Blucher Design Proceedings
Dezembro de 2014, Volume 1, Número 7
www.proceedings.blucher.com.br/evento/sigradi2013

Pushing the Envelope: Stretching the Limits of Generative Design

António Leitão
INESC-ID/Instituto Superior Técnico, Portugal
antonio.menezes.leitao@ist.utl.pt

Luís Santos
University of California, Berkeley, USA
luis_sds82@berkeley.edu

Rita Fernandes
Instituto Superior Técnico, Portugal
rita.serra.fernandes@gmail.com

Abstract

Design is characterized by change. Nowadays, addressing change in design requires enormous efforts, particularly, when the designer is exploring different solutions or when it is necessary to adapt the design to evolving constraints. This paper discusses the potential of Generative Design to help designers handling change, namely in the generation of several alternative design solutions. We propose a programming-based approach that, although requiring an initial investment, dramatically reduces the efforts of design modification. We evaluate the approach in a case study and we show that it is cost-effective in the development of an architectural design.

Keywords: Design process; Change; Modeling; Generative design; Programming.

Introduction

Generative Design (GD) (McCormack, 2004) helps overcome these limitations. GD can be summarily described as form creation through algorithms (Terdizis, 2003). One of the main benefits of GD is the fast and effortless generation of a wide range of solutions, exploring different design approaches and implementing the continuously changing constraints and requirements which characterize the design process. Thus, it supports the architects in the exploration of a larger solution space as well as in the conceptualization of innovative solutions through the interaction between design constraints and parameters (Kilian, 2006).

Writing a GD program that describes an architectural model requires a large initial effort, and might be less cost-effective than the typical digital modeling approach. However, this initial cost can be quickly recovered when it becomes necessary to incorporate changes in the design. This happens because the design is unambiguously represented in a parameterized GD program and a significant number of changes only require different values for its parameters. These parameters might represent numerical values, geometric shapes, mathematical functions, or even subprograms. The important idea is that by changing the current values of the parameters or by including additional parameters, the designer can quickly generate a different model that expresses the changes in the design. Obviously, not all changes are trivial to implement and some might require additional efforts to adapt the GD program. Thus, the research question which can be posed is: how far can we go with this approach?

Case Study

In order to answer the previous question, we made an experiment: we formalized the design of a building in a GD program which supports as many changes as possible. We then tested the resilience of our formalization by simulating several realistic changes and measuring their impact, particularly, in the time and effort required for their implementation compared to the conventional use of CAD tools. Our case study was focused on a large and complex commercial building that is under construction: MVRDV's Market Hall (Boranyak, 2010). This building is characterized by an unconventional bent shape, which is closed by two glass facades. Because it constraints both the building elements and the interior spaces, the shape of the building was the starting point for our modeling approach. However, instead of capturing the exact geometry of the building, we captured the underlying ideas of the design, and formalized them in a corresponding GD program.

The modeling process we used can be divided into four phases: (1) formalization of the building shape, (2) modeling of building elements (walls, slabs, etc.), (3) openings and frames, and finally (4) detailed elements (posts, the elements of a glass facade, etc.). In the first phase, we defined the underlying geometry that captures the overall shape of the building. This geometry was then used as input for the remaining phases of the modeling process. The decomposition of the process into four phases, each one increasing the detail of the model, led to the existence of dependencies between elements. Their sequence ensures that the geometry produced in each stage fits in the one produced in previous phases, allowing automatic change propagation. This

Leitão, António; Fernandes, Rita; Santos, Luís; "Pushing the Envelope: Stretching the Limits of Generative Design", p. 235-238 . **In: Proceedings of the XVII Conference of the Iberoamerican Society of Digital Graphics: Knowledge-based Design** [=**Blucher Design Proceedings, v.1, n.7**]. São Paulo: Blucher, 2014.
ISSN 2318-6968, DOI 10.5151/despro-sigradi2013-0043

means that a change in, e.g., the building form, causes a correspondent change in the window openings, frames, shape of the walls and slabs, etc.

It is precisely these changes that are visible in Figure 1. The variant building (below) was generated by simply changing the building form. The remaining parameters were kept identical to those used to generate the Market Hall (top). The instanced model reveals a large number of differences in the elements of the building, including their geometry, position, direction, and number of sub-elements.

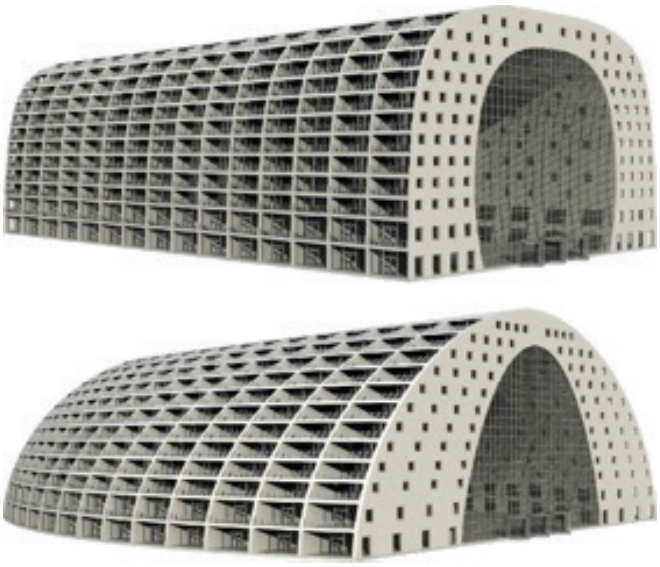

Figure 1: Top: MVRDV's Market Hall as generated from our GD program. Below: A variant building generated by changing the overall shape of the building.

Usually, changes in the overall shape of a building are very difficult to accommodate using conventional modeling approaches. However, using our GD-based approach, we implemented the change in minutes.

In the previous example we simulated just a refinement in the building shape, without changing its general dimensions. In the next example we simulate a more extensive change, not only in the shape, but also in the positions and dimensions of specific elements of the building. In Figure 2 we show two outcomes resulting from changing the parameters that control the form, the number and position of the slabs and walls, the number of windows, their position and dimensions, etc. The relationships between elements were kept and adapted to a different scale. Similarly to the first example, these results were produced by changing a small set of parameters, a task that was implemented in minutes. Compared to the previous example, these changes are even more difficult to handle in a conventional approach because it is not possible to reuse any element from one model to the next.

In the first two scenarios we modified the actual values of parameters but it is also important to consider changes that also require modifications in the implemented algorithms. For example, Figure 3 illustrates changes on the geometry of the fencing posts.

To implement this change we generalized the parts of the program that dealt with fencing posts so that, instead of always using a rectangular section (identical to those of the Market Hall, visible in the top), they now accept an additional parameter for a procedure that produces the desired shape. Figure 3 shows this feature by replacing the default rectangular section with a circular one.

Figure 2: Two buildings generated by introducing several changes in the shape and in the number and position of slabs, walls, windows, and other elements.

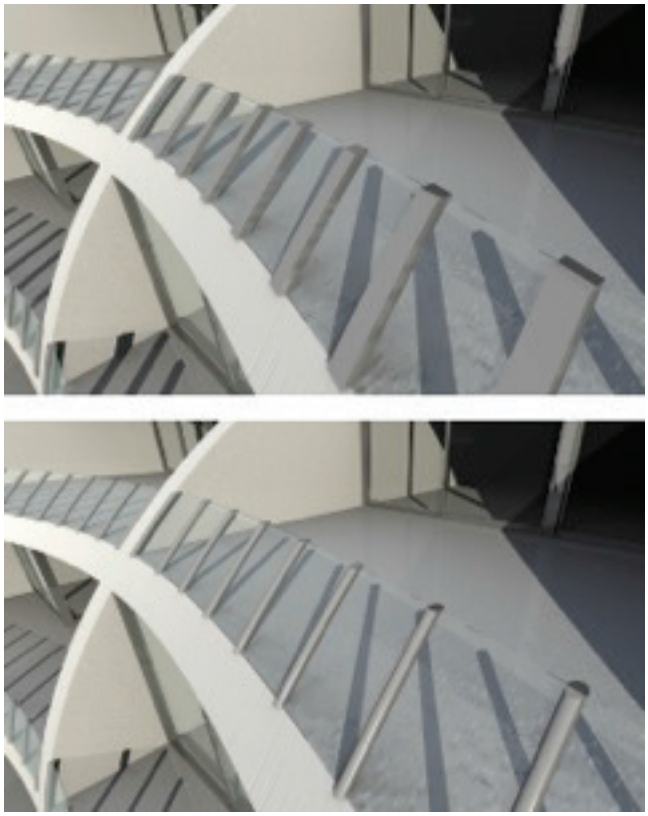

Figure 3: Top: Posts with rectangular sections. Below: Posts with circular sections.

Figure 4: A parallelepiped building generated by changing the formalization of the overall shape of the Market Hall.

The modifications we made to our GD program were completed in less than an hour, a very short amount of time for a task that, in a conventional approach, requires changing a large number of elements and, moreover, must be repeated each time we try a different alternative. On the contrary, in our approach, the generalization effort only needs to be done once, allowing the designer to effortlessly test many different alternatives.

Finally, we simulated a more drastic scenario: changing the building from a bent and longitudinal shape to a parallelepiped shape (Figure 4). In this case, we were forced to implement additional algorithms that produce the specific geometry required, and change some of the program parameters. The important point, however, is that all these changes were made in approximately half an hour. If an equivalent reuse is intended in a conventional approach, the designer might reuse some elements from other models, such as the frames of the lateral facades, but only when their overall dimensions are identical. This is such a severe restriction that, in practice, designers rebuild their models from scratch.

Evaluation

To validate the results of our experiment we did an inquiry, asking 10 experts in several CAD tools to determine the expected time to implement any of these changes using their conventional approaches.

In Table 1 we present the results of our inquiry. The analysis shows that the expected times of the conventional approach are, without exception, much longer than the actual times of our approach. As a result, the inquiry corroborated our belief that GD helps designers handling change.

Table 1: Results of the inquiry: comparison of the time needed to implement the changes between the conventional and the programming-based approach.

	Time (hours)	
Change	**Conventional approach**	**Programming-based approach**
Building overall shape (Figure 1)	70	< 1/6
Building overall shape, dimensions and positions of specific elements (Figure 2)	92	< 1/4
Shape of specific elements - fencing posts (Figure 3)	18	< 1
Formalization of the building overall shape (Figure 4)	42	< ½

During this experiment we noticed that when a change in the overall shape was required, all the participants in the inquiry would throw away their previous models and start the modeling

process from scratch. At most, they would just try to reuse and adjust specific elements with standard geometries, e.g., rectangular frames or revolving doors. This shows that, despite the underlying logic inherent to a specific design and the application of the same modeling process to change its features, the major modeling task would always be repeated. This causes a tremendous waste of time and effort that was already spent in previous modeling activities.

Although our proposed approach shows large gains in handling change, these gains depend on the development of a GD program that, obviously, requires time and effort. As a result, it is crucial to this research to also evaluate this effort. To compare the time required to develop this program with the conventional modeling task we made a second inquiry to the same participants, asking them about the time they would need to model the Market Hall building. As we predicted, the results, shown in Table 2, proved that the initial development cost using our approach is larger than using the conventional one. However, when we analyze the cumulative time to implement the initial model and handling the changes, we verify that the initial investment in the GD program was highly rewarded.

Table 2: Results of the inquiry: comparison of the time needed to produce the model / develop the GD program.

	Time (hours)	
	Conventional approach	**Programming-based approach**
Production of the model / program	131	160

Conclusions

We designed this experiment to determine how far we can go using a pure programming approach to architectural design. The results of the experiment allow us to conclude that it is possible to go very far indeed.

Given that this was an experiment, we decided to stop it as soon as we had collected enough information, but it was clear to us that we could go much further. Although more experiments are needed (e.g. implementing this programming-based approach in an architecture office during a real architectural design process), we believe that it is not only possible but actually cost-effective to develop designs with the aid of a pure programming approach.

Designs that can be formalized by explicit and implicit patterns and rules, such as the Market Hall, are prime candidates for applying this approach. Our experiment proved that a programming modeling method can be integrated in the design process right after the first crystallization of the architectural concept. This approach may be also useful during the design conceptualization phase, allowing the quick production of simple models that explore different conceptual alternatives.

Although a larger initial effort is needed to formalize a design, our experiment shows that this effort is quickly recovered. Our evaluation proved that this approach is sufficiently flexible, not only to accommodate expected changes, but also changes that were not planned. This conclusion can be extended to the architectural practice, where changes frequently arise without being anticipated.

An important lesson from this work is that the developed program should attempt to generalize the design as a mean to simplify the generation of different instances from an initial model. This is possible due to the creation of parameter and variable interdependencies that automatically propagate changes between building elements. The proposed approach is aligned with the requirements of the design process, by allowing an effortless introduction of changes and also the exploration of different design solutions, thus assisting the designers in decision-making activities. Our evaluation shows that a programming approach can also support large scale designs and can easily incorporate mass-customization strategies (Duarte, 2005), in which the effort required to produce one program is recovered by its use to model several related buildings.

Finally, we should mention an important advantage of the programming-based approach: any inconsistencies or errors in the design quickly show up as bugs in the program. This allows early discovery of problems that are more costly to solve using conventional approaches.

Acknowledgments

This work was partially supported by Portuguese national funds through FCT under contract Pest-OE/EEI/LA0021/2013, by the Rosetta project under contract PTDC/ATP-AQI/5224/2012, by the project PTDC/AUR-AQI/103434/2008 and by the PhD scholarship SFRH/BD/91939/2012.

References

Boranyak, S. (2010). Archetype. Civil Engineering, ASCE, 80(2), 76-79.

Duarte, J. P. (2005). Towards the mass customization of housing: the grammar of Siza's houses at Malagueira. Environment and planning B: Planning and Design, 32(3), 347-380.

Kalay, Y. (2004). Architecture's New Media: Principles, Theories, and Methods of Computer-Aided Design. Cambridge, Massachusetts: The MIT Press.

Kilian, A. (2006). Design innovation through constraint modeling. International Journal of Architectural Computing, 1(4), 87-105.

McCormack, J., Dorin, A., & Innocent, T. (2004). Generative design: a paradigm for design research. Proceedings of Futureground, Design Research Society, Melbourne.

Terdizis, K. (2003). Expressive Form: A Conceptual Approach to Computational Design. London and New York: Spon Press.