Multiplying Architectural Layouts and 3D Forms: Interplay of Necessity and Contingency in Architectural Modeling

Hao Hua

CAAD/ ITA/D-ARCH, ETH Zurich, Switzerland
hua@arch.ethz.ch

Abstract. Since 1960s there were many models for architectural layout planning which formulated design activities as problem-solving. On the other hand, various form-finding models had emerged after 1980s. The former seeks the necessity of architectural modeling as an objective science, while the latter encourages the contingent characters of individual modeling. This paper proposes a method of integrating the two families of models. A commutation channel is defined thus every member in one family can work with any member in the other. Therefore the models of architectural layouts can "multiple" the models of 3D forms, which leads to rich variety of architectural structures and forms. The method is implemented and tested in Java.

Keywords: layout, form, necessity, contingency.

1 Introduction

The history of Computer-Aided Architectural Design (CAAD) commenced with the computational approaches to spatial planning. It was highly influenced by the optimization methodology in the field of industrial engineering [1]. A typical problem was how to arrange various functional units in the floor plan to minimize the costs or to maximize the merits. In 1960s and 1970s, the forerunners in the CAAD field concentrated on symbolic formulation of architectural layout planning as well as the possible implementations with computer [2-4]. It is significant the nature of design was defined as problem-solving, "a process of searching through alternative states of the representation in order to discover a state that meets certain specified criteria" [4]. Such contemporary formulation of design problem serves a computational revival of the architectural functionalism originated by J.N.L. Durand in the early nineteenth century. To some degree, the problem-solving approaches to spatial layout planning achieved the goal of rationalizing and systemizing the activities of architectural design as a science.

A paradigm shift took place in the field of CAAD after 1980s. In natural science and computer science, a bundle of new concepts and technologies had emerged, just to name a few: self-organization, evolutionary computation, artificial neural networks. Later Frazer's "An Evolutionary Architecture" [5] reflected this conceptual shift in architecture, followed by a burst of methodologies of computational design such as evolutionary design [6], biomimetics and morphogenesis [7]. In a general sense, computational design shifted from the machine paradigm (deterministic models) to the biology paradigm (probabilistic models). While in a narrow sense, the research interests moved from the "dull" 2D layout planning to the logic of making novel 3D forms. An important character of these forms is "continuous differentiation" [8], i.e. the configurations of the enormous components are sensitive to their local environments. With the help of modeling software like Rhino and scripting environment like Processing, free form-finding has become practical for most designers. The contemporary form-finding instead of the spatial planning became the first (probably the only) computational design method which is practical and productive in architectural design.

This paper interprets the difference between architectural layout planning and novel form-finding in terms of modeling. Modeling represents certain objects or phenomena by mathematics or other formal languages. On the one hand, there is necessity in modeling [9], i.e., the models should be "true" or precise enough (the models fit the observations of the reality). On the other hand, there must be contingency in modeling [9], since there are always more than one model suitable for the same target. The choice on models depends on modeler's individual perspective and on the formal languages (e.g., a circle can be modeled in a 2D Cartesian system or in a polar coordinate system). We can observe that the computational approaches to architectural layouts before 1980s emphasized the necessity of modeling but unfortunately overlooked the contingency of modeling. In other words, they quested a universally valid architecture without any desire for novel solutions to particular problems or for exotic forms. By contrast, the current form-finding approaches always seek new models for generating novel topologies and shapes. They often lead to novel articulations without any functions defined explicitly, i.e., they prefer the contingency to the necessity of modeling. The "scientific" planning would find form-finding unnecessary while the latter would regard the former valueless.

However, the "objective" layout planning methods actually do not conflict with the "subjective" form-finding approaches. For example, self-organization of architectural components can produce both valid architectural layouts and unpredictable structures/forms [10]. This paper resolves the conflicts between the two approaches by connecting two types of models: the models of architectural layouts planning and the models of 3D forms. Usually, it brings confusions and difficulties to the modeling process if there are two incompatible models must be employed. However, the differences or the conflicts between the models actually offer great opportunities for creating new models. Such unusual modeling process takes account of both the necessity (making the multiple available models work reasonably) and the contingency (creating new ways of orchestrating multiple models) of modeling. Theoretically there would be infinite ways in which two symbolic models can interplay with each other. This paper proposes a particular method of integrating the two kinds of models, thus it only serves as one case of infinite solutions to "gluing" two heterogeneous models.

2 Method

2.1 Overview

The project works with "families" of models instead of individual models directly. One family is for the models of layout planning, the other for 3D forms. The models in one family do not need to bear similarities in the structures. Besides, no shared representation between the two families is required, which is opposite to many collaborative design methods [11]. The key is a valid communication of information from the two families of models. To some degree, the nature of communications rather than the models defines the family of models. All models at one side of "communication channel" constitute one family, the ones at the other side make the other family. Therefore each member from one family can work with any member from the other family. Hence the models of architectural layouts can "multiple" the models of 3D forms, which leads to rich variety of architectural structures and forms. If there are n models in one family and m models in the other, then there are n×m composite models.

Technically, the family of layout planning models is defined as an abstract class in Java. The particular models extend the abstract class. The same for the other family. The java program takes one particular model for layout planning and one particular model for 3D forms as two parameters, and then makes syntheses of architectural solutions. The algorithm makes the articulations from the two heterogeneous models coherent, rather than carries out any optimization procedures. More precisely, the algorithm contains three main steps:

1. Generate a particular 3D form via a particular model (one model can produce infinite number of forms).
2. Construct floor plan layouts via a particular model under the constraints of the forms generated in step1 (this procedure may fail if the conflicts between the two models can not be solved).
3. If step 2 succeeds, a valid solution is produced. If it fails, go back to step 1 (restart with a new particular form).

The two kinds of models are not strictly coupled with each other under most circumstances. The composition of floor plan is still open when the overall 3D form is fixed. The other way around, there are many variations of 3D form after the floor plan is made. In the project a procedure of communication is defined as follows: first, the model of 3D form defines two types of volumes: the positive (to be occupied by buildings if possible) and the negative (not to be occupied if possible). Second, the whole volume produces a set of 2D planes by making horizontal slices out of the volume. Every plane saves the information of the positive regions and the negative regions. The former are supposed to be occupied by rooms, the latter are only allowed to be occupied by circulations (corridor, stair case and elevator) if necessary. Thus the model of floor plan layouts can operate on these 2D planes. It is important that the 2D layout planning can also influence the 3D forms, e.g., some positive volumes will be eliminated if no rooms can occupy them properly. Thus, the two families of models have interactive communications.

2.2 The Models of 3D Forms

Any model generating 3D geometry can be transformed into a member of the family of “3D form”. The project builds two particular models: the “Cubes” and the “Perlin”. The “Perlin” model creates a series of “iso-surfaces”. One iso-surface divides the whole volume into two parts; two iso-surfaces three parts (the number iso-surfaces is equal to the number of disjunct volumns plus one). A configuration of two iso-surfaces is adopted in the experiment. First, the volume is voxelized. Perlin’s noise function [12] maps the position of every voxel to a “density” value. The “iso-surfaces” (Fig. 1 left) are constructed where the “density” value is equal to the predefined iso values. One of the three disjunct volumes is marked “negative”.

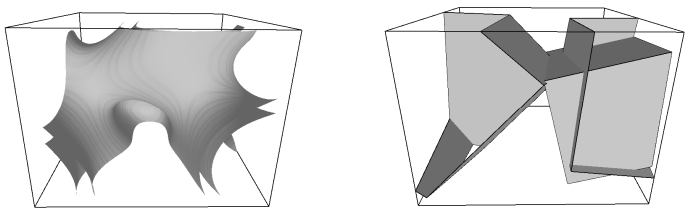


Fig. 1. The 3D geometry generated by the “Perlin” model (left); the 3D geometry generated by the “Cubes” model (right)

The 2D planes are created by “cutting” the whole volume at the level of floors. The planes inherited the information of positive/negative from the labeled volumes. The light gray areas in (Fig. 2) denote the negative parts. Besides, the virtual floors at the level of 1m and 2m higher than each floor are also calculated. The dark gray areas denote the additional negative areas from the virtual floors (they are essential for making reasonable floor plans in later stage).

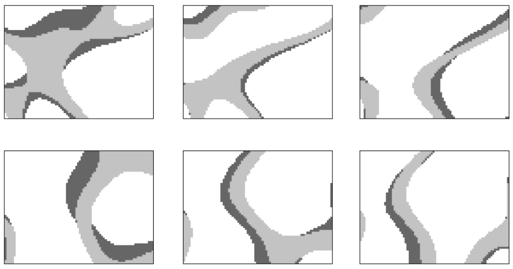


Fig. 2. 2D planes cut from 3D “Perlin” model

The model of “Cubes” first generates three cubes, then the cubes are cut by the cuboid (Fig. 1 right). The whole volume of the cuboid is divided into voxels, the sign of each voxel is calculated by:

$$s = (-1)^n \tag{1}$$

Symbol n denotes the number of cubes (excluding the cuboid) which contain the target voxel. According to the formula, the volume outside all cubes is positive. If the volume is inside odd number of cubes it's negative, otherwise positive.

2.3 The Models of Architectural Layouts

The models of architectural layouts manipulate the 2D planes made from the model of 3D forms to produce reasonable floor plans. The positive areas are for rooms and circulation (corridor, stair case and elevator), while the negative areas are only for circulation if necessary. The "Central Corridor" model constructs a corridor along the long side of the floor. The staircase and the corridor must be connected to the corridor. Other spaces are further divided into small rectangular rooms which are all connected to the central corridor. (Fig. 3) illustrates a result under the condition that all the areas are positive.

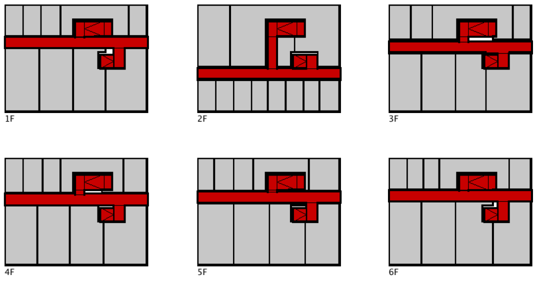


Fig. 3. The layout generated by the "Central Corridor" model

However, the situations become more complicated when there are negative areas on the floors. To be precisely, there are four "invalid" situations: a room is too small; a room is too narrow; a room is not connected to the central corridor; the shape of a room is invalid for the opening of a door. Some rooms (Fig. 4) are not accessible to the central corridor. In order to avoid these situations, an algorithm merges every invalid room with its neighbors. If the floor plan is still not valid after that, some invalid rooms are eliminated and some are merged as part of the corridor.

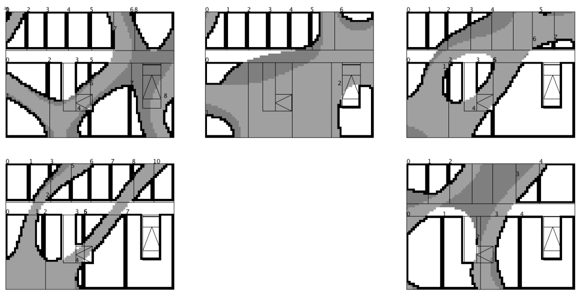
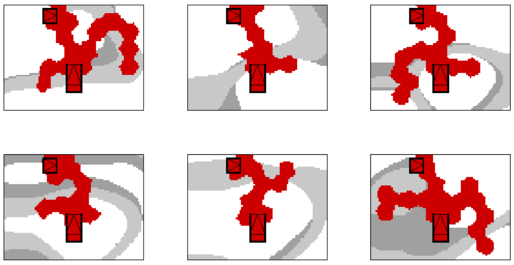
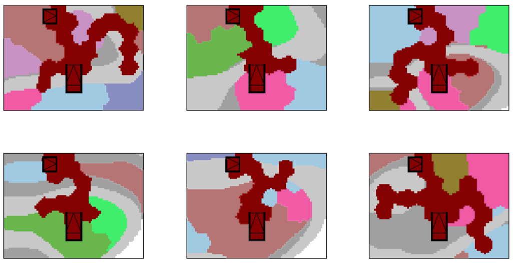


Fig. 4. Analysis of the layouts (some rooms are inaccessible)

The “Voronoi” model arranges floor plans based on the Voronoi tessellation of the floor area. It first generates the corridor with three goals: first, the corridor should connect the staircase and the elevator; second, it should reach all the regions isolated by negative areas (Fig. 5 (a)); third, the area of the corridor should not be very large. The second step is creating rooms, it actually subdivides the positive areas of the floor by grouping the cells in the Voronoi tessellation (Fig. 5 (b)). Each room should have a relatively compact shape.



(a)



(b)

Fig. 5. Corridord generated by the “Voronoi” model (a); creating rooms (b)

3 Results

The models are constructed and tested in Java. The results (the 3D forms and the floor plans) below are directly produced by the java program (the 3D forms are rendered in Maxwell) without tuning by hand. The results imply that each model from the family of 3D forms works well with each model from the family of architectural layouts. Different pairs of the models make different articulations. Though there are only two members in each family in this project, it is possible to add more members into each family according to the “communication channel” between the two families.

The first results come from the model of architectural layouts working with none of 3D forms. It suggests that a single model from one family is able to function alone. While a combination between two families yields more variety. There are four different situations: Central Corridor-Cubes (Fig. 6), Voronoi-Cubes (Fig. 7), Central Corridor-Perlin (Fig. 8), Voronoi-Perlin (Fig. 9). Only one instance from each situation is shown here, however, there are actually infinite numbers of results in each situation.

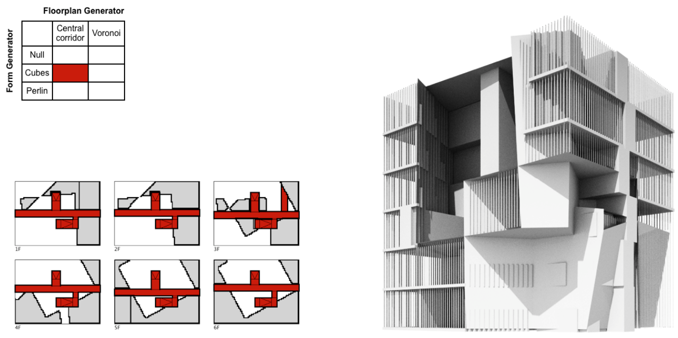


Fig. 6. The layouts and the forms generated by Central Corridor - Cubes

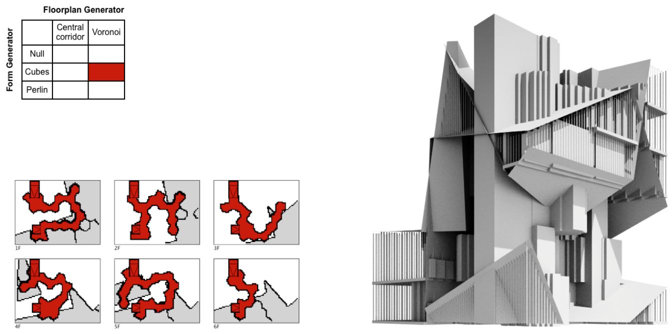


Fig. 7. The layouts and the forms generated by Voronoi - Cubes

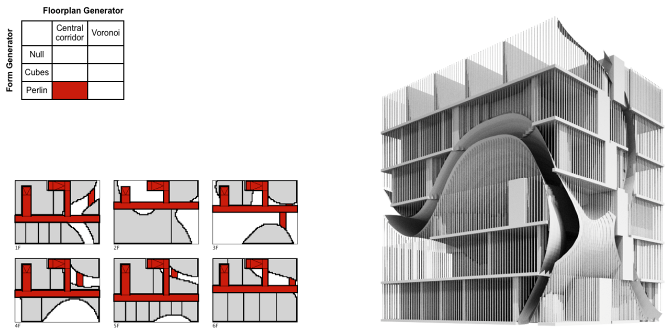


Fig. 8. The layouts and the forms generated by Central Corridor – Perlin

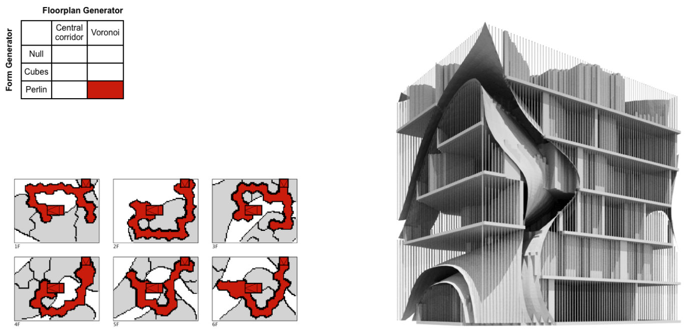


Fig. 9. The layouts and the forms generated by Voronoi – Perlin

The results imply that the interplay of the two families of models results in a wide range of structures and shapes. Each pair of models constitute a parametric model. The final articulations are closely associated with both the model of 3D forms and that of 2D layouts. Therefore the results can be read in two ways (corresponding to the two models) simultaneously. Though there are ambiguities in the final forms, the two underlying generators are more or less transparent. To summarize, the results suggest that employing heterogeneous models in one generative process is valid and productive.

4 Conclusions

Half century after the birth of architectural modeling, today we already confront with abundant/redundant models in architectural design. Some researchers strove to make generalization over particular models (like BIM approach). However, more general the model is, the more difficulties in making sense of the models in architectural design. This research argues for a flexible method for employing available models: coupling heterogeneous models without generalization or shared representations. On the one hand, it's a solution to the inconsistency between the models which have to be evolved in a design task. Up to this point, it's subject to the correspondence between the models and the reality. On the other hand, it is a design strategy since it builds new models by orchestrating available models. Hence it is open for the designer's inventions in design concepts during the design process.

References

1. Koopmans, T.C., Beckmann, M.: Assignment problems and the location of economic activities. Econometrica 25(1), 52–76 (1957)
2. Whitehead, B., MEldars, M.Z.: The planning of single-storey layouts. Building Science 1(2), 105–165 (1965)

3. Seehof, J.M.: Automated facilities layout programs. ACM National Meetting, 191–199 (1966)
4. Mitchell, W.J.: Computer-Aided Architectural Design. John Wiley & Sons, Inc., NY (1977)
5. Frazer, J.: An Evolutionary Architecture. John Frazer and Architectural Association, London (1995)
6. Bentley, P.: An Introduction to Evolutionary Design by computers. In: Bentley, P. (ed.) Evolutionary Design by Computers, pp. 1–73. Morgan Kaufmann, California (1999)
7. Hensel, M., Menges, A., Weinstock, M. (eds.): Techniques and Technologies in Morphogenetic Design. AD, vol. 76(2). John Wiley&Sons, Ltd. (2006)
8. Schumacher, P.: Parametricism as Style - Parametricist Manifesto. In: 11th Architecture Biennale, Venice (2008), `http://www.patrikschumacher.com/Texts/Parametricism%20as%20Style.htm` (January 29, 2013)
9. Vuillemin, J.: Necessity or contingency: the master argument. CSLI publications, US (1996)
10. Hovestadt, L.: Beyond the Grid - Architecture and information technology. Birkhäuser, Germany (2010)
11. Saad, M., Maher, M.L.: A computational model for synchronous collaborative design. AAAI Technical Report WS-93-07. pp.191- 206 (1993)
12. Perlin, K.: Improving noise. ACM Transactions on Graphics (TOG) 21(3), 681–682 (2002); Proceedings of ACM Siggraph 2002