AC 2011-2728: CONCEPTUAL DESIGN EXPLORATION IN ARCHITECTURE USING PARAMETRIC GENERATIVE COMPUTING: A CASE STUDY

Dr. Stan Guidera, Bowling Green State University

Stan Guidera is an architect and chair of the Department of Architecture and Environmental Design at Bowling Green State University. His primary teaching and research area is in 3D applications for computer aided design for architecture and construction.

Conceptual Design Exploration in Architecture Using Parametric Generative Computing: A Case Study

Abstract

This paper documents design strategies using Grasshopper and Rhino 3D as an instructional tool for conceptual design. It discusses the underlying concepts of generative design and includes examples using Grasshopper with Rhino 3D for both massing and for basic structural layouts. It also discusses the necessary skill set, beyond that associated with the operation of the underlying CAD applications, required for students to utilize these applications. It then proposes a framework for incorporating generative design into CAD courses utilizing a 2-D to 3-D sequence of instructional activities.

Part 1: Introduction

The digital revolution and its associated discourse is increasingly influencing all of the design fields, particularly architecture [1]. In his book Constructing Complexity, William Mitchell referenced to shift to digital design in architecture stating that “buildings were once materialized drawings, but now, increasingly, they are materialized digital information – design with the help of computer-aided design systems, fabricated by means of digitally controlled machinery, put together on-site with the assistance of digital layout and positioning devices, and generally inseparable from flows of information through global computer networks.”[2]

However, design exploration is an integral aspect of the design process in any discipline. Traditionally sketching has functioned as a primary conceptual design tool due to its indeterminacy and ambiguity. Goel [3] suggested that the ambiguity in sketching promoted cognitive shifts from one proposed conceptual idea to other alternative concepts, a process he referred to as lateral transformation. Won [4] proposed that during the drawing process designers demonstrate a “seeing behavior” in which they will concentrate on the figural properties of a sketch. He stated that as a result the designer may “see the image as something else” and added that the shift of ‘seeing’ to ‘seeing as’ stimulates imaging. Similarly, Suwa and Tversky [5] proposed that as designers inspect sketches “they see unanticipated relations and features that suggest ways to refine and revise ideas.”

As design practices have been restructured around Computer Aided Design (CAD), processes to integrate digital technologies in conceptual design have been ongoing. The success of these applications has been limited. CAD has been perceived as a medium intended for production that is difficult to use in the early stages of the design process where the priority is creativity rather than precision. [6, 7, 8] The precision inherent in CAD results in a lack of ambiguity making commercially available computer aided design applications largely ineffective as a medium for design exploration.

Most commercial CAD applications have provided some form sketch emulation mode [9] or have provided display options that generate digital representations with hand-drawn characteristics. The former were largely perceived as ineffective (Figure 1). The latter were typically perceived as yet another alternative display which, while providing an effective representation tool, has not been widely adopted as a design interface (Figure 2).

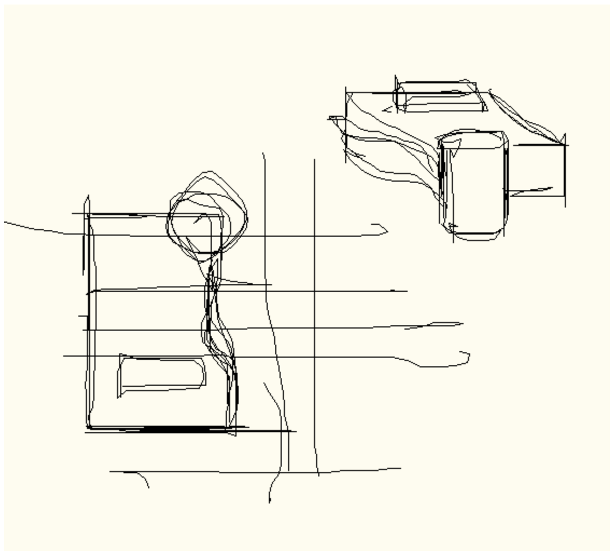


Figure 1. AutoCAD Sketch command example

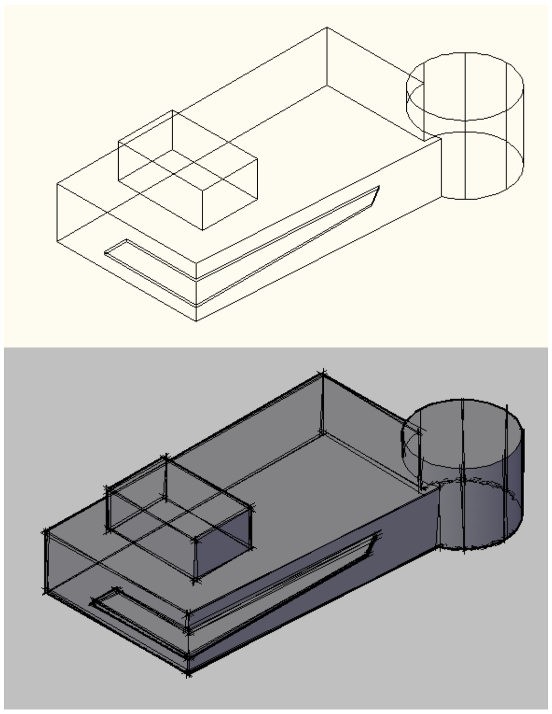


Figure 2. AutoCAD 3-D Wireframe display (above), with Sketch emulation (below)

As an alternative, applications have been developed that attempt to operate in a manner that emulated traditional manually-based graphic techniques. These applications can include tools to support 2-D sketching mechanisms. Autodesk's Architectural Studio interface was based on a trace-paper overlay mechanism in which designers could use drawing tools that created line-work modeled after traditional markers and pencils. It could also merge sketches into 3-D models, thus bridging the gap between 2-D and 3-D graphics. However, its limited adoption has been attributed to the lack of wide-scale adoption of pen-based input devices [10]. Many other sketch based 2-D to 3-D have been proposed or developed by researchers over the past decades. These include seminal applications such as Sutherlands SketchPad, a constraint-based drawing environment developed in the 1960's, and STRAIT, a program developed in the 1970's that interpreted sketch geometry as straight lines [11]. Recent developments in the interface between sketching and digital design include "Digital Clay,"[12] "SmartPaper,"[13] and displacement modeling. [14]

However, it may be argued that attempts to re-create the traditional sketch-based design processed with a computer may be an inappropriate strategy for conceptual design in a digital environment. Further, it may be argued that such a strategy is an extension of the early adoption of CAD, in which many of the commercially available applications were developed and used as electronic versions of manual drafting practices. Other than a change in the medium and devices used, no change in the actual processes associated with producing the work actually occurred.

This has resulted in the criticism that CAD has failed to meet the expectations of its users and that it's true potential has gone unrealized. Therefore a more effective strategy may lie in re-conceptualizing conceptual design by utilizing processes that embrace and exploit computation.

Generative design approaches have emerged from the search for strategies to facilitate the exploration of alternative solutions in design, using computers as variance-producing engines to navigate large solution spaces and to achieve unexpected but viable solutions. [15] Kolaravec used the term "digital morphogenesis" to refer to design processes in which digital media is not used for representation but as a generative tool for the derivation of form and its transformation. [16] He stated that "the predictable relationships between design and representation are abandoned in favor of computationally-generated complexities" and that "models of design capable of consistent and continual dynamic transformation are replacing the static norms of conventional processes.' For Kolaravec, generative computing, or, as he referred to it, "digital morphogenesis," is a "radical departure from centuries-old traditions and norms in architectural design" – the emphasis shifts from form-making to form-finding."[16]

In generative design, algorithmic procedures are often used to produce arrays of alternative solutions based on predefined goals and constraints, which the designer then evaluates to select the most appropriate or interesting. [17] This position is reiterated by Oxman [1], who stated that "the generative model is the design of, and interaction with, complex mechanisms that deal with the emergence of forms deriving from generative rules, relations and principles." However, she also argued that designer interactivity is a key component. She stated that "Interaction has a major priority in this model" and added that "in order to employ generative techniques in design, there is a need for an interactive module that provides control and choices for the designer to guide the selection of desired solutions."

Chase [18] made a clear distinction between CAD and generative design. He argued that CAD applications do not have the exploratory potential of generative computing. He further argued that "traditional CAD software can aid students in understanding their designs, and develop their knowledge and skills in areas such as geometry, but they act only as aids; the user must directly input and manipulate forms", and added that "the power of generative design tools is that these can guide a novice down an exploratory path."[18]

The use of generative design technologies have been expanding architectural education and among design professionals as well. Generative design is a parametric computer modeling technology that is typically operated using an alternative interface for a Computer Aided Design application. Examples of these are Generative Components, which works with Microstation, and Grasshopper, which works with Rhino 3D.

Part 2. Developing an introduction to Generative Computing using Grasshopper

Generative design processes are characterized by the following:

1. A design schema that provides criteria requirements
2. A means of creating variations
3. A means of selecting desirable outcomes

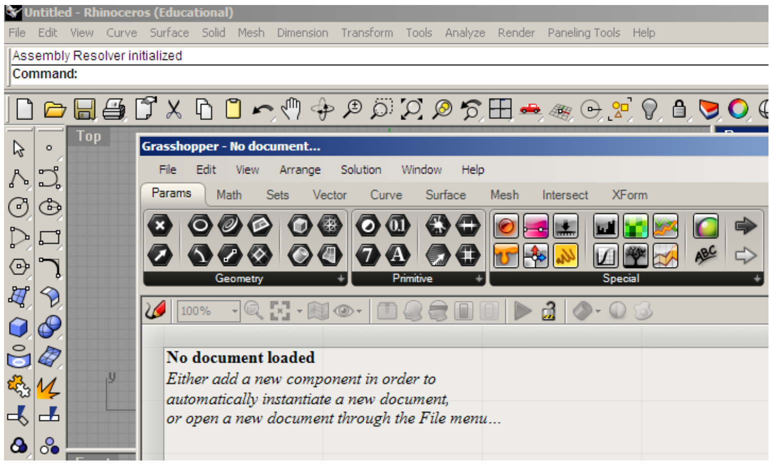
Based on these characteristics, generative design environments provide significant advantages for conceptual design as the emphasis is on exploration of alternatives. However, one of the most significant advantages is that generative design environments are dynamic and interactive, providing real-time visual feedback as the geometric and dimensional variations are manipulated.

A generative computing application that is rapidly expanding in use is Grasshopper, which runs with Rhino 3D. This expanded can be attributed to two factors. First, the extensive modeling capabilities of Rhino 3D, particularly in terms of nurbs (non-uniform rational b-spline) curve and surface modeling, has lead to its widespread adoption among architectural educators and professionals. The command structure has many parallels with applications AutoCAD as well as 3D Studio, thus reducing the learning curve for students already familiar with this application. Secondly, the graphical interface of Grasshopper provides an explicit representation of the geometric relationships and sequences used to generate the digital model. This explicit representation is linked to the Rhino 3D viewports. This enables designers to receive immediate visual feedback as these relationships are manipulated by user-defined mathematical and geometric parameters.

Additionally, Grasshopper utilizes a simple strategy for storing variations of the results of these manipulations. A process, driven by clicking on a single icon, is used to store an option. When combined with layers, this process, referred to as “baking”, can store any number of variations on discrete layers, so the options can be saved for future display and review.

Grasshopper Overview

Grasshopper functions as a plug-in (add-in application) for Rhino 3D. However, unlike most Rhino plug-ins, Grasshopper utilizes a separate interface window that references geometry generated in the original Rhino 3D file (Figure 3). Additionally, Grasshopper operations are saved in a separate file format (.ghx) from Rhino3D. In Grasshopper, operations are built upon initial Rhino geometry, in some cases as simple as a point. The Rhino geometry is typically linked to grasshopper using the Grasshopper Parameters panel, where single or multiple points or curves in Rhino are defined as the geometry that will be used as the basis for further operations.


Untitled - Rhinoceros (Educational)
File Edit View Curve Surface Solid Mesh Dimension Transform Tools Analyze Render Paneling Tools Help
Top
Grasshopper - No document...
File Edit View Arrange Solution Window Help
Params Math Sets Vector Curve Surface Mesh Intersect XForm
Geometry
Primitive
Special
100%
Assembly Resolver initialized
Command:

No document loaded
Either add a new component in order to automatically instantiate a new document, or open a new document through the File menu...

Figure 3. Grasshopper interface

Once the Rhino geometry is linked to grasshopper, the relationships “piped” – the input of one operation is channeled into a second operation or operations. These are graphically represented by operation icons and input and output curves that connect between the input and output parameters of the operation icons. The end result of an operation is displayed in the Rhino viewport. At any point along a sequence of channeled operations, the display of the result of that operation can be turned on or off. Therefore, an operation in the sequence can be piped into two or more alternatives, and the designer can toggle the display of the options generated by the different processes.

Figure 4 illustrates an example in which a Grasshopper curve parameter icon is “set” to a nurbs curve created in Rhino. In setting a parameter to specific geometric element, that element is then used as input for subsequent operations. In this example. the curve parameter is “piped” into the C (curve) input of the “offset curve” operator. The offset distance input D is driven by a number slider. The “offset curve” icon output C generates a curve that is offset the D distance set by the user with the number slider. The “offset curve” operator output C is then piped into the B (input geometry) parameter of the “extrude curve” operator. The second input parameter on the “extrude curve” operator is D, which is both direction and distance. A Z-direction vector parameter is piped into D to establish the direction. The same number slider used to determine the offset distance is piped into the Z vector parameter. As a result, as the number slider is manipulated, the curve is extruded the same height as the offset distance so that the further the curve is offset the higher the curve is extruded. The result is illustrated in the Rhino 3D viewport adjacent to the Grasshopper window in Figure 4. The only actual geometry in the Rhino 3D file is the single curve used in the initial “set curve” parameter operation.

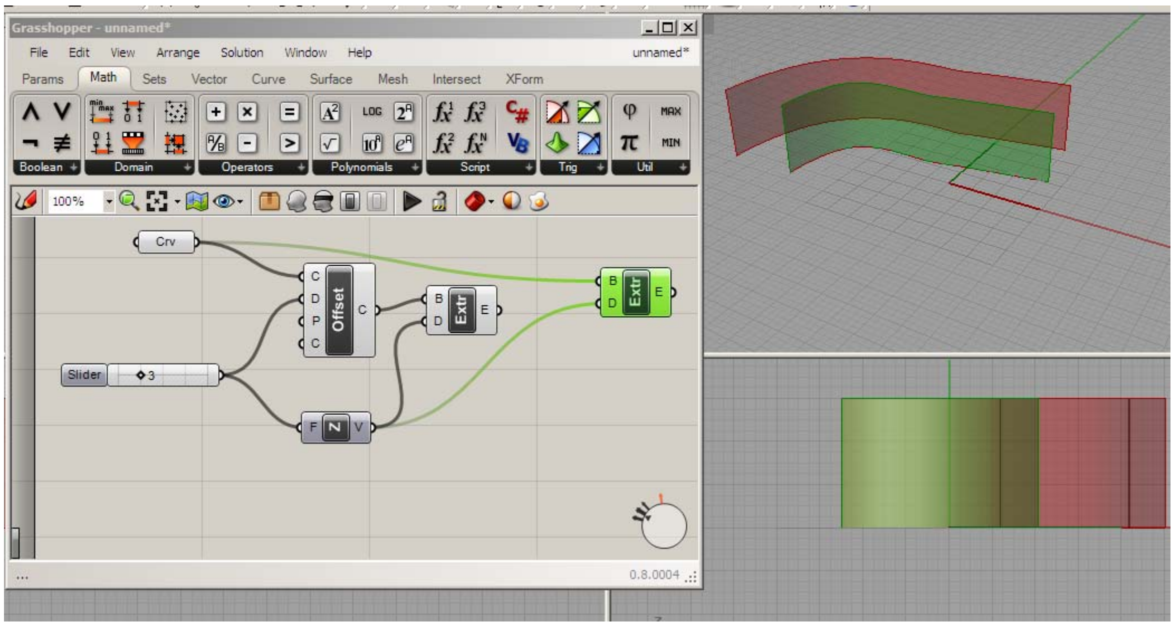

Grasshopper - unnamed*
File Edit View Arrange Solution Window Help
unnamed*
Params Math Sets Vector Curve Surface Mesh Intersect XForm
Boolean Domain Operators Polynomials Script Trig Util
100%
Crv
Offset
C D P C
Extr
B D E
Slider
3
F N V
0.8.0004
Figure 4. Grasshopper interface with offset curve and extrude operations.

Figure 5 illustrates an expanded example using the example illustrated in figure 4 as a starting point. A second extrude operator is added and the original curve is piped into its B (geometry) input. The Z vector establishing the direction of the first extrusion is then piped into D.

However, D is modified with a mathematical formula, but rather than utilize the icon-based mathematical operations available in Grasshopper, the Expression editor, available for many input or output parameters, is used to directly apply a fixed formula. In the example in figure 5, D/2 is input into the D expression editor of the new extrusion operator. This will divide any input into D by two. As a result, a single number slider is used to drive the offset distance and both extrusion, with the original curve extrusion always being ½ of the distance and the height of the offset curve. Variations of the scheme are generated using the number slider.

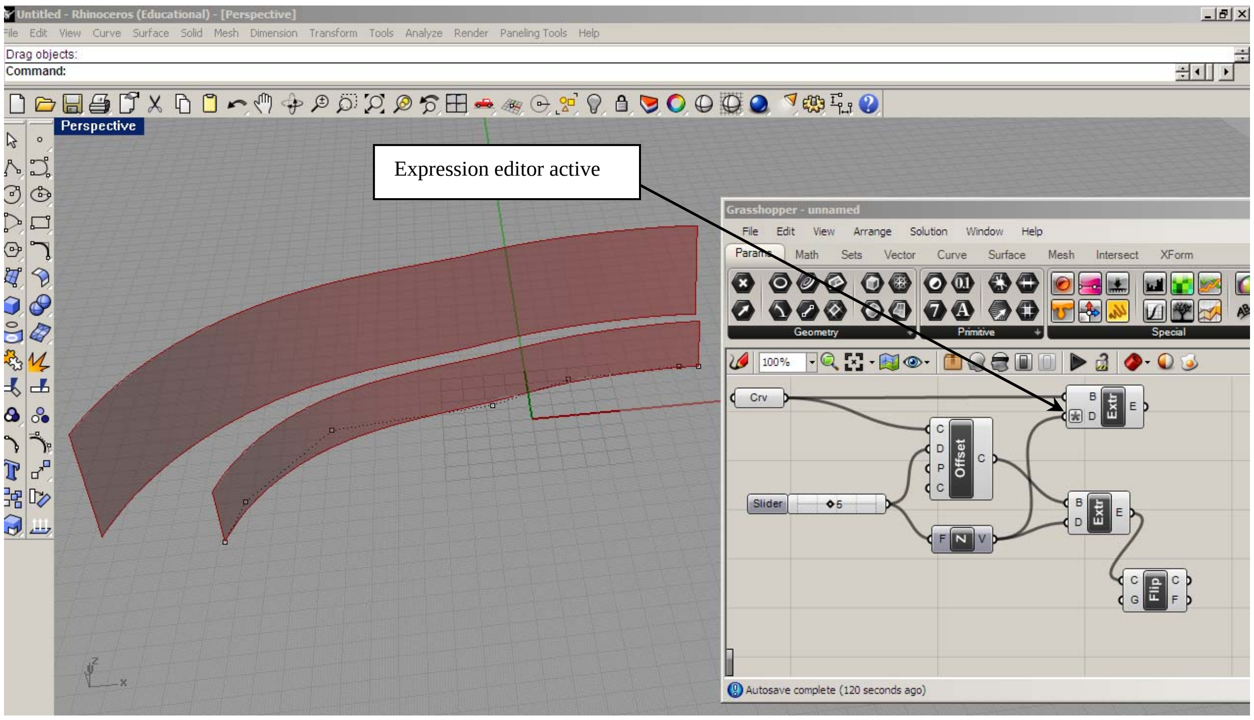

Untitled - Rhinoceros (Educational) - [Perspective]
File Edit View Curve Surface Solid Mesh Dimension Transform Tools Analyze Render Paneling Tools Help
Drag objects:
Command:
Perspective
Expression editor active
Grasshopper - unnamed
File Edit View Arrange Solution Window Help
Params Math Sets Vector Curve Surface Mesh Intersect XForm
Geometry
Primitive
Special
100%
Crv
Offset
Slider
5
Extr
Extr
Flip
F N V
Autosave complete (120 seconds ago)

Figure 5. Grasshopper interface with offset curve and extrude operations with expression editor used to develop a parametrically-driven height dimension for one extrusion that is always twice the height of the other.

Figure 6A and 6B illustrate a further expanded example using the example illustrated in figure 5 as a starting point. In figure 6A "edge surface" operator is added to the Grasshopper window. In Rhino 3D, the basic application of the edge surface command creates a 2-D surface between two curves. Similarly, in Grasshopper, the edge surface operator uses the curve parameter set from the original curve in Rhino as the first curve and to create an offset curve in Grasshopper, which is manipulated with the number slider as the second curve. The result is a planar surface that adjusts dynamically with the curve as the number parameter for the distance is manipulated. In figure 6B the example is further developed into a closed 3-D mass. The original Rhino curve and Grasshopper offset curve are disconnected from the edge surface operator and replaced with the output of the Grasshopper extrusion operators which had been applied to the original and the offset curve in figure 5. This creates surfaces between all edges simultaneously. In Grasshopper, the resulting geometry is a three-dimensional volume that has a width dimension and a height dimension manipulated by the single number slider.

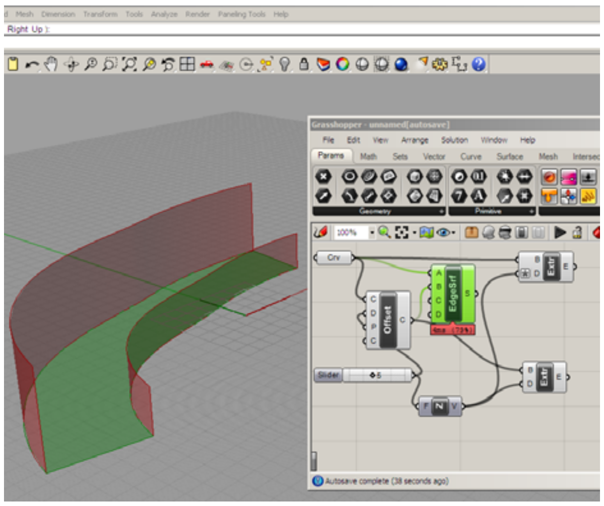


Figure 6A: Edge surface applied between original Rhino curve and Grasshopper offset curve to create 2-D surface.

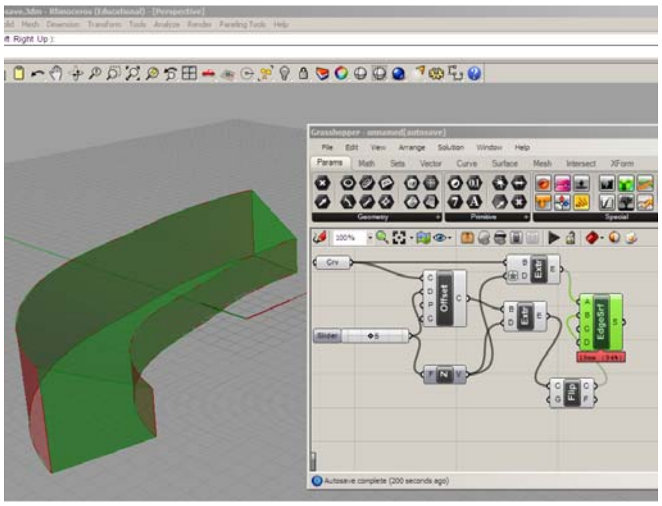


Figure 6B. Edge surface applied between Grasshopper extrusions of original Rhino curve and Grasshopper offset curve to create 3-D mass.

Organizing the class activities

Ongoing review of the research related to generative design lead architecture faculty at Bowling Green State University to develop an introductory assignment which would enable both students and instructors to experiment with utilizing GC (Generative Computing) as a design platform. While expanding student's technical skills with software was an intended learning outcome, the primary objective was to analyze the effectiveness of the technology as a medium for design exploration, particularly when used by novice designers. The course selected to introduce generative design concepts had been previously developed as an elective upper-level architectural computing class emphasizing advanced BIM (Building Information Modeling) content. Therefore, it was necessary to limit the course time allocated to an experimental introduction of generative design to approximately four weeks. During this period students would need to develop competencies with Rhino 3D and Grasshopper. Therefore, the instructional strategy was divided into three areas of content focus:

1. Content Area One: A single-track introduction to 2-D and 3-D operations in Rhino 3D (Version 4.0, SR8) which was to draw extensively on students prior experiences.
2. Content Area Two: A two-track phase which involved covering 3-D operations specific to Rhino 3D while simultaneously covering 2-D generative computing operations in Grasshopper (Version 8.0004).
3. Content Area Three: A single track 3-D generative computing track using both 2-D and 3-D operations in Grasshopper, culminating in a "design exploration project" assignment in which students were to use Rhino 3D and Grasshopper to generate three discrete alternatives using the Grasshopper "bake" operations.

The sequence of the tracks used in this instructional strategy is illustrated in Figure 7.

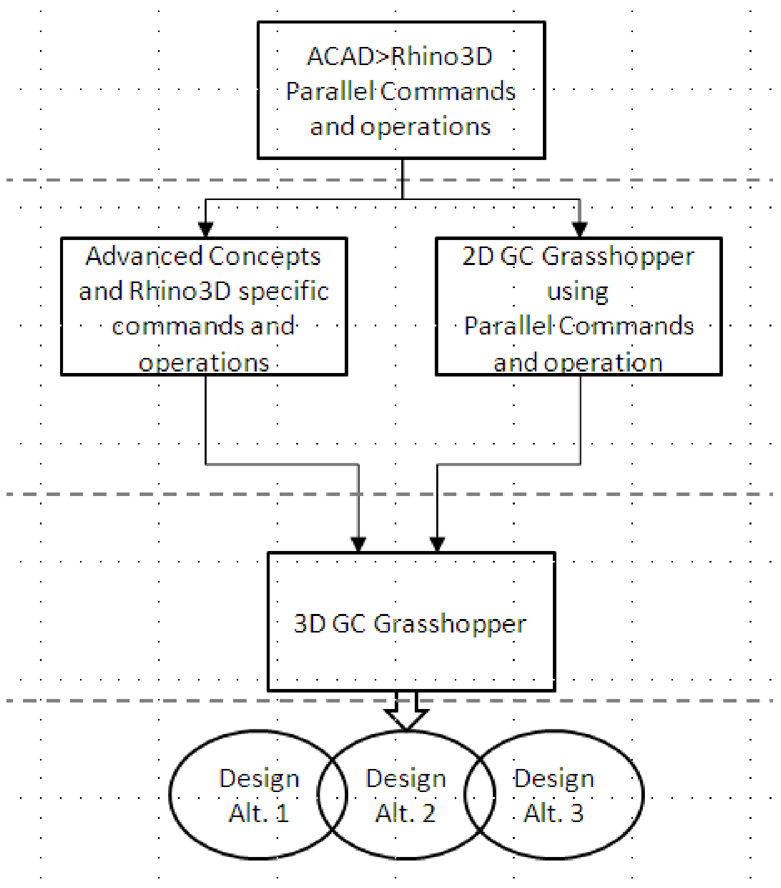

ACAD>Rhino3D Parallel Commands and operations
Advanced Concepts and Rhino3D specific commands and operations
2D GC Grasshopper using Parallel Commands and operation
3D GC Grasshopper
Design Alt. 1
Design Alt. 2
Design Alt. 3

Figure 7. Instructional strategy diagram for introducing GC concepts

Embedded within the stages of the class project organization are the three characteristics of generative design, which were aligned as noted in Table 1.

Table 1. Alignment of three characteristics of generative design and class project stages

A design schema that provides criteria requirements	Design Project Parameters
A means of creating variations	Grasshopper parameters and operations
A means of selecting desirable outcomes	Grasshopper “Baking” operation

The initial area content focus was structured to identify the parallel operations between AutoCAD and Rhino in order to draw on the student’s existing skill base to introduce Rhino3D. The primary parallel operations and commands covered are documented in Table 2.

Table 2. Primary ACAD to Rhino parallel operations and commands

ACAD/3DS Max Concept:	Rhino Concept:
Relative and Absolute Coordinates	Relative and Absolute Coordinates
Viewport Navigation	Viewport Navigation (parallels to 3DS MAX)
Zoom, Pan, and View options	Zoom, Pan, and View options
Viewport display: Wireframe, Shading options	Viewport display: Wireframe, Shading options, render options

Object snaps	Object Snaps
Drawing Aids (Grid snap, Ortho modes)	Drawing Aids (Grid snap, Ortho modes)
Function keys and keyboard shortcuts	Function keys and keyboard shortcuts (parallels with AutoCAD)
Layers	Layers
User coordinate Systems, Work Planes	C-plane Operations: Setting c-planes, orienting views to c-planes
Line and Polyline commands	Line and Polyline commands
2-D Shape creation	2-D Shape creation
Grip editing	Control point editing
Polyline editing (PEDIT)	Control point editing, curve degrees
2-D Geometry Editing (Offset, Trim, Fillet, Etc.)	2-D Geometry Editing (Offset, Trim, Fillet, Etc.)
Extrusions	Extrude Planar Curve
Solid Primitives	Solid Primitives
Boolean operations (Union, Subtraction, Intersection)	Boolean operations (Union, Difference, Intersection)
2-D and 3-D Geometry Manipulation: Move, copy, rotate, scale	2-D and 3-D Transforms: Move, copy, rotate, scale
Inquiry Commands	Inquiry Commands
Block (ACAD) and Group (3DS Max)	Group
Lighting: Point, Direct, Spot	Lighting: Point, Direct, Spot

In the second section covering 3-D operations specific to Rhino 3D, the emphasis was primarily on advanced surface modeling. While some of these operations, such as lofts and sweeps did have analogous operations in AutoCAD, few students had experience with the CAD counterparts. Additionally, many of these operations were considerably more complicated in AutoCAD than in Rhino 3D. Therefore, these functions were covered only in Rhino. While it was not anticipated that all the advanced Rhino content would be used in Grasshopper operations, some content, such as cage editing, was included to reinforce the student's grasp of the software's functionality and modeling potential. The 2-D generative computing operations in Grasshopper which were covered along with the more advanced Rhino modeling concepts were primarily limited to curve operations. However, using grasshopper operations to divide geometry and operations used to move and scale geometry, called X-forms in Grasshopper, were also covered. A complete list of the topics covered in the Rhino modeling/Grasshopper concept content areas are listed in Table 3.

Table 3. Rhino 3D modeling/Grasshopper concept alignment

Rhino3D Modeling Concepts:	2-D generative computing operations in Grasshopper
Lofts	Set point
1-rail and 2-rail sweeps	Set curve
Flow along curve operation	Offset curve
Flow along surface	Input and Output Parameters
Curve Extrusion options	Number slider (Floating Point and Integer)

Curve degree	Expression editor
Surface control points	Divide curve
Surface editing (offset)	X, Y, and Z Vectors
Capping planar holes	Xform (Euclidean) Move and Rotate
Curve from Object operations	Xform Scale
Cage editing	Line: from Point
Record History function	Line: Through/between points
Extend curve	Frames (perpendicular frames on curve)
Extend surface	Geometry around point
Orient around curve	Piping surfaces
Advanced c-plane operations	
Rebuilding surfaces	
Changing curve directions	

Examples of the basic 2-D and 3-D generative computing content covered are documented in figure 8. In this student example, two Rhino curves, a line and an arc, have equally spaced points placed along their length using the Grasshopper "divide curve" command. The number of points along both curves is driven by a single slider. Using the Grasshopper Move X-form, the points are repositioned vertically in the Z-direction. Grasshopper then generates lines between points to form a frame of 2-D geometry. In figure 8, the height of the points generated by the Move X-form relative to the original curves is driven by two independent number sliders, allowing both the frame height and frame slope to be manipulated dynamically.

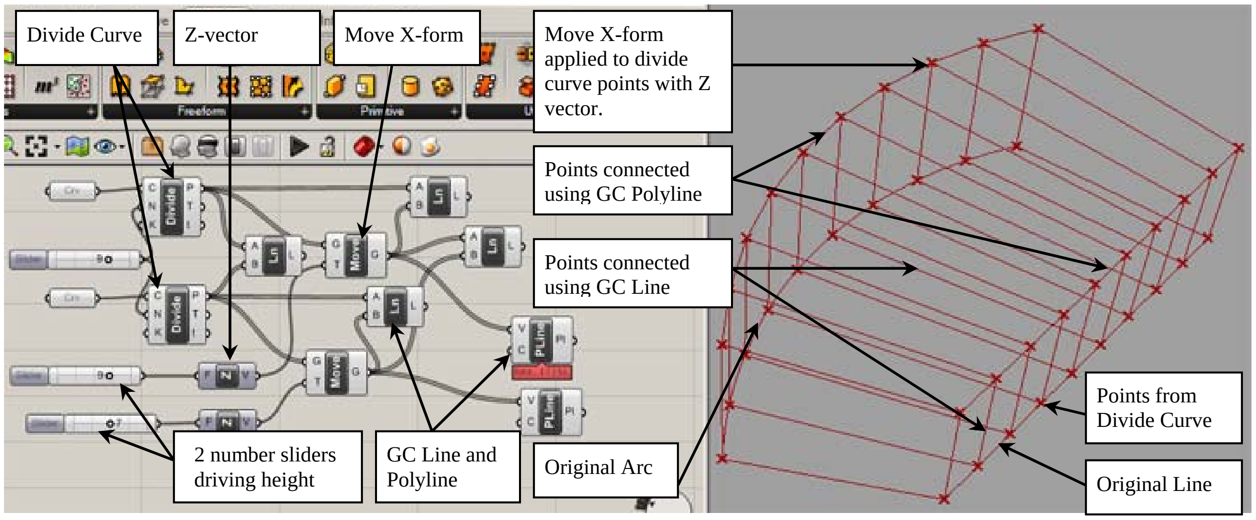

Divide Curve
Z-vector
Move X-form
Move X-form applied to divide curve points with Z vector.
Points connected using GC Polyline
Points connected using GC Line
Points from Divide Curve
Original Line
Original Arc
GC Line and Polyline
2 number sliders driving height

Figure 8. Basic 2-D and 3-D Rhino-to-Grasshopper GC modeling.

Once frames were developed, students could use the Grasshopper piping command to place a tube form around the linear geometry in order to simulate structural frames (Figure 9). Once the line geometry had been channeled into the piping operation, the results could then be "stacked" into a multi-level frame by using the output of the piping operation as input for additional Move X-forms. A student example is illustrated in figure 10. The height of both sides of the frame in this example is driven by a single number slider to facilitate stacking. Additionally, the vertical stacking is driven by the same slider. As a result, the heights of all levels of the heights of the individual frames could be dynamically manipulated and remain connected.

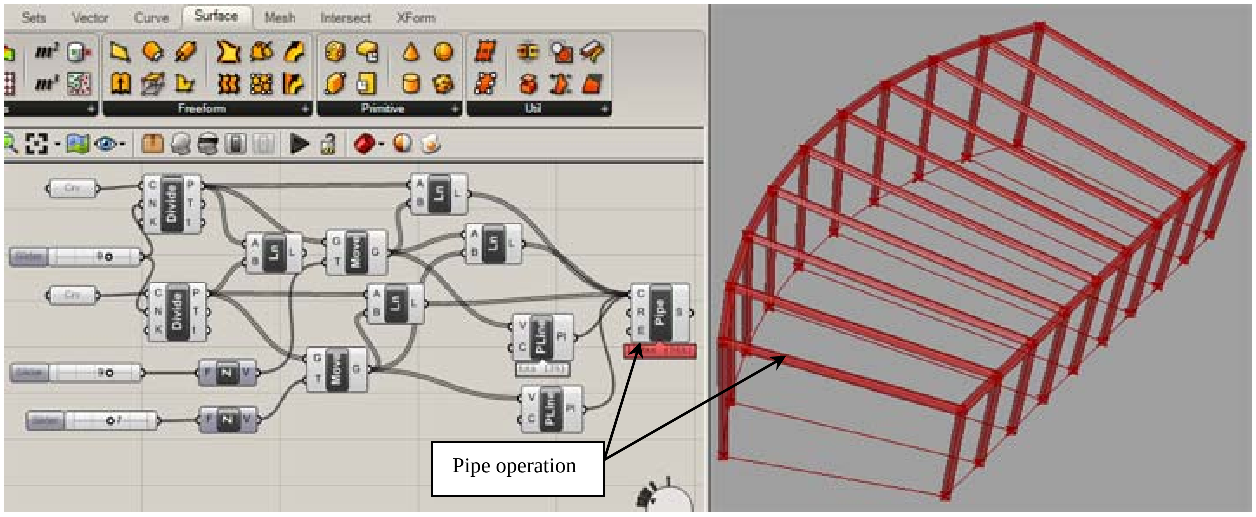


Figure 9. Tube-shaped form extruded along 2-D geometry using the Grasshopper pipe operation.

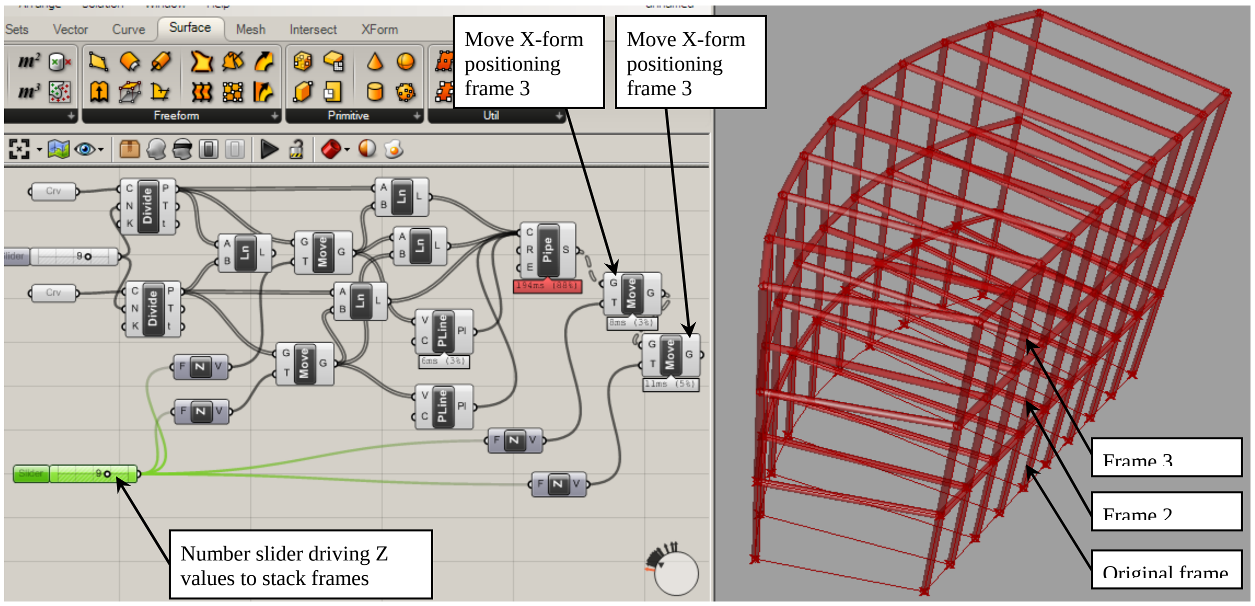


Figure 10. Multi-level frame generation. The height of both sides of frame is driven by a single number slider. The Z-direction vector assigned to all Move X-forms is driven by a single number slider to facilitate stacking, thus retaining frame continuity.

The third section focused on introducing Grasshopper operations using advanced Rhino 3D surfaces. In many cases the new operations were 3-D or surface operations which were analogous to the 2-D operations covered in the previous section. However, some of the content required substantial allocation of time. Specifically, material related to frames and lists proved challenging. Some additional advanced content and operations related to the use of formulas and functions was eliminated from the topics to be covered. The operations that were covered in the third section included those listed in Table 4. Sample outcomes are documented in Figure 10.

Table 4. Advanced Grasshopper operations with surfaces
Lists
Series
Flip (inverting curve/surface direction)
Lofts
Revolve
Points through curves
Set-plane operations
Grids
Meshes
Sweeps (1 Rail and 2 Rail sweeps)
Curve-piping
3-D primitives
Surface offsets
Planar Surface and operations
Cap Holes
Surface divide
Surface frames
Morphing
Baking operations

Design Project: Generating Design Alternatives

For the application project a computer model of the downtown area of a mid-sized Midwestern city was provided to the students. The mass model included limited detail, but did provide a real-world framework for the project. A vacant site was selected for the project location. The site was 260’ by 380’ (Figure 11).

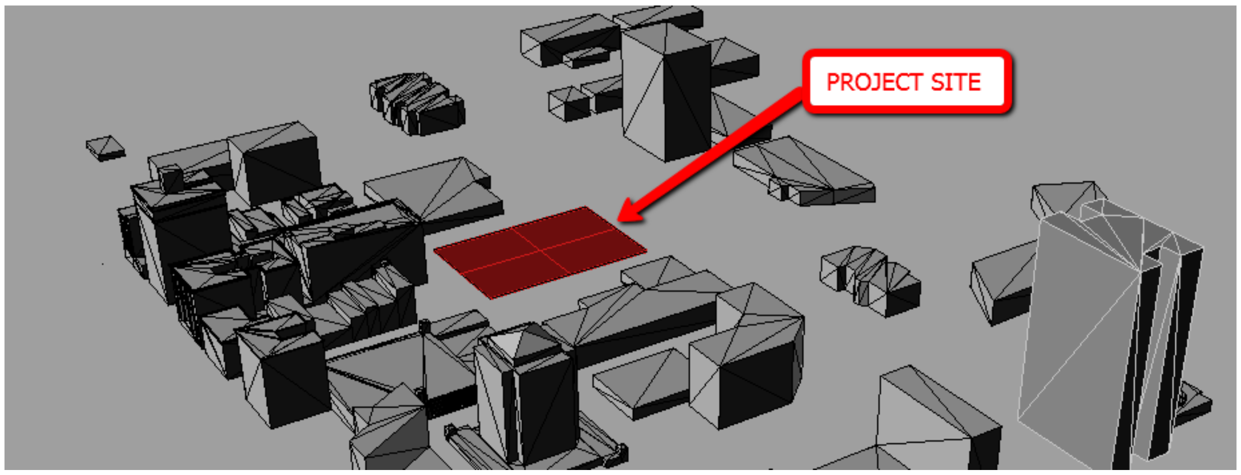


Figure 11. Project site and context.

Students were asked to use grasshopper to generate three design alternatives for the massing of a new office tower structure on the designated site. Project parameters were as follows:

- The building footprint was required to be at least 25% of the site but no more than 50% of the site.
- The structure was to be between 200' and 250' tall.
- The structure could not extend/project beyond the property line.
- The structure could not extend/project more than 15% beyond the building footprint.
- Floor-to-floor heights were set at 10' minimum and 20' maximum.
- The portion of the site remaining empty was to be considered as an urban plaza – however it was not to be developed as part of the project.
- The starting geometry in the Rhino file was to be limited to a single point and or curve segment.
- Only One .ghx (Grasshopper) file was to be used. This required students to establish geometric relations sufficient to drive variations that could generate the three alternatives. However, students were allowed to have geometry alternatives that could be swapped out within the file. For example, a file could contain both an ellipse and a rectangle that could be exchanged in an operation for different results. However, both operation icons were to be contained in the single .ghx file.
- Required project submission:
    1. Rhino .3dm file of initial geometry (point or curve segment)
    2. Grasshopper .ghx file
    3. Site model with the three final alternatives baked on three discrete layers Specific instructions were provided for layer as well as file naming conventions. For example, the three layers that were used to store the design alternatives were to be identified as Alt1, Alt2, and Alt3. However, students were encouraged to store any preliminary design development operations that were "baked" during the course of developing their proposal on any other layers under a naming convention of their choosing.

It was intended that the extent to which students utilized baking operations during preliminary design experimentation could provide an indication of the effectiveness of Grasshopper as a design exploration tool.

All class participants were able to generate at a minimum basic massing alternatives. Most relied on establishing integer-driven relationships that manipulated the massing through alternate floor plate geometries as well as X-form operations, number of floor plates, manipulation of floor-plate areas, and manipulations of floor-to-floor heights. Figure 12 documents the Grasshopper operations that were typical in many submissions. A point was positioned on the site. The Grasshopper point parameter was then set to that point. The point parameter is located in the upper left corner of the Grasshopper window. Geometric operations were then developed using the point parameter as a reference. The example in figure 12 used two discrete geometries, an ellipse and a rectangle, as alternatives, which was specifically identified in the project parameters as an option for the students to pursue. In the example provided, number sliders were used to drive the dimensions of either the rectangle or the ellipse geometry, depending upon which geometry the designer was exploring, as well as the operations the geometry is piped into. These operations included vertical distribution (to emulate floor-to-floor as well as overall building height), "series" operations (which generated numbers of floor plates and plate rotation angles), extrusions, and curve and surface offsets, each of which had dimensional parameters driven by shared or independent number sliders.

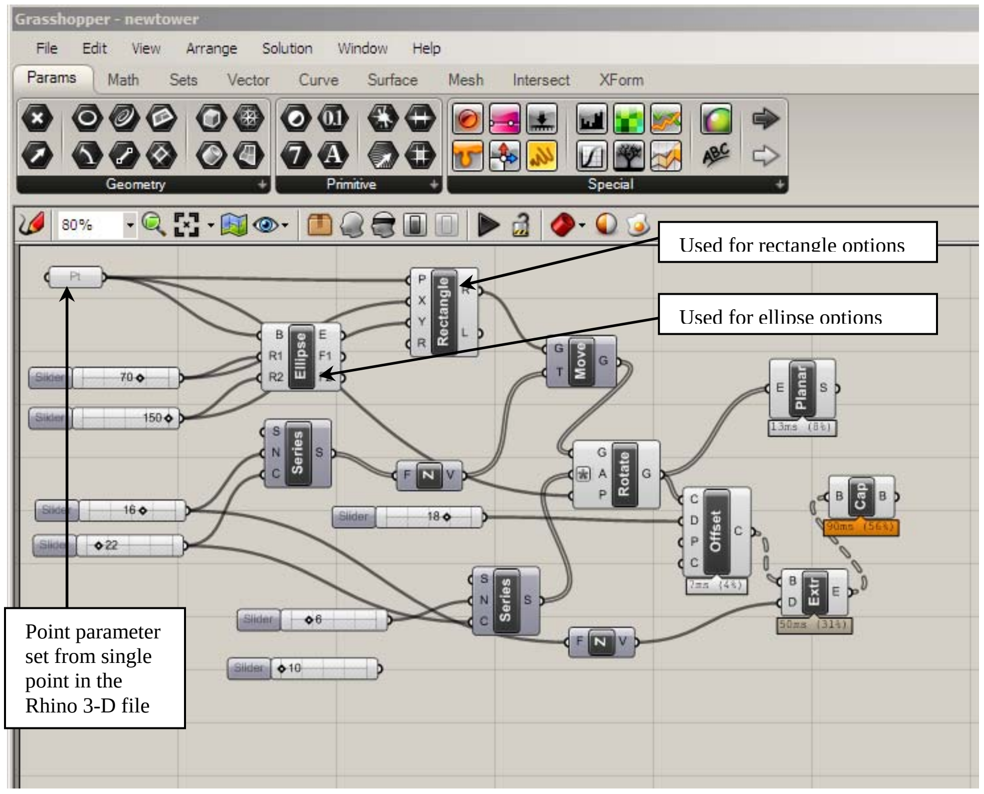

Grasshopper - newtower
File Edit View Arrange Solution Window Help
Params Math Sets Vector Curve Surface Mesh Intersect XForm
Geometry
Primitive
Special
80%
Used for rectangle options
Used for ellipse options
Pt
Rectangle
Ellipse
Move
Planar
Series
Rotate
Offset
Cap
Extr
70
150
16
22
18
6
10
Point parameter set from single point in the Rhino 3-D file

Figure 12. Grasshopper interface showing input and output operations based on a single point.

Figure 13 illustrates the output of explorations involving dynamically manipulating the dimensions and the rotations of the option using rectangular geometries developed in Grasshopper based upon the single point in the Rhino 3D file.

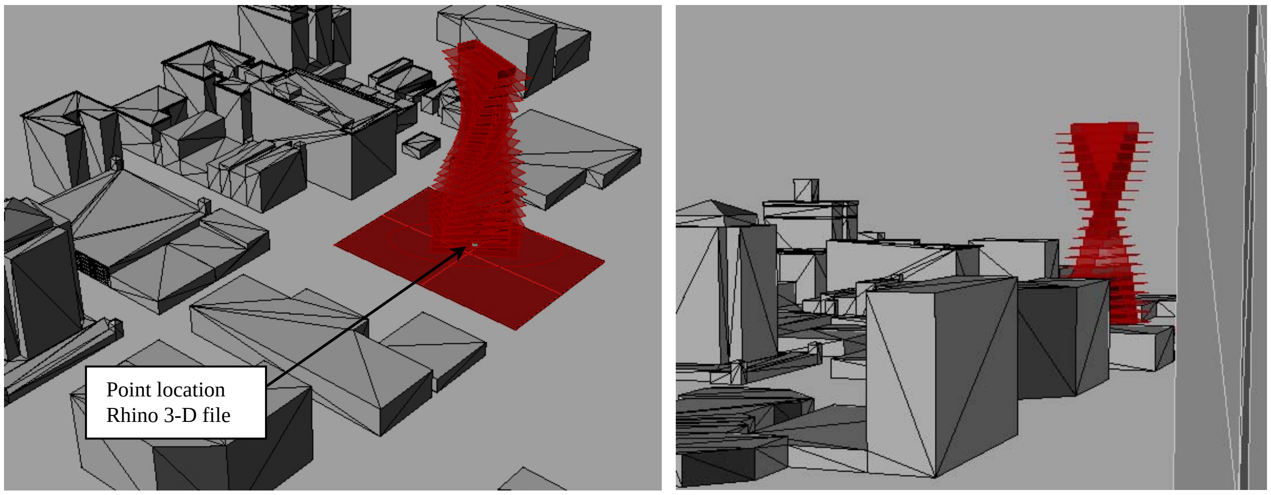


Figure 13. Output of Grasshopper operations illustrated in figure 12.

Other design explorations worked with more traditional massing strategies. In one such example, a student developed a proposal with three tower-blocks, each of which used a formula for the heights and taper angles so that a minimum number of sliders would manipulate the heights and angles using formulas in the expression editors of the several operations. This established relationships between the tower blocks. For example, the height of the central core tower was always a multiple of 2.75 times the height of the lowest tower. Therefore, the height of the lowest tower drove the heights of the other two based on formulas. Additionally, this proposal used a Grasshopper Move X-form to position the overall tower massing relative to the site, and to rotate the orientation of the massing relative to the property lines. Number sliders were used to drive the massing location positively or negatively relative to the X and Y coordinates of the center of the site. This allowed the designer to quickly explore massing options at any location on the site. Three alternatives generated with this proposal are illustrated in figures 14, 15, and 16 in the appendix.

Discussion and Summary

The success of the students in developing design proposals using Grasshopper clearly indicated that the strategy for introducing the software and associated modeling processes, including the strategy of building on prior skills and knowledge base, was effective. However, while developing software knowledge and skills was an intended outcome, faculty were primarily interested in how the students explored design options as they developed and finalized their schemes. Studio observations indicated that a common strategy was to place multiple geometry operators in the Grasshopper window, and then pipe a sequence into those multiple operators and toggle the display of the alternatives off and on to view the options. Additionally, students quickly found that those alternatives could be left in place while further design experimentations were developed "downstream." Most importantly, it became evident that for most students the experimentation with generative design operations yielded unanticipated geometric outcomes, thus enhancing its use as a mechanism for exploration.

Once the geometric relationships were established, generating representations of design alternatives proved to be one of the least challenging. The process of "baking" design alternatives and storing them via layer management was easily mastered by all the students. Interestingly, all student submissions contained at least three additional baking operations on student-defined layers, and in some cases the number was as high as 15, thus indicating a level of design experimentation that was consistent with that observed in the studio.

As noted previously, the class used for this experiment was an elective course. The students enrolled in this course had a high-level of interest in computing in architectural design and would be inherently motivated to utilize new digital technologies. While it was anticipated that this would have a positive influence on the outcomes of the course activity, the extensive design exploration documented in the submissions and in studio observations indicated that the use of Grasshopper and similar generative design tools should be further developed in the curriculum.

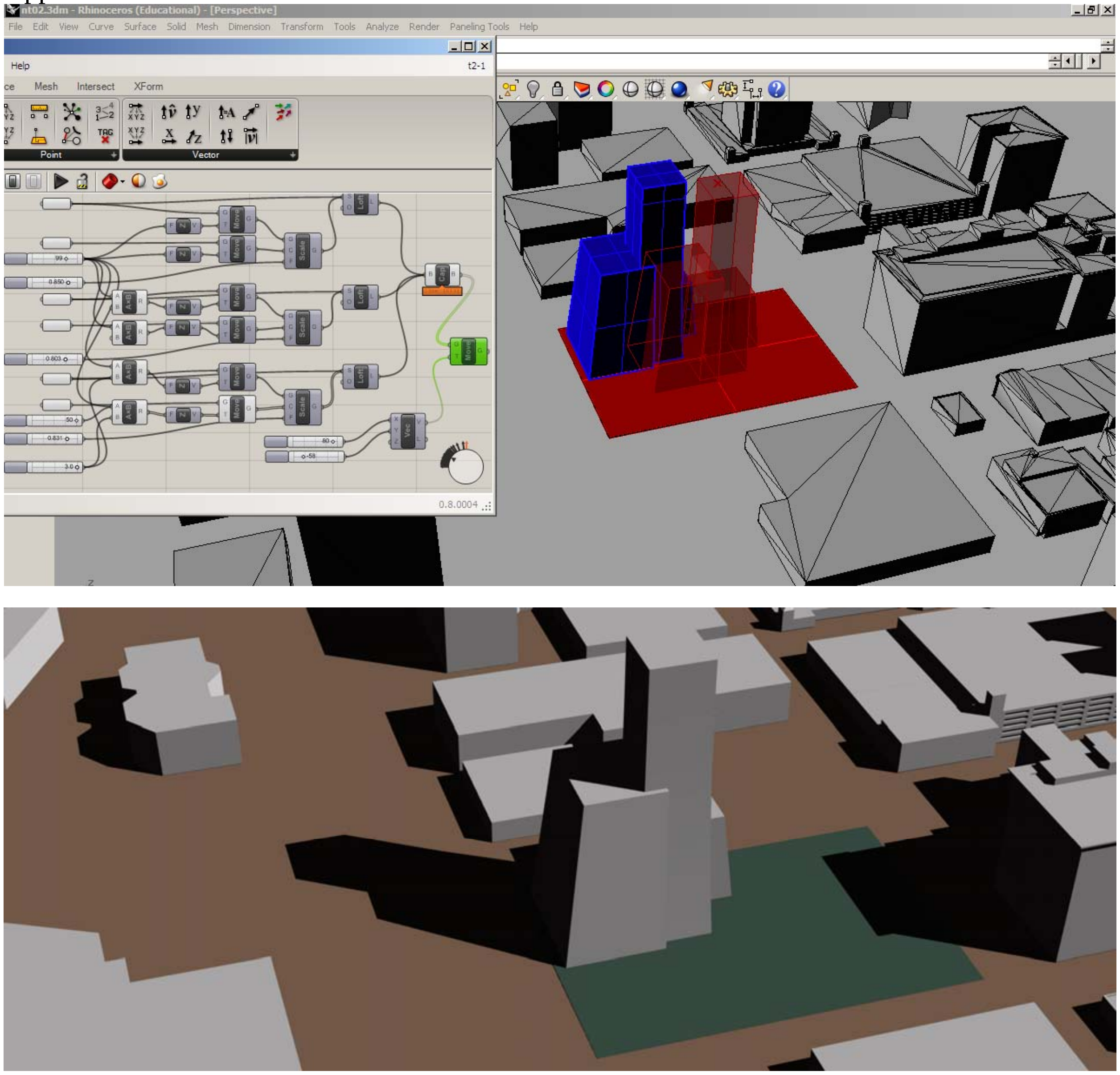
However, despite the potential of generative design processes in both education and practice, a degree of caution must be maintained. The strategy utilized in this case study was based on leveraging students prior CAD and computing knowledge in order to introduce higher-order modeling skills along with the generative design concepts. Introducing generative design concepts into the broader curriculum could exacerbate the lack of digital design skills prevalent among many senior faculty in architecture programs. More importantly, the ability to easily generate complex geometry may potentially inhibit the development of design skills, particularly in terms of the use of these technologies by novice designers. According to Brown [19] "the seductive nature of architectural forms that CAD systems can now produce can lead designers to focus on the exploration of form early in the design process, through particular types of visualization" but added that "this can be at the expense of the technological, financial and social constrains that should be balanced with such investigations of form." Further he suggested that "in discovering this new digital art it is easy to forget the science, or lose the connection between the two; and lose track of the fact that architecture is richer if it addresses both art and science with equal respect."[18] Generative design processes present a true paradigm shift in the architectural design process in terms of both the relevance of traditional media as well as existing applications of digital media. Therefore, faculty must pay close attention to where generative design is introduced into the curriculum and to the skill sets of faculty who will be charged to guide students in its utilization.

References

1. Oxman, R. (2006). Theory and design in the first digital age. Design Studies Vol. 27 No. 3, 229 – 265.
2. Mitchell, W. (2005) Constructing complexity. In "Computer Aided Design Futures 2005". B Martens and A. Brown (eds.) (41 -55). Dordrecht, The Netherlands: Springer Publications
3. Goel, V. (1995). Sketches of thought. Cambridge, MA: MIT Press.
4. Won, P. (2001). The comparison between visual thinking using computer and conventional media in the concept generation stages of design. Automation in construction. 10 (1), (25-35).
5. Suwa, M. and Tversky, B. (1997). What do architects and students perceive in their design sketches? A protocol analysis. Design studies, 18 (4), (385 – 403).
6. Schweikardt, E., & Gross, M. (2000). Digital clay: deriving digital models from freehand sketches. Automation in construction, 9 (1), (107 – 115).
7. Van Elsas, P. & Vergeest, J. (1998). New functionality for computer-aided design: the displacement feature. Design Studies (19), 1 (81 – 102).
8. Leglise, M. (1995). Art under constraint – preserving the creative dimension in computer-aided architectural `
9. Robert C. Zeleznik, R., Herndon, K., and Hughes, J. (1996) SKETCH: An Interface for Sketching 3-D Scenes. SIGGRAPH '96 Proceedings of the 23rd annual conference on Computer graphics and interactive techniques
10. Khemlani, L. (2004). The Rise and Fall of Autodesk Architectural Studio. AECbytes Newsletter #13 (September 9, 2004). Accessed online at http://www.aecbytes.com/newsletter/2004/issue_13.html
11. Yi-Luen Do, E. (2002). Drawing marks, acts, and reacts: Toward a computational sketching interface for architectural design. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 16, 149–171.
12. Schweikardt, E., & Gross, M. (2000). Digital clay: deriving digital models from freehand sketches. Automation in construction, 9 (1), (107 – 115).
13. Van Elsas, P. & Vergeest, J. (1998). New functionality for computer-aided design: the displacement feature. Design Studies (19), 1 (81 – 102).
14. Shesh, A., and Chen, B. (2004). SMARTPAPER: An interactive and user-friendly sketching system. In Eurographics 2004, M.-P. Cani and M. Slayer, eds. 23 (3). 301-310
15. Negroponte, N. (1970). The Architecture Machine: Toward a More Human Environment, MIT Press, Cambridge, Mass.
16. Kolaravec, B. (2003). Architecture in the Digital Age: Design and Manufacturing. Spon Press, New York.

17. Herr, C. and Kvan, T. (2007). Adapting cellular automata to support the architectural design process. Automation in Construction 16 (2007) 61 – 69.
18. Chase, S. (2005). Generative design tools for novice designers: Issues for selection. Automation in Construction. Volume 14, Issue 6. 689-698.
19. Brown, A. (2003). Visualization as a common design language: connecting art and science. Automation in Construction 12 (2003) 703– 713

Appendix


nt02.3dm - Rhinoceros (Educational) - [Perspective]
File Edit View Curve Surface Solid Mesh Dimension Transform Tools Analyze Render Paneling Tools Help
Help
Mesh Intersect XForm
Point
Vector
Move
Scale
Loft
Vec
0.8.0004
Figure 14. Massing option alternative for tower-block scheme with rendering of the geometry after alternative was “baked”.

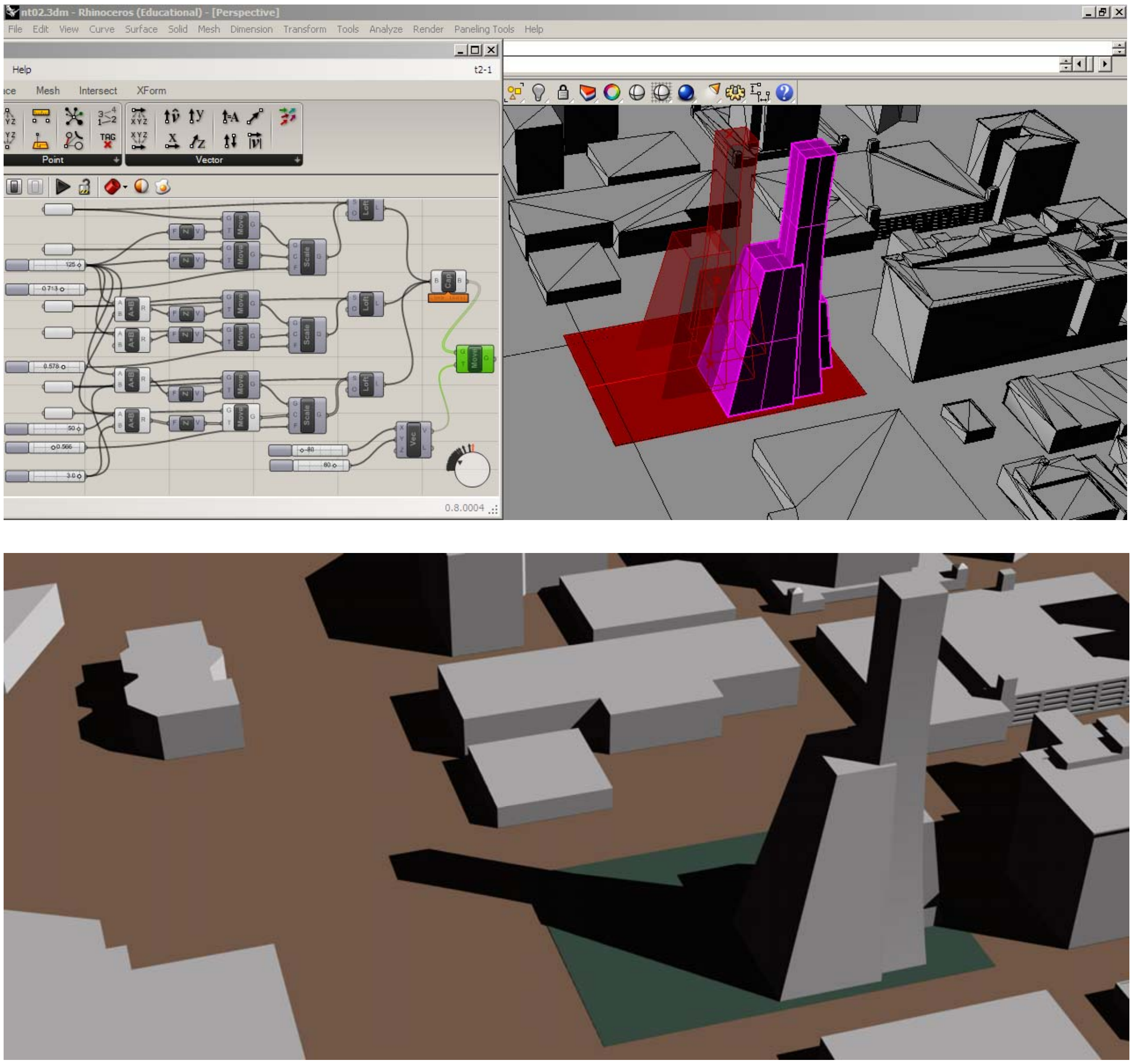

nt02.3dm - Rhinoceros (Educational) - [Perspective]
File Edit View Curve Surface Solid Mesh Dimension Transform Tools Analyze Render Paneling Tools Help
Help
t2-1
Mesh Intersect XForm
Point
Vector
0.8.0004

Figure 15. Massing option alternative for relocated tower-block scheme with rendering of the geometry after alternative was “baked”. Geometry adjusted for repositioned massing.

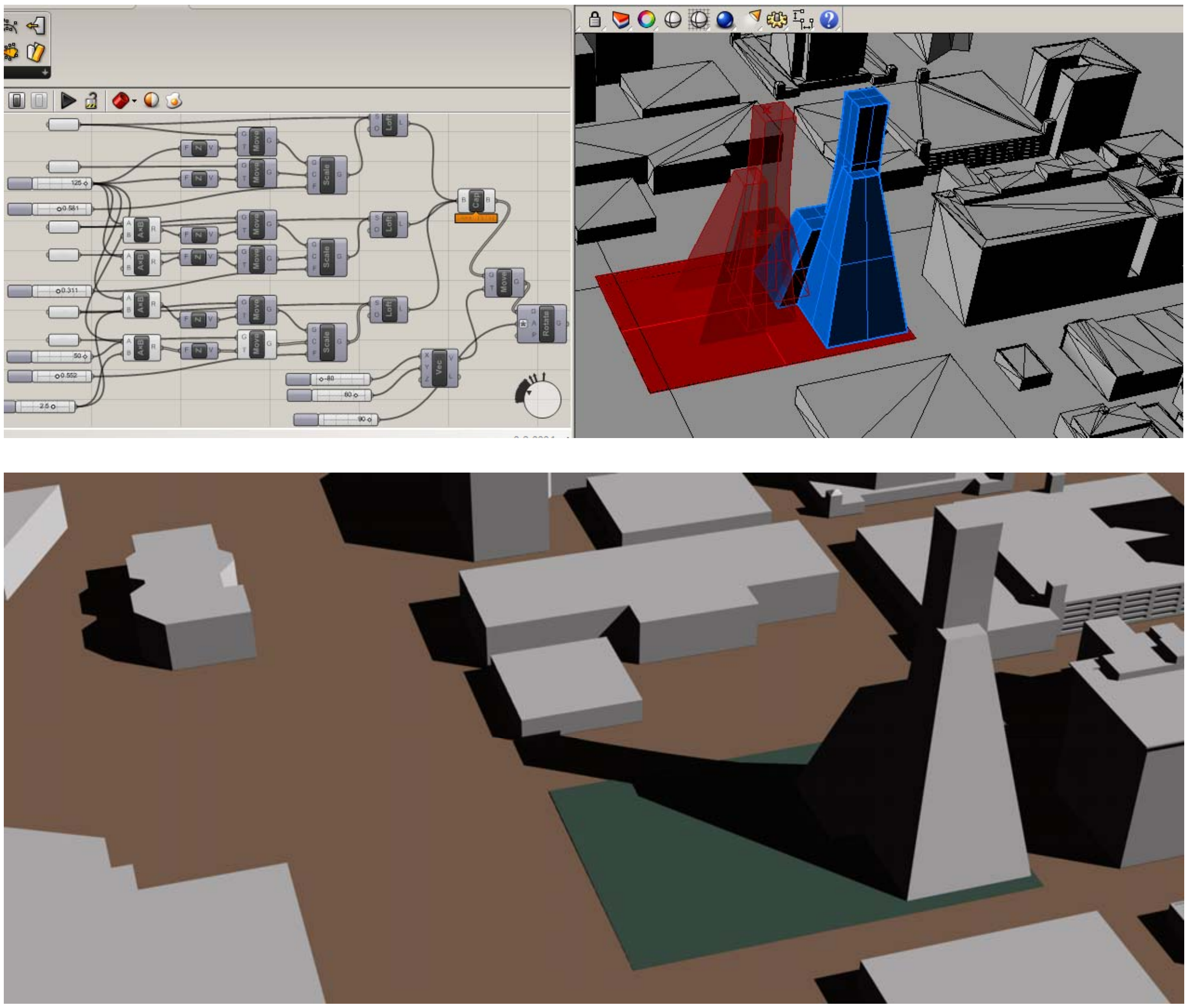


Figure 16. Massing option for rotated alternative for tower-block scheme with rendering of the geometry after alternative was “baked”. Geometry adjusted for repositioned massing.