A Parametric Workflow: from *Grasshopper3D* to *Autodesk Inventor*

Eli MELTZER
School of Architecture, Newark Institute of Technology
Newark, NJ 07102, USA

ABSTRACT

This paper seeks to explore the parametric relationships guiding the design of seating arrangements in concert halls, specifically relating to ergonomics, sight lines, means of egress and placement of balconies, and their relation to the wider considerations of the Time Delay Gap and Reverberation Time. The exploration uses computational parametric tools, primarily the *Grasshopper* plug-in for *Rhino3D*, and *Autodesk Inventor*, to graphically model, display and test these parameters. Interoperability between these two software platforms is, therefore the primary area of study for this paper, as we seek to marry the open-ended design capabilities of *Grasshopper* with the robust production tools of the *Inventor* suite. Research focuses on the sharing of data through the *gHowl* add-on for *Grasshopper*, to export parameter values to *Excel*, which are applied to *iPart* factory tables in *Inventor*, to generate the various seat arrangement iterations. The optimization and specialization of both software platforms is a key goal of this research. Specifically, we are seeking to leverage the iterative and recursive parametric modeling capabilities of *Grasshopper*, to supplement the "piece-by-piece" logic of *Inventor*; while *Inventor* will be used primarily to produce BIM (Building Information Modeling), and presentation quality drawings.

Keywords: Parametric, Grasshopper, Inventor, Rhino, Fabrication, Workflow

1. CONTEXT

The integration of information based design tools is continuing to revolutionize the design disciplines, in particular, the fields of architecture and industrial design. Amongst the tools developed, primarily within the engineering industry, parametric software platforms and BIM are the most widespread and influential. The parametric refers to a design process "based not on fixed metric quantities but on consistent relationships between objects, allowing changes in a single element to propagate corresponding changes throughout the system" (Meredith, 4). BIM, on the other hand, refers to a data model capable of capturing "domain-specific information about entities … that is constructed around building entities and their relationships to one another [of which] geometry is only one of … these building entities" (Cheng, 491).

It is important to understand that traditionally the fields of architecture, engineering and construction each have their own method of organizing information and hence their own tools and softwares. With the introduction of information-based modeling, it is possible for the designer to include any aspect of the design or construction process into one virtual master model, which accounts for every aspect of each individual part of the building. Within the field of parametric tools themselves, there have developed two primary vehicles: algorithmic modelers, and constraint-based modelers. The algorithmic is defined as "a method of generation, producing complex forms and structures based on simple component rules," and often relies on the use of complex computer scripting tools (Meredith, *ibid*). A constraint-based modeler, on the other hand, relies primarily on the application (often visual) of "dimensions and constraining geometries" upon predetermined 2D sketches or 3D models, and these dimensions and constraints are the "parameters, or input points, that you [the designer or engineer] would then change to update or edit the [model]" (Waguespack and Tremblay, 1). To date, primarily constraint-based modelers integrate with BIM, while algorithmic modelers are mostly reserved for form finding, and the formal integration of design constraints. The goal of this paper is to explore new design methodologies and methods that unify these two related, and yet disparate, design tools. The development of an effective and efficient workflow that moves between the algorithmic, the constraint-based, and BIM is the primary purpose of this design research.

2. AREA OF RESEARCH: DESIGN METHODS AND METHODOLOGIES

For the purpose of this paper, we have selected *Grasshopper*, "a graphical algorithm editor tightly integrated with [McNeel] *Rhino*'s 3-D modeling tools", developed by David Rutten at Robert McNeel & Associates (www.grashopper3d.com), as an example of an algorithmic modeler, and *Autodesk Inventor* as an example of a constraint-based parametric and BIM modeler. Communication between these two platforms occurs exclusively at the level of *Microsoft Excel* spreadsheets, through the sharing of data within a common domain. The *gHowl* add-on for *Grasshopper* and the *iPart* and *iFactory* tools within *Inventor* are the specific tools of data input and output for these two programs. The optimization and specialization of both software platforms is a key goal of this research. Specifically, we are seeking to leverage the iterative and recursive parametric modeling capabilities of *Grasshopper*, to supplement the "piece-by-piece" logic of Inventor; while Inventor will be used primarily to produce BIM and presentation quality drawings. The introduction of *Excel* in to the design workflow reduces the focus of design parameters to quantifiable pieces of data, or mathematical relationships, and exposes the designer directly to the underlying data model that drives his or her project.

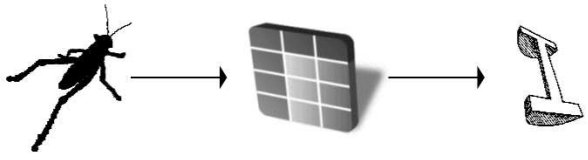

Figure 1. Workflow Diagram: from Grasshopper to Excel to Inventor

In general, the proposed workflow begins by establishing a series of user-defined parameters in Grasshopper either through direct panel input, interactive dynamic sliders, or culled from direct manipulation of referenced Rhino geometry. Through the application of *both* mathematical and geometrical means, the parameters are processed and analyzed, which generates the parametric design, or space, as well as a new series of driven parametric dimensions. These combine with the design parameters to form the data model. A key difference to note exists between a "user-defined parameter" and a "driven parametric dimension". A user-defined parameter constitutes the input and first buildings blocks of the parametric design process, which determine the basic geometric principles involved. A driven parametric dimension, on the other hand, comes about as a result of measurements and analysis performed on the generative design. These driven parameters may drive further generative principles within the design process. Finally, and perhaps most importantly, the designer must deal with the primary "design parameters": those pieces of data by which the designer can *evaluate* the performance and efficacy of his or her design.

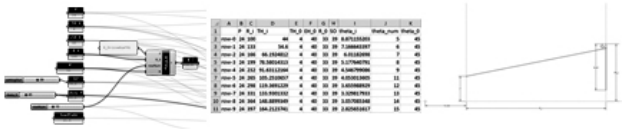

Figure 2. Left: User Defined Parameters in Grasshopper; Middle: Driven Parametric Dimensions in Excel; Right: a base Sketch in Inventor, showing the desired sight line

3. CASE STUDY

For our research, we have chosen to focus on the parametric relationships guiding the design of seating arrangements in concert halls. To expand further on the distinctions made above (Sec. 2, Par. 2), it will be helpful to look at the following case study. In this case, the specific *user-defined parameters* are primarily ergonomic, and many come directly from the building code. For example, in determining the placement of seats to achieve unobstructed sight lines of a particular "viewing point" (a *design parameter*), a typical eye-height from the floor (and from the eyes to the top of the head), and a row-to-row distance can be established by the designer (*user-defined parameters*). These parameters, using simple trigonometry and arithmetic series, can determine the ideal seat locations within a coordinate system (*driven parametric dimension*). Although eye or head heights and row widths are human dependent, and therefore, in a sense, not truly "user" defined, designing further parameters into the process will add layers of designer specificity. In this case, the distance from the first seat to the viewing point changes, and will alter the preferred seat rake (see Fig. 3). Even changes in seat type (with or with-out padding) and manufacturer can alter the outcome.

While designers examine the problem of sight lines primarily in section, they must design the layout of the rows themselves in plan. Seat width is of primary concern, and the most freedom occurs in the choice of a "setting out point", or the center point for the arcs of seats. This can be identical with the viewing point or anywhere in front or behind it. This will affect the depth or shallowness of the rows' arcs. Here, the primary design parameters may be financial, as the owners look to pack as many seats in the hall as possible, while maintaining adequate site-lines. Similar considerations will be necessary for the placement and design of balconies, as well as gangways and paths of egress.

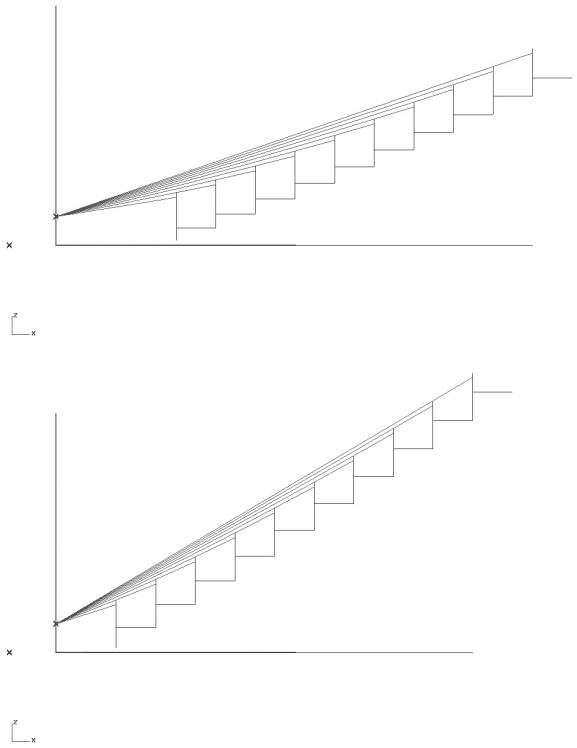

Figure 3. Sightlines in Grasshopper, updating parametrically, as the depth of the first seat row decreases, increasing the rake

Most importantly, in the design of concert halls, all of these design decisions will have an impact on the larger design parameters of Time Delay Gap and Reverberation Time that are essential in determining the quality of these spaces (see Beranek). The interplay between the design parameters of the seating arrangement and of the acoustical performance of the hall quickly becomes a very complicated design problem. For example, in a certain style of seating now popular, known as terrace-style seating, blocks of seating are placed throughout the hall and raised up slightly above the floor plane and the seats in front of them (Fig. 4). This provides a reflective surface on the front (and sides) of these terraces that can be effectively leveraged in the acoustical design of the hall. Of course, changing the height, pitch, or depth of the arc of these terraces will have wide-ranging effects on the seating design parameters. While traditional design methods might make determining these effects a tedious process, with parametric computational tools, they can be analyzed in real time, for both their acoustical and seating arrangement performance.

Figure 4. The Disney Concert Hall in Los Angeles, with its terraced-style seating; designed by Gehry Partners and Nagata Acoustics

Unfortunately, there are no current software platforms capable of effectively analyzing the complex nature of this problem, and interfacing with the design and production software actually capable of managing such a project. An algorithmic solver, such as *Grasshopper*, could analyze the problems, and even provide graphical feedback, but to maintain that adaptability over the course of an entire project would be quite tedious. A BIM program on the other hand, such as *Inventor*, would be flexible enough to adapt to the various parameters, however, to perform the actual analysis within the software would be quite difficult, and likely require some supplemental source coding. This led to the development of our proposed workflow, necessitated by the iterative and recursive nature of the seating problem. Specifically, the placement of any individual seat in the layout requires prior knowledge of the location of all other seats in the plan that may obstruct that seat's view. In the typical "piece-by-piece" Inventor workflow, this would require the tedious placement of each row in order. Even to use parametric *iParts* would require manually adding a new line to the *iFactory* table for each row. In *Grasshopper*, however, we reduced the relationship to a single equation describing the placement of seats within a mathematical series, which drives the generation of the seating geometries. We can update the number of rows, as well as all of the other parameters, through either numerical input, with a dynamic slider, or even by direct manipulation of Rhino geometry, and the geometry updates in real time. The data model driving the updated geometry can be exported to a properly formatted *Excel* spreadsheet and through a simple copy and paste in to the *iPart* table, the entire series of parts (or rows in this case) are generated; they merely need to be placed in an assembly file. At this point, we begin to leverage the great strength of *Inventor*, because the base assembly file can be layered with any number of details, from actual seats, to material finishes, to the form and structure (and even structural framing) of the hall. Furthermore, we can push this data back out to Grasshopper, or any other suitable program for further analysis. This would be particularly useful, for example, in looking at the finishes, and the absorptive qualities of the various materials.

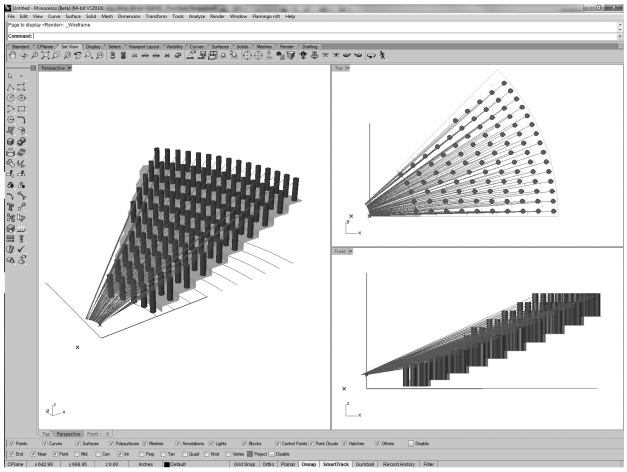

Figure 5. Graphical feedback of the Grasshopper algorithm in the Rhino environment

While similar effects could be achieved by using *Excel* spreadsheets alone (and propagating the mathematical series) to drive *iFactories*, the geometric and graphical feedback provided by *Grasshopper* makes this workflow much more fruitful in the design process. Additionally, this workflow requires no special knowledge of *Excel*, or *VB.NET*, scripting. The use of *Grasshopper* becomes even more essential when the designer must reference existing conditions, such as the floor plan, from a DWG, or other architectural drawing. Only through the *Grasshopper* workflow can the designer translate those conditions into a data model accessible in *Excel*, and other tools down the line.

4. CONCLUSIONS

Research will continue to focus on the exact place of each tool in the workflow, considering the possibility that more than one can likely do each task, and that one or the other may be more properly suited to the task-at-hand. For example, in our case study, to parametrically model the section for a seating arrangement, the software must incorporate some iterative or recursive functions. Such capability typically lies in the realm of an algorithmic modeler. To solve the plan, on the other hand, requires only trigonometric and geometrical analysis, because seats are generally spaced at equal intervals along a fixed path, a simple process for constraint-based modelers

In general, the proposed process fills two large gaps in the *Grasshopper* workflow, especially regards the difficulty in extracting drawings from *Grasshopper* to a level of detail that might be required of shop or working drawings. Further investigations include the ability to translate free-form NURBS geometries modeled in *Rhino* in to the *Inventor* BIM environment, extending its formal capabilities. For example, one could write a *Grasshopper* script that takes a NURBS surface and tessellates it in some controlled manner, then exports the vertex points to *Excel* for import in to *Inventor*. From these works points, the designer could model a series of panels in *Inventor* to populate the surface. Finally, *Inventor* can produce the shop drawings and BOM (Bill of Materials).

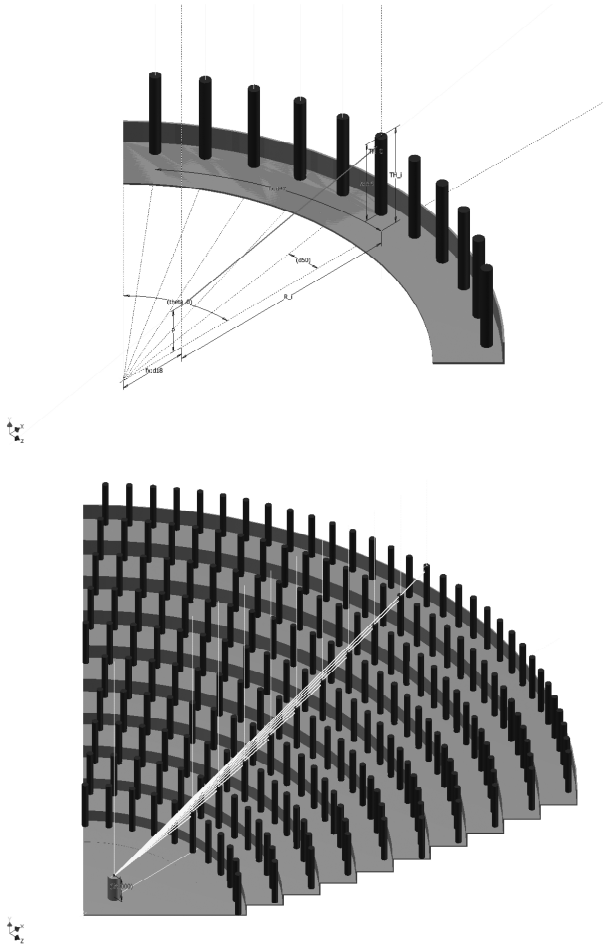

Figure 6. Part and Assembly files in the Inventor parametric environment

While constraint-based modelers, including *Inventor*, typically have difficulty with NURBS geometry, this workflow bypasses this obstacle altogether by using an algorithmic modeler to "translate" the geometry into *Inventor*'s "native" language. By reducing the NURBS geometry into a raw data model, *Inventor* can re-create the geometry based off work points, planes, and axis defined entirely within *Inventor*. Other workflows require that models pass through one or more conversions between file types, which inevitably leads to data loss and corruption, and lack of accuracy in the final product. For example, to get NURBS geometry out of *Rhino* in to *Inventor* would one might save the native *Rhino* .3DM file as an .OBJ or .3DS file, and then import this into *Inventor*. Because *Inventor* has no native support for NURBS these file types will have likely converted the geometry to a polygonal mesh model, which provides, at best, a close approximation of the original design. Even at this point to turn this polygonal mesh into a satisfactorily adaptable parametric model will take a great amount of work, and may require essentially starting from scratch with the imported model as nothing more than a guideline.

With the workflow presented in this paper, however, the designer has complete control over the manner in which the NURBS geometry translates into the *Inventor* design environment. Most importantly, this process adds an entire layer of sophistication into the design process that directly relates to fabrication. For example, the way in which a designer chooses to tessellate a surface (and provide work points for *Inventor*) depends on the number of panel types, the framing system (or the absence thereof), the method of connection to one another (and the frame), and any other number of design parameters. Therefore, by solving an efficient way to transfer a design between the algorithmic and the constraint-based parametric environments, between *Grasshopper* and *Inventor*, the designer must also grapple with design issues related to the actual construction and completion of the project. The design of a data model that allows for efficient parametric modeling and detailing in *Inventor* goes hand in hand with a design logic that allows for efficient and economical fabrication and construction of the project, a boon for designers and those that employ them.

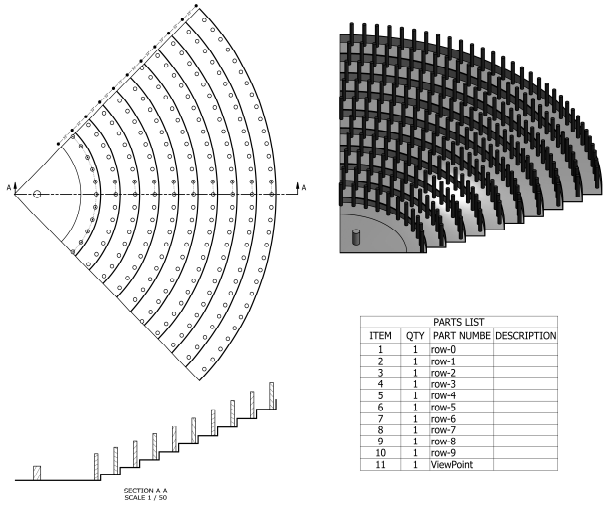

Figure 7. A drawing file produced by Inventor, including Bill of Materials (BOM)

ACKNOWLEDGEMENTS

The research in this paper was conducted in conjunction with a graduate architectural elective course, ARCH 662, entitled, *The Design of Parametric Space*, taught by Assistant Professor Rhett Russo, at the New Jersey Institute of Technology, in the Fall of 2011.

REFERENCES

[1] Leo Beranek, **Concert Halls and Opera Houses: Music, Acoustics, and Architecture**, Springer, 2004.

[2] Renee Cheng, "Computing Technologies: Building Information Modeling (BIM)", **Architectural Graphic Standards, Eleventh Edition**, Hoboken, New Jersey: John Wiley & Sons, Inc., 2008.

[3] Michael Meredith, **From Control to Design**, Actar, 2008.

[4] Curtis Waguespack and Thom Tremblay, **Mastering Autodesk Inventor 2011 and Autodesk Inventor LT 2011**, Indianapolis, Indiana: John Wiley & Sons, Inc., 2010.