an appropriate visualisation toolkit and to subsequently study its application programming interface (API). Note that an API defines the data types and functions that are available to the calling application.

As mentioned in the previous paragraph, OpenGL is used to render the spatial and structural designs. OpenGL defines a standardised API for a cross-platform 2D/3D graphics library for producing raster graphics. The standard defines the functions and data types an OpenGL library must provide to the calling application. OpenGL implementations exist for nearly all operating systems and most graphics hardware can accelerate OpenGL rendering tasks. The features provided by the OpenGL API are sufficient to satisfy the visualisation requirements for this research project.

Although it is possible in OpenGL to render translucent polygons through the notion of blending, care has to be taken in the order in which the translucent polygons are rendered. Because a translucent object ’filters’ the colours of the objects behind it, it is important for the correctness of the image that objects behind a translucent polygon are rendered first. A simple solution is to render the opaque polygons first and then render the translucent polygons in back to front order. However, for an arbitrary set of convex polygons, finding such a back to front order may not be as trivial as sorting a set of polygons on the distance to the viewpoint. Sets of polygons exist that do not have an explicit depth order. One such an example is depicted in Figure 3 (a), where three triangles have a cyclic overlap when viewed from an arbitrary direction. However, by splitting the white triangle over the dotted line as depicted in Figure 3 (b), a depth order can be imparted on the triangles and triangle fragments as indicated by the numbers in the figure. Another more trivial case is the situation where two or more polygons intersect; no back to front order can be established among the intersecting polygons.

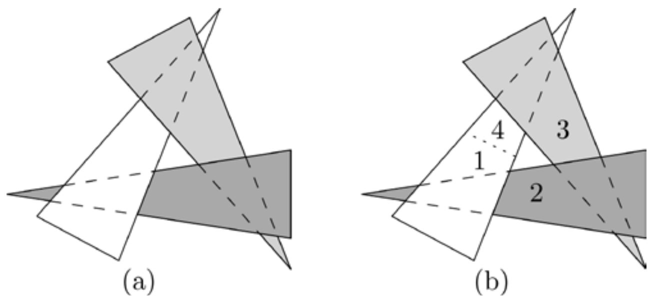

Figure 3: (a) Arrangement of triangles with no depth order. (b) By splitting the white triangle a depth order relation can be established

As a solution to the depth order problem, a render engine has been developed that builds a binary space partition tree (BSP tree for short) to store the polygons. Using the BSP tree, the renderer is able to efficiently compute a back to front (or front to back) order of the polygons from an arbitrary view point. Another benefit of the BSP tree variant used by the render engine is that during the construction of the tree, difficult polygon arrangements are dealt with in an elegant matter.

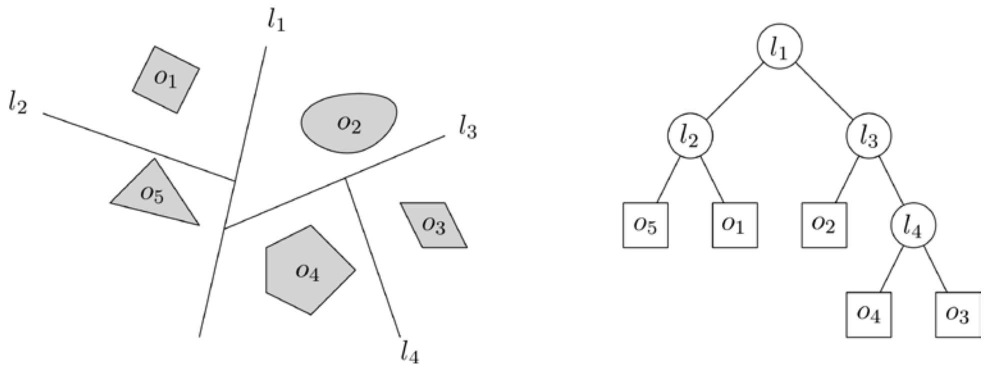

Figure 4: An example of a 2D BSP tree storing a set of objects, the left image shows the space partition, the right image shows the tree structure

To get an impression of a BSP tree, consider Figure 4. The left image shows a set of five 2-dimensional objects. The space and object set is partitioned by the four lines l_1, l_2, l_3, l_4. The right image shows the subsequent structure of the BSP tree. For a 3-dimensional hyperplane $h : ax + by + cz + d = 0$, let h^+ be the open positive half-space bounded by h and h^- be the open negative half-space (open means not including its bound). A BSP tree for a set S of objects in 3-dimensional space is defined as a binary tree T with the following properties:

- If $|S| \leq 1$ then T is a leaf v storing S. The possibly empty set of objects stored at node v is denoted $S(v)$.
- If $|S| > 1$ then the root v of T stores a hyperplane h_v, together with the set $S(v)$ of objects that are fully contained in h_v. The left child of v is the root of a BSP tree T^- for the set $S^- := \{h_v^- \cap s \mid s \in S\}$ and the right child of v stores the root of a BSP tree T^+ for the set of objects $S^+ := \{h_v^+ \cap s \mid s \in S\}$.

Note that $X := \{f(x) \mid P(x)\}$ denotes the set of elements $f(x)$ for x satisfying $P(x)$, where $f : \alpha \rightarrow \beta$ is a function, $P : \alpha \rightarrow \boldsymbol{B}$ is a Boolean predicate and $X \subseteq \beta$. The size of the resulting BSP tree depends on the number of polygon fragments that are generated during its construction. Every leaf node contains at most one polygon fragment, the internal nodes store the fragment s that was used to create the hyper plane $h(s)$ and all other fragments in S that are fully contained in $h(s)$. Therefore the size of the BSP tree is linear in the number of fragments generated by the splitting planes. The randomised 3-dimensional BSP construction algorithm using auto partitions is expected to generate $O(n^2)$ fragments for an input consisting of n triangles (De Berg et.al 2008). Although better BSP construction algorithms exist, this randomised auto partition variant is the easiest to implement and in practice performs quite well.

Using the BSP tree, a render engine can determine a back to front order to correctly render the translucent polygon fragments as follows. Given a tree T with root v and viewpoint p_{view}, when v is a leaf node then simply render the polygon fragment in $S(v)$. Otherwise, if $p_{view} \in h_v^+$ as shown in Figure 5, then no polygon fragment in T^- will obscure a polygon fragment in T^+. The render engine first draws the polygon fragments in T^-, followed by the polygon fragments in $S(v)$ and finally the polygon fragments in T^+. When p_{view} is contained in h_v, the order in which the sub trees of T are rendered is not important. See Algorithm 1 for the pseudo code of RenderBackToFront.

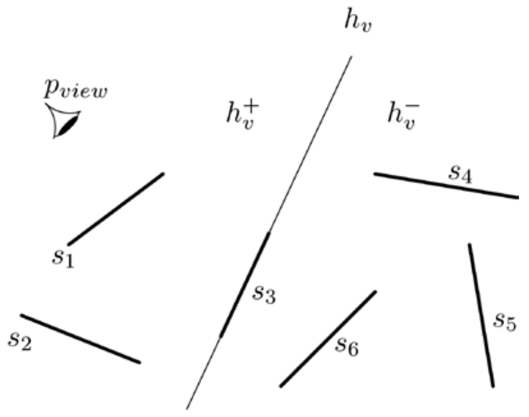

Figure 5: Diagram illustrating the depth order algorithm, using line segments

Algorithm 1. **RenderBackToFront(*T*, *p*$_{view}$)**

```
1   Let $v$ be the root node of $T$ ;
2   if $v$ is a leaf node then
3       Render the polygon fragments in $S(v)$ ;
4   else
5       if $p_{view} \in h_v^+$ then
6           RenderBackToFront($T^-$, $p_{view}$) ;
7           Render the polygon fragments in $S(v)$ ;
8           RenderBackToFront($T^+$, $p_{view}$) ;
9       else if $p_{view} \in h_v^-$ then
10          RenderBackToFront($T^+$, $p_{view}$) ;
11          Render the polygon fragments in $S(v)$ ;
12          RenderBackToFront($T^-$, $p_{view}$) ;
13      else
14          RenderBackToFront($T^+$, $p_{view}$) ;
15          RenderBackToFront($T^-$, $p_{view}$) ;
```

As mentioned before, the BSP construction algorithm deals with cyclic overlapping and intersecting polygons in an elegant manner. In fact the construction algorithm does not deal with them explicitly, these situations are resolved as a consequence of the definition of the BSP tree. Let T be a BSP tree with root node v, then by the definition of a BSP tree, v is either a leaf node or an internal node. When v is a leaf node, then $S(v)$ stores at most one polygon fragment, for which a depth order can be established from any point of view. Otherwise, if v is an internal node storing hyper plane h_v, a list of fragments $S(v)$ that are fully contained in h_v and sub trees T^+ and T^-, then the hyper plane h_v splits space in two half spaces h_v^+ and h_v^-. All polygon fragments that are stored in T^+ are fully contained in h_v^+ and all polygon fragments in T^- are fully contained in h_v^-. Furthermore, assuming that a depth order exists for the polygon fragments in T^+ and depth order exists for the polygons in T^-, a depth order can be established for all polygon fragments stored in the tree T, by following the three cases in Algorithm 1.

4 OUTLOOK

In order to realise the three goals of this research project, the research engine will be developed further. The first transformation step will be finalised by adding the finite element analysis procedure, which will involve several aspects. First of all, the stabilised structure has to be loaded with gravitational, wind and possibly other forces. Subsequently, a finite element mesh should be generated for the structure and solved to predict strains and stresses within the structural elements.

The next transformation step transforms structural design 2*n*-1 into structural design 2*n*, optimising one or multiple structural properties. One approach for such an optimisation would be to use the finite element results to detect structural elements having small strain energy and to remove these structurally redundant elements.

The third transformation step takes the optimised structural design 2*n* and produces spatial design 2*n*. This might prove to be a difficult transformation when using nothing more than the structural design, since the structural design only describes the geometry of the structural elements. Information on the utilisation of the space between the structural elements has been lost during the process. A first strategy could be to compute the convex hull of the vertices of the structural elements, however the resulting spatial design would be a single volume shaped as a convex polyhedron, which is likely to contain much more space than the original - not necessarily convex - spatial design.

The fourth transformation step takes spatial design 2*n* and transforms into spatial design 2*n*-1 *(*with *n* increased by 1). The intent of this transformation step is to change the design such that it resembles the original spatial design more closely, by restoring features that were lost during the previous transformations. One strategy could restore the number of rooms in the new spatial design or scale the spatial design to restore the original volume. An interesting transformation would be to restore the original topology of the spatial design. The topology can be represented as a graph whose nodes correspond to the rooms in the spatial design, the edges in the graph connect nodes whose corresponding rooms are adjacent in the spatial design. Given the topology of the original (or previous) spatial design and the topology graph of the current spatial design, graph operations can decide whether they are equal or relate in some other way. Using the topology graphs, it might be possible to restore the original topology without having to resort to complicated 3D geometrical representations of the spatial designs.

Parallel to the development of the research engine, cases studies will be performed on actual design processes, typically those of buildings. The design processes of at least a small residential building, a medium sized multi-story building and a large industrial building will be studied. The focus of these case studies will be on the interaction between the structural and spatial design processes, e.g. to find out which design decisions have had an impact for the other discipline involved and why these decisions have been taken. The results of these case studies, defined conform the data and process models as used in the research engine, will be used to benchmark the first version of the research engine. The research engine is supplied with spatial designs that resemble the studied cases in order to test whether the results produced by the engine are comparable to the actual design processes. Depending on the success of these tests, recommendations are made for improvements to the current transformations and of addition of other transformation strategies. A selection of the recommendations will subsequently be implemented.

Additionally, units of measurement will bee developed to quantify the properties of spatial and structural designs. As a result, current spatial-structural design processes can be understood and managed better, as the influence of different transformation strategies on the design instances can be predicted. For this, the next phase of the research project focuses on the relation between spatial and structural design itself. By performing experiments on specially crafted academic designs, fundamental knowledge about the relation between the structural and spatial designs can be developed (Hofmeyer 2007).

5 CONCLUSION

A research engine is under construction that simulates a cyclic design model where spatial and structural design processes are considered. This research engine will be used to investigate the interaction between spatial and structural design processes and to gain knowledge in the fundamental relationship between spatial and structural designs themselves.

A significant part of the research engine's first transformation procedure together with a first version of a visualisation tool has been implemented. The visualisation tool renders opaque and translucent polygons, with or without polygon outlines. Translucent polygons are rendered in back to front order for the purpose of image correctness. From any point of view, the render engine can establish a back to front order quickly, by using a binary space partition data structure that stores the polygons.

Although case studies, examples and implementations will be developed within the architectural domain, the developed research engine and principles of spatial-structural design interaction are equally useful for the domains of mechanical engineering, industrial design, automotive engineering, etc. Also, within the architectural domain, sub domains like building physics and construction technology could be integrated with the spatial design similar to the structural design in the research.

6 REFERENCES

Björk B.-C. 1992. A conceptual model of spaces, space boundaries and enclosing structures. *Automation in Construction* 1:193 – 214.

Bletzinger, K.-U. Ramm, E. 2001. Structural optimization and form finding of light weight structures, *Computers & Structures* 79:2053 – 2062.

Bletzinger, K.-U. Wchner, R. Daoud, F. Camprub, N. 2005. Computational methods for form finding and optimization of shells and membranes, *Computer Methods in Applied Mechanics and Engineering* 194:3438 – 3452

Borkowski, A. Fleischmann, N. Bletzinger, K. U. 1991. Supporting conceptual decisions in structural design, *International Conference on the Application of Artificial Intelligence Techniques to Civil and Structural Engineering*, 87–96.

De Berg, M. Cheong, O. van Kreveld, M. Overmars, M. 2008. Computational Geometry: Algorithms and Applications, Springer-Verlag TELOS, Santa Clara, CA, USA.

Eastman, C. Jeng, T. S. 1999. A database supporting evolutionary product model development design, *Automation in Construction* 8:305–323.

Fenves, S. J. Rivard, H. Gomez, N. 2000. Seed-config: a tool for conceptual structural design in a collaborative building design environment, *Artificial Intelligence in Engineering* 14:233 – 247.

Fuyama, H. Law, K. H. Krawinkler, H. 1997. Computer assisted conceptual structural design of steel buildings, *Computing in Civil and Building Engineering*, 969–976.

Harty, N. Danaher, M. 1997. Evaluating preliminary structural designs in an expert system, *Computers & Structures* 63:1243 – 1249.

Haymaker, J. Fischer, M. Kunz, J. Suter, B. 2004. Engineering test cases to motivate the formalization of an aec project model as a directed acyclic graph of views and dependencies.

Hofmeyer, H. 2007. Cyclic application of transformations using scales for spatially or structurally determined design, *Automation in Construction* 16:664 – 673.

Hofmeyer, H. Bakker, M. 2008. Spatial to kinematically determined structural transformations, *Advanced Engineering Informatics* 22:393 – 409.

Hofmeyer, H. Kerstens, J.G.M. 2006. Full 3d structural zoning of space using a geometrically related reducer and matrix coupling, *CAADRIA06, Proceedings of the 11th Conference on Computer Aided Architectural Design Research in Asia*, 161–168.

Hofmeyer, H. Russell, P. 2009. Interaction between spatial and structural building design: a finite element based program for the analysis of kinematically indeterminable structural topologies, *Proceedings of the 9th International Conference on Construction Applications of Virtual Reality*, 247–256.

Hofmeyer, H. Van Roosmalen, M. Gelbal, F. 2010. Pre-processing parallel and orthogonally positioned structural design elements to be used within the finite element method, accepted for publication in *Advanced Engineering Informatics*.

Keatruangkamala, K. Nilkaew, P. 2006. Strong valid inequality constraints for architectural layout design optimization, *CAADRIA06 Proceedings of the 11th Conference on Computer Aided Architectural Design Research in Asia*, 179–185.

Khemlani , L. Timerman, A. Benne, B. Kalay, Y. 1998. Intelligent representation for computer aided building design, *Automation in Construction* 8:49–71.

Kicinger, R. Arciszewski, T. De Jong, K. 2004. Morphogenic evolutionary design: cellular automata representations in topological structural design, *Adaptive Computing in Design and Manufacture VI*, 25–38.

Kotsopoulos, S. 2006. Constructing design concepts, a computational approach to the synthesis of architectural form, Ph.D. thesis, Massachusetts Institute of Technology.

Maher M.L. 1985. Hi-rise and beyond: directions for expert systems in design, *Computer-aided design* 17:420–427.

Markov, I.J. Gabriel, J.F. 2001. Spatial and structural aspects of polyhedra, *Engineering Structures* 23:4–11.

Martini, R. Powell, G. H. 1990. Geometric modeling requirements for structural design. *Engineering with Computers* 6:93–102.

Matthews, K. Duff, S. Corner, D. 1998. A model for integrated spatial and structural design of buildings, *CAADRIA98: Proceedings of The Third Conference on Computer Aided Architectural Design Research in Asia* 123–132.

Mullins, M. Kiregaard, P. Jessen, R. Klitgaard, J. 2005. A topology optimization approach to learning in architectural design, *Proceedings eCAADe 23 Digital Design: The Quest for New Paradigms*, 155–162.

Nguyen, T. Ha, H. Bédard, C. 1996. Architectural and structural design with code compliance checking, *Third Design & Decision Support Systems in Architecture & Urban Planning Conference* 357– 364.

Oxman, R. 1997. Design by re-representation: a model of visual reasoning in design, *Design Studies* 18:329–347

Rafiq, M. Y. MacLeod, I. A. 1988. Automatic structural component definition from a spatial geometry model, *Engineering Structures* 10:37 – 40.

Reffat, R. 2006. A computational system for enriching discovery in architectural design, *CAADRIA06 Proceedings of the 11th Conference on Computer Aided Architectural Design Research in Asia*, 169–177.

Rivard, H. Fenves, S. J. 2000. A representation for conceptual design of buildings, *Journal of Computing in Civil Engineering* 14:151–159.

Sacks. R., Warszawski. A., 1997. A project model for an automated building system: design and planning phases, *Automation in Construction* 7:21–34.

Sause, R. Martini, K. Powell, G. H. 1992. Object-oriented approaches for integrated engineering systems, *Journal of Computing in Civil Engineering* 6:248–265.

Sause, R. Powel, G. 1991. A design process model for computer integrated structural engineering: Design phases and tasks, *Engineering with Computers* 7:145–160.

Scherer, R. Gehre, A. 2000. An approach to a knowledge-based design assistant system for conceptual structural system design, *Product and Process Modelling in Building and Construction*.

Shreiner, D. Group, T. K. O. A. W. 2009. OpenGL Programming Guide: The Official Guide to Learning OpenGL, Versions 3.0 and 3.1, Addison-Wesley Professional.

Steirteghem, J. V. Wilde, W. P. D. Samyn, P. Verbeeck, B. P. Wattel, F. 2005. Optimum design of stayed columns with split-up cross arm, *Advances in Engineering Software* 36:614 – 625

Weise, M. Katranuschkov, P. Scherer, R. 2000. A proposed extension of the ifc project model for structural systems. *Product and Process Modelling in Building and Construction* 229–237.

Yulin, M. Xiaoming, W. 2004. A level set method for structural topology optimization and its applications, *Advances in Engineering Software* 35:415 – 441.