VISUAL PROGRAMMING FOR BUILDING INFORMATION MODELING: ENERGY AND SHADING ANALYSIS CASE STUDIES

Karen Kensek[1]

INTRODUCTION

Although visual programming is being broadly implemented in other disciplines, it has only relatively recently become an important supplement to three-dimensional modeling programs in the architecture, engineering, and construction industry. Currently, Grasshopper in conjunction with Rhino is a leading example of a visual programming environment that is strongly supported by a user community that is developing additional functionality, but Grasshopper does not yet work directly with building information modeling (BIM) software. Dynamo is relatively new, but shows considerable promise in becoming a constructive tool to complement BIM, 3D modeling, and analysis programs because it includes parametric geometries and works with Revit, a leading BIM software program. Three case studies are described: extensibility of Dynamo through the use of a building energy simulation package, controlling a virtual model's response through light level sensors, and interactively updating shading components for a building facade based on solar angles. They demonstrate that one can work directly within building information models (BIM) using a visual programming language through updating component parameters. These case studies demonstrate the feasibility of a workflow for sustainable design simulations that is different than that more commonly used -- having a separation between design and analysis models and using a neutral file format exchange such as IFC or gbXML to transfer data. As visual programming languages are still a bit uncommon in the building industry, a short background is provided to place them within the tool set of other customizable tools that designers have been developing.

KEYWORDS

visual programming language, VPL, building information modeling, BIM, Dynamo, Revit, Grasshopper, computational design

1. University of Southern California, School of Architecture, Watt Hall #204, Los Angeles, CA 90089-0291, kensek@usc.edu, 213-740-2081

BACKGROUND

Key terms

To place this paper in context, it is beneficial to understand a few key terms: building information model (BIM), building energy modeling (BEM), interoperability, and visual programming languages (VPL).

A *building information model* is a 3D model created from parametric objects (e.g. wall, door, window, roof) that contain data about the objects (e.g R-value, hardware set, cost, slope). This is fundamentally different from other 3D modeling programs whose basic component is a surface or solid (e.g. Rhino 3D) that does not have associated data. A BIM can be used throughout the life-cycle of a building, but the information that is useful is different for the various stake-holders. For example, an architect might use the BIM to produce conceptual designs, working drawings, preliminary cost estimates, and door schedules. A consultant might use it as the basis for energy simulation or structural calculations. Contractors find it useful for clash detection, construction sequencing, and detailed cost estimating, and often facilities managers can use the information for tracking assets in a completed building. There are several common building information modeling programs: Revit, ArchiCAD, VectorWorks, Digital Project, AECOsim Building Designer, among others.

Building energy modeling is a general term for hundreds of software programs (from very simple to incredibly complex) that are used for energy simulation of buildings, often for calculating heating and cooling loads, but also thermal comfort and HVAC sizing and control. A few common software programs are DOE 2, EnergyPro, DesignBuilder, eQuest, HEED, Sefaira, Green Building Studio, AECOsim Energy Simulator, but there are many choices depending on one's specific needs.

Interoperability refers to the ability to move data from one program to another. The data could be a 3D model; file formats such as DWG, SAT, DXF, and many others can be used to import and export raw geometry (like surfaces and solids) from one 3D program to another, usually successfully. It is more difficult to transfer BIM data. Two file formats that are commonly used are gbXML that is used for the transfer of the geometry and energy data and IFC (Industry Foundation Classes), which is in use and under further development to provide an open source method of transferring "all" types of BIM data. Neither technique is fool proof, and some firms have created custom methods of translating data that fits their own workflow. Other workflows include maintaining separate models for design and for simulation. This lessens potential advantages of using BIM data throughout the design/engineering/construction/occupancy process.

Visual programming languages use graphics (such as blocks and wires) to create a software program. Essentially the flow diagram is the program. Examples of visual programming languages include Simulink, Grasshopper, Dynamo, Marionette, and others. There user interfaces are quite different from text based programming languages where the code is typed into a text editor or a more sophisticated code editing program, for example BASIC, LISP, C#, Python, Java, Processing, etc.

Workflows for energy simulation

Three major workflows are often used by designers when studying sustainable options for their buildings:

1. maintaining two separate 3D models, one for design and another for energy simulations.
2. transferring the 3D model to the energy software via formats such as gbXML or IFC.
3. using a plugin within the 3D software that connects directly to the energy simulation software.

Hybrid variations also exist.

For example,

1. There could be a separate Rhino file and an EnergyPro file. The designer updates each of them separately, and there is no flow of data between them.
2. An IFC or gbXMLfile could be exported from ArchiCAD to DesignBuilder for simulation with (hopefully) a set of complete and accurate data.
3. IES <VE> or Sefaira can be accessed via a plugin (a small program for a specific purpose) in Revit.

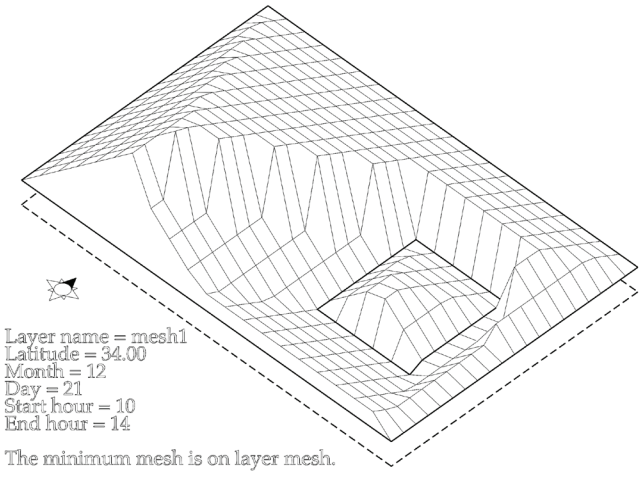
Each of these has its own set of advantages and disadvantages that effect the turn-around time of design decisions, the re-entering of data, and which software one can actually use.

A fourth workflow, still relatively uncommon, is to associate the 3D model to energy software within a visual programming environment that could allow for more direct passing of data from one simulation engine to another, even those written by different developers, in a common working environment.

Customization of existing software

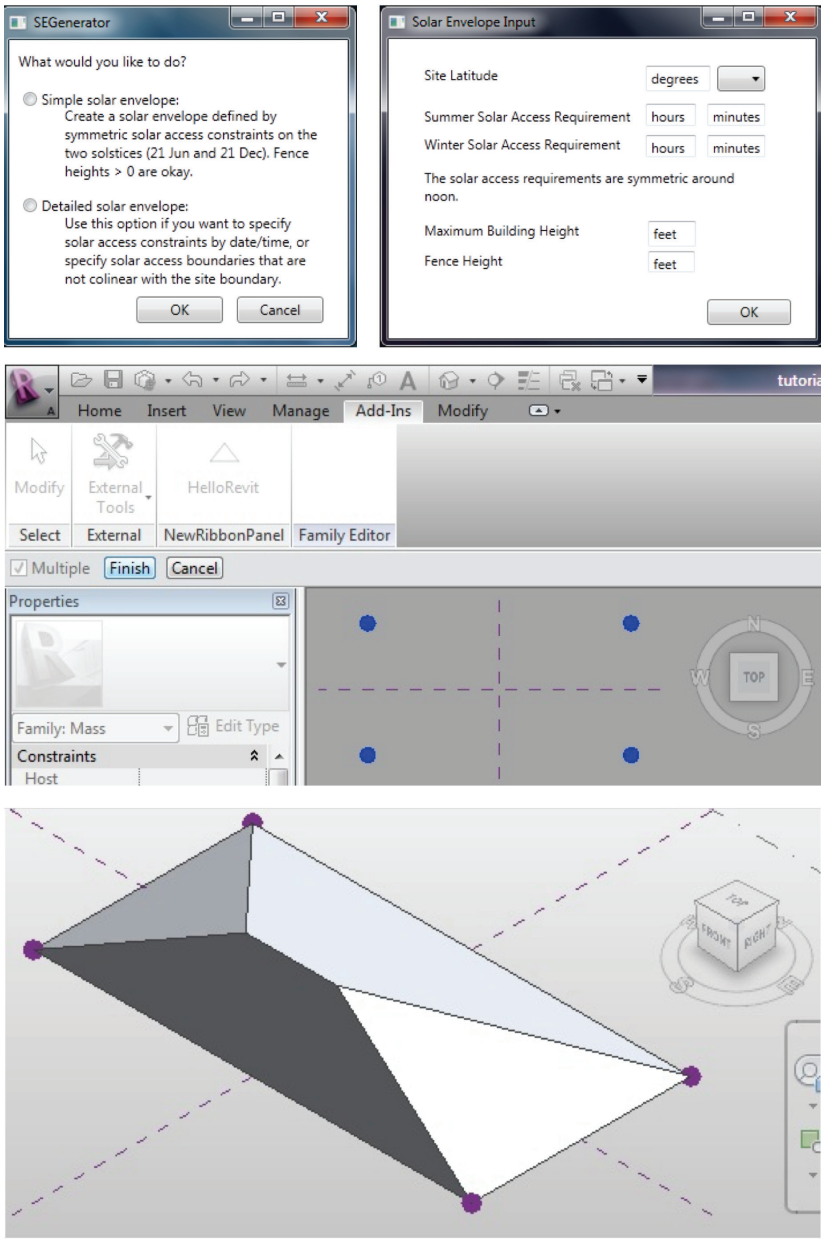
Users need to have the ability to customize existing software so that they can work more effectively in their discipline specific applications. For example, early AutoCAD software was released with AutoLISP capabilities that allowed for the development of scripts to automate time-consuming tasks and add new capabilities (Fig. 1). Text based scripting languages have not disappeared and are still in use, including examples such as Python, Rhino Script, MEL, and MaxScript. In addition, computer software is supplemented with application programming interfaces (API) that allow developers and other skilled users to access the internal guts of the software to create new sub-programs or plug-ins (Fig. 2, Fig. 3).

FIGURE 1: Solar envelope generator program showing some of the AutoLISP code. (Kensek and Knowles 1997)




```
; cosalt is the cosine of the altitude
; alt is the altitude in degrees

(setq latrad (DTR lat))
(setq declrad (DTR decl))
(setq hranglerad (DTR hrangle))

(setq sinalt (+ (* (cos latrad) (cos declrad) (cos hranglerad))   (* (sin latrad) (sin declrad))))

(setq sin2alt (* sinalt sinalt))
(setq cosalt (sqrt (- 1 sin2alt)))

(setq alt (RTD (atan sinalt cosalt)))
(print "Altitude:  ) (print alt) (print)
)

;-------------------------------------------------------------------------------end CalcAlt
;-------------------------------------------------------------------------------CalcAzm
(defun CalcAzm ( / hranglerad declrad latrad xazm yazm tanazm)
; calculates the solar azimuth based on values from CalcInt and CalcAlt

; hranglerad is the hour angle in radians
; declrad is the declination in radians
; latrad is the latitude in radians
; xazm and yazm are the east and north component of the sun vector
; tanazm is the tangent of the azimuth
; azm is the azimuth in degrees

; do not need these if the program is changed to get the values non-locally
(setq hranglerad (DTR hrangle))
(setq declrad (DTR decl))
(setq latrad (DTR lat))

(setq xazm (* (sin hranglerad) (cos declrad)))
(setq yazm
    (+ (* (- (cos hranglerad)) (cos declrad) (sin latrad)) (* (cos latrad) (sin declrad))))
(setq azm (RTD (atan xazm yazm)))

; gives the proper angle for convention N=0, + CW
(cond
    ( (< azm 0) (setq  azm (+ 360 azm)))
)

(print "Azimuth:  ) (print azm) (print)

)

;-------------------------------------------------------------------------------end CalcAzm
;-------------------------------------------------------------------------------CalcSolar
(defun CalcSolar ( )
; CalcSolar calls CalcInt, CalcAlt, and CalcAzm; InputVar must be run before
; calculates altitude and azimuth

(CalcInt)
```

FIGURE 2: Solar envelope plugin written in C# using the Revit API. (Kensek and Henkhaus 2013)



Recently, there has been a trend towards visual user interfaces for programming that allow users to create customized, flexible, and powerful form-generating algorithms without having to first learn how to write code. Visual programming is a type of computer programming where users graphically interact with program elements instead of typing lines of text code. Nodes are created; they can be numbers, sliders for adjusting values, operators and functions, list manipulation tools, graphic creators, scripts, notes, "watch" nodes, customizable nodes, and other types depending on the tool. They are virtually wired together, and the program is resolved from left to right. (Fig. 4).

FIGURE 3: In-progress Revit plug-ins that calculate heat gain of different types of window shading devices (left) and that calculates the size of a rainwater cistern based on location and roof area (left: mage courtesy of USC students Daubert, Harrison, and Reego and right: Barley, LinShiu, and Tucker) (Kensek 2014)

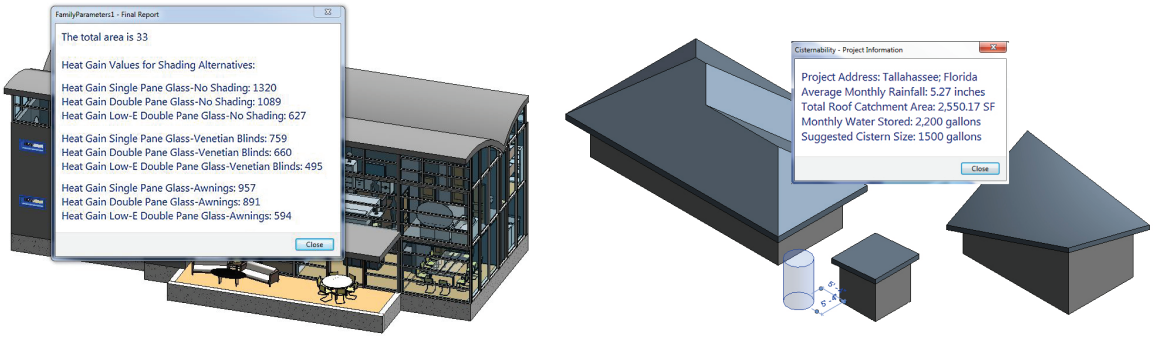
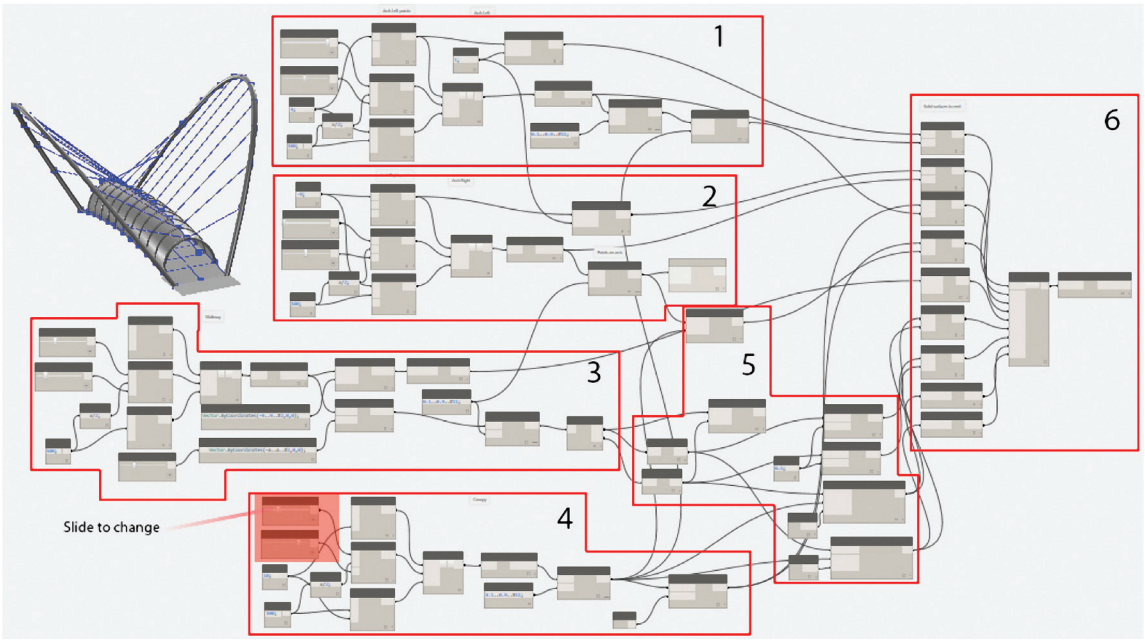


FIGURE 4: Dissection of a Dynamo graph for bridge creation (note the nodes and wiring): 1) left supporting arch; 2) right supporting arch; 3) walkway (slab, points, connecting lines); 4) canopy script; 5) cylindrical shape applied to the lines; 6) exporting geometry to Revit. The red lines were added to make the diagram easier to understand. (image courtesy of USC students Ilyassov and Tazhibayev) (Kron 2014)



The slider nodes for the canopy allow the user to experiment with a range of design solutions (Fig. 5).

Different variations of the bridge are thus produced (Fig. 6). Packages (other sets of nodes with their own capabilities packaged together for download and installation) from other developers can also be integrated for additional features.

And although a trend has been observed in the increasing use of visual programming languages (VPL) in schools of architecture, VPLs compliment rather than replace traditional

FIGURE 5: Canopy script (section 4 of Fig. 3) is used to interactively change the size and shape of the canopy. (image courtesy of USC students Ilyassov and Tazhibayev))

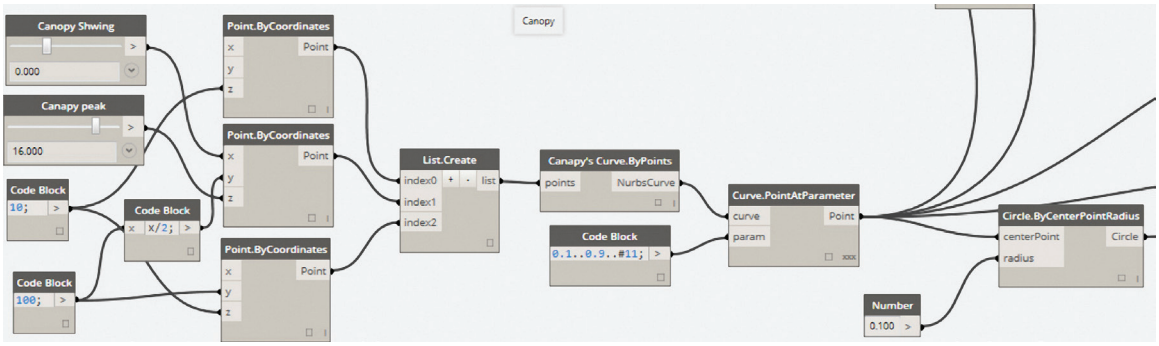
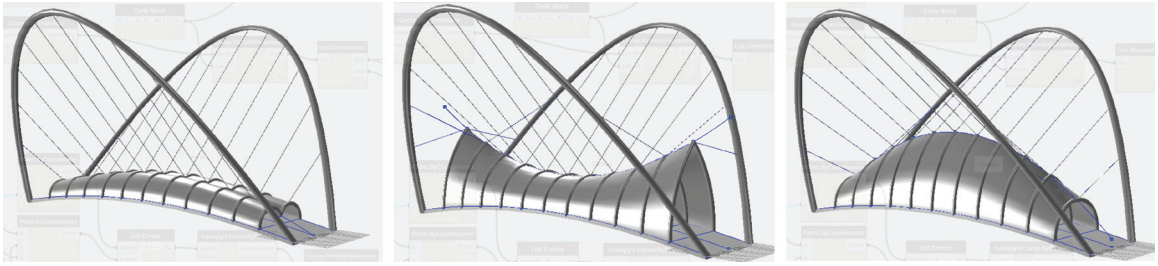


FIGURE 6: Three iterations of the canopy parameter. (images courtesy of USC students Ilyassov and Tazhibayev)



text-based programming languages (TPL). In a recent study, both scripting and visual programming were taught to a set of architecture students with no computing experience, and it was determined that for beginners, VPL lead to better results that tended to only explore parametric variations while students using TPLs achieved more complex implementations (Celani and Vaz 2011). Some visual programming environments also allow for the use of text-based scripting to combine the advantages of both. Research efforts are also underway that provide mechanisms for teaching text-based programming through the use of visual programming (Wurzer and Pak 2012).

VISUAL PROGRAMMING FOR ARCHITECTURE, ENGINEERING, AND CONSTRUCTION

"Many new ways of generating parametric models have been developed … The commonality of all these parametric modelling environments is the ability for designers to modify parameters and relationships that trigger the transformation of related model parts. This is now a popular way to create and modify digital models" (Davis 2013). Davis goes on to develop his own interactive textual programming environment (Yeti) that continuously evaluates the code while generating the geometry.

Generally, visual programming environments do not typically exist within commercially available building information modeling applications. The popular Grasshopper plug-in to Rhino is free, powerful, and has a large group of users who are actively expanding its capabilities. However, it is used exclusively with Rhino, which is a surface modeling program. The

resultant graphic data can be transferred to BIM software, but this is often problematic due to different file structures and geometric definitions.

Dynamo was developed as a plug-in for Revit, a BIM software program to provide a visual programming environment. It can also be used with other software programs although that is currently not widely implemented in practice. In addition, bridges between the Rhino/Grasshopper environment and Revit/Dynamo are being created. For example, Rhynamo allows the resultant geometry and data associated in it to be brought into Dynamo (Miller 2014). Overall though, portability between the environments is still a problem with transferring geometry and data and reusing programming libraries (Leitão and Santos 2011).

Grasshopper

Explicit History (the first version of Grasshopper) was released in 2007 as an add-in to Rhino 3D by David Rutten at Robert McNeel & Associates. Since its original inception, the functionally of Grasshopper has grown tremendously with the development of third-party plug-ins including environmental simulation components that allow for solar studies, input into energy engines, and optimization. A number of components have been developed by outside developers for a myriad of uses; for example, many have been developed to automate environmental calculations: visualization of weather data, daylight and energy simulation, optimization, or other tasks related to energy simulation. A partial list of these tools includes ArchSim, Gerilla, and Ladybug + Honeybee that links to EnergyPlus; DHour for visualizing weather files; DIVA for daylight and energy modeling; Geco that links to Ecotect; Heliotrope for sun angles; Mr. Comfy for visualization of thermal and climate data, and Tortuga a global warming potential evaluator. Visual programming tools can make studying sustainable design strategies easier at the early stage of design. There are other components for Grasshopper for optimization (e. g. Galapagos and Octopus) that are useful for energy and lighting calculations. The ecosystem growing around Rhino and Grasshopper is continuing to grow in size and sophistication (Fig. 7, Fig. 8).

FIGURE 7: Diagram of Grasshopper canvas sub-sections for complex thermal and daylighting optimization. (image courtesy of USC student Gamas) (Gamas et al. 2014)

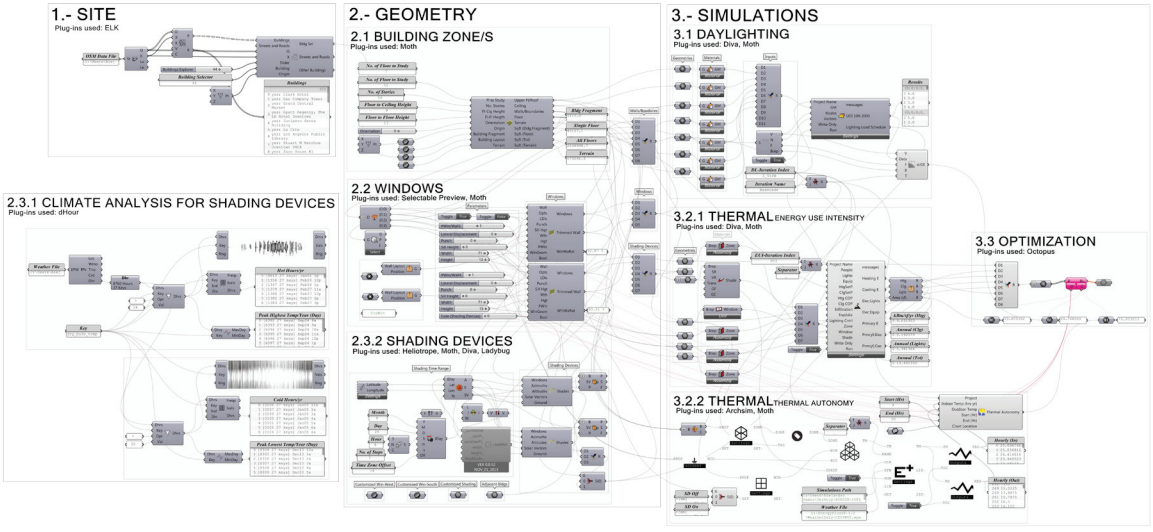
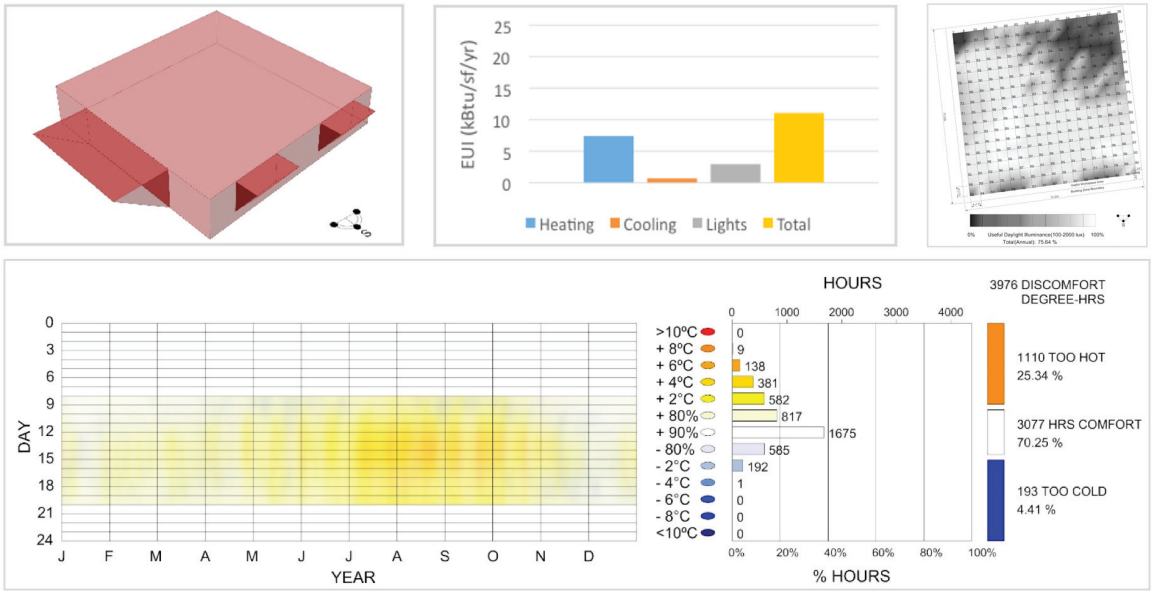


FIGURE 8: One run from the study shown in Figure 7 seeking to optimize EUI and thermal autonomy (for the 51st floor southwest corner with a window-wall ratio of south 34%, west 53%) using Rhino 3D, Grasshopper, DIVA, Dhour, Heliotrope, Ladybug, ArchSim, and others. (image courtesy of USC student Gamas) (Gamas et al. 2014)



Grasshopper is specifically mentioned because it has a strong set of tools available and an active user base. However, it can only be used with Rhino 3D, which is not building information modeling software. Also, although the Rhino/Grasshopper environment is currently more deeply developed along these lines than Revit/Dynamo, the latter does have features that are distinct from the former such as the existence of parametric objects. (Note that another VPL called Marionette has just been released.)

Dynamo

"Dynamo is a plug-in for Revit and Vasari that displays a graphical interface for adding and adjusting parametric functions of BIM components" (Ogueta 2012). Autodesk Revit is a widely used BIM program, and Dynamo was originally written by Ian Keough as a plug-in to Revit using the Revit API and the Windows Presentation Framework (WPF). In the early 2000's, Autodesk started participating heavily in the development of Dynamo and by 2015 had released Dynamo 0.8.0 as an OpenSource program. It is evolving rapidly. As of August 29, 2014, there were nearly 25,000 downloads of more than 350 packages (custom nodes and plug-ins) by under 100 developers. By May 21, 2015 those numbers had reached almost 75,000 downloads and 550 packages by more than 500 authors. Dynamo is in an explosive stage of growth.

Ian Keough recently explained the principles behind the development of Dynamo (webinar, May 8, 2015) including these three:

1. Users should be able share tools. This is exemplified through the development of packages that users can upload.
2. Every user should be able to use Dynamo. For example, a client might just want to be able to view a model and manipulate key parameters interactively or a programmer in the office might develop a new tool for streamlining the workflow in the office.

3. Users should be able to do real-time analysis so that they can use the feedback immediately for real-time design decisions. Simulation of environmental related concerns is an excellent example where this would be useful.

2.4 Revit parameters

One key difference between Rhino and Revit is that Rhino is a surface modeling 3D program, and Revit is based on parametric objects like walls, doors, floors, and custom components that it calls families. For example, a door has parameters of width, height, sill height, manufacturer among others, and custom parameters can be added. Dynamo allows for the creation of 3D geometry in the Dynamo environment ("graph"), but also lets the users access for viewing and for editing the values of the parameters of a family in the Revit modeling environment.

CASE STUDIES

All three case studies focused on the use of Dynamo for studying environmental design issues, not just for creating parametric geometry. The visual programming workflow can be especially useful in the early stages of design when some (but not all) of the constraints are known about the building, leading to a "design space" conducive to exploration. Jabi found that he had to develop a new tool that that would "more closely harmonise the outputs of parametric digital spatial representations with the input requirements for building performance simulation," but hoped that Dynamo/Revit could also provide this functionality if his requirements of non-manifold topology were met (Jabi 2014).

Dynamo can access directly sun path data in Revit. In addition, with the use of the custom package from Thornton Tomasetti, "Energy Analysis for Dynamo," it can link to conceptual geometry and provides custom nodes for the input of data for Green Building Studio. The first case study demonstrates this. The second and third examples demonstrate the applicability of parameters where the custom parameters of shading devices are updated based on stimuli.

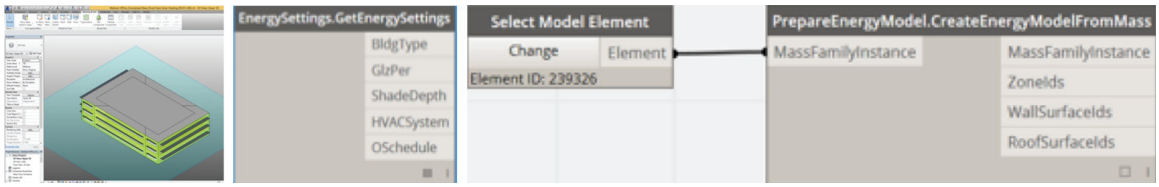
Case Study One: "Energy Analysis for Dynamo" Package

Although the functionality of Dynamo is growing very quickly, the environmental simulation components available for Dynamo are almost non-existent both for passive and non-passive calculations. However, there are some tools. One is an academic research implementation of a non-dominated sorting genetic algorithm to optimize daylighting and energy use (Asl et al. 2014). The plug-in used for the following example was Thornton Tomasetti's CORE Studio's "Energy Analysis for Dynamo" package (beta version was released on October 22, 2014). It provides a link between Revit and Green Building Studio from within Dynamo by connecting Revit's conceptual modeling environment with many of the core features of Green Building Studio. Technically, conceptual masses are not driven by parameters in the same method that the fully detailed building model components are in Revit, but this example instead demonstrates the extensibility of Dynamo through outside programs.

For this workflow, Revit 2015 and Dynamo 0.7.2 were used to compare the energy consumption of a building with and without window shades (Fig. 9) (Konis and Kensek 2015). The first set of analysis was done changing only the shading depth and assuming double pane clear glass, lightweight construction with a typical mild climate insulation, and a cool roof with typical insulation. The results of these analysis have than been compared to each other in

order to find the configuration for which the yearly fuel cost is the minimum available in that particular site for that particular building. The energy settings available in the node are building type, glazing, shade depth, HVAC system, and operating schedule.

FIGURE 9: Window shading turned on (left). Energy setting variables (middle). Link of Revit model to energy settings (right). (images courtesy of USC students Gao, Milian, Toldo)



The energy settings available in the node are Building Type, Glazing, Shade Depth, HVAC System, and Operating Schedule. Changing the values of the two overhang sliders allows to set different values for north/south and east/west facades.

The mass was exported to Green Building Studio for the energy analysis. A summary of the results shows that shading devices save money in fuel cost (Table 1). Twenty-two analyses were carried out, and the results were compared both with the use of Dynamo and without.

Table 1: Simple Building with and without Shading Devices

Glass Material	Double Pane Clear – No Coating	
Exterior Walls Material	Lightweight Construction – Typical Mild Climate Insulation	
Roof Material	Typical Insulation - Cool Roof	
Shade depth	**Analysis_001**	**Analysis_020**
North facade	0.0	0.0
South facade	0.0	0.8
East facade	0.0	1.8
West facade	0.0	3.0
Annual Fuel Cost	**$1900**	**$1374**

The simulation results are not shown directly in Dynamo, but must be accessed in Green Building Studio. In addition, because the Thornton Tomasetti package used the Revit conceptual mass instead of the detailed building model, parametric values in the 3D model were not used. The next two case studies show how parameters can be directly accessed from the building information model and used within the visual programming environment.

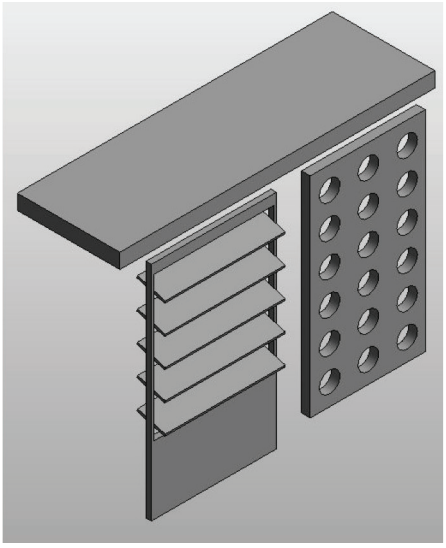
Case Study 2: Dynamo linking photoresistor value with 3D model

This example is from a previously published research project where the brightness of light registered by a photoresistor on an Arduino board caused a Revit model to change (Kensek 2014). It did this by indirectly updating the parameters on three Revit families: the size of the holes in the dynamic panel, the rotation of the louvers, and the length of the overhang on the house (Fig. 10).

Lighting levels were adjusted manually to partially block daylight access to the photoresistor, so that the amount of light hitting the photoresistor lessened and its value output

changed from 255 to almost 0. This value was passed from the Arduino board through a processing script to Dynamo that then updated the LightLevel parameter in the components and changed their width, rotation, and radius as applicable. Slowly, the Revit model updated (Fig. 11).

FIGURE 10: Three Revit families: overhang, louvers, and panel and their parameters.



Parameter	Value	Formula
Dimensions		
width (default)	4' 10"	=4' 10" + LightLevel / 50
LightLevel (default)	0' 0"	=
Identity Data		

Parameter	Value	Formula
Dimensions		
rotation (default)	90.000°	=1° + LightLevel / 255 * 89
LightLevel (default)	255.000°	=
Identity Data		

Parameter	Value	Formula
Dimensions		
radius (default)	0' 8 75/256"	=1' 1" - LightLevel / 255
LightLevel (default)	100' 0"	=
Identity Data		

FIGURE 11: 3D model in Revit: light level is 255 (left). 3D model in Revit: light level is 75 (right). (the original house was modeled by USC student Martinez)

The third case study also uses changes to the family parameters to change the components.

Case Study 3: Attractor Exercise and Dynamic Solar Shades

Updating custom parameters was the method used for creating interactive solar shades. The first part of the exercise was to set up a simple Dynamo sketch that change the circumference of circles based on the location of a slider controlled script – the attractor (Fig. 12 and Fig. 13).

FIGURE 12: Attractor sketch in Dynamo. (image courtesy of USC students Hopper and Lohgren)

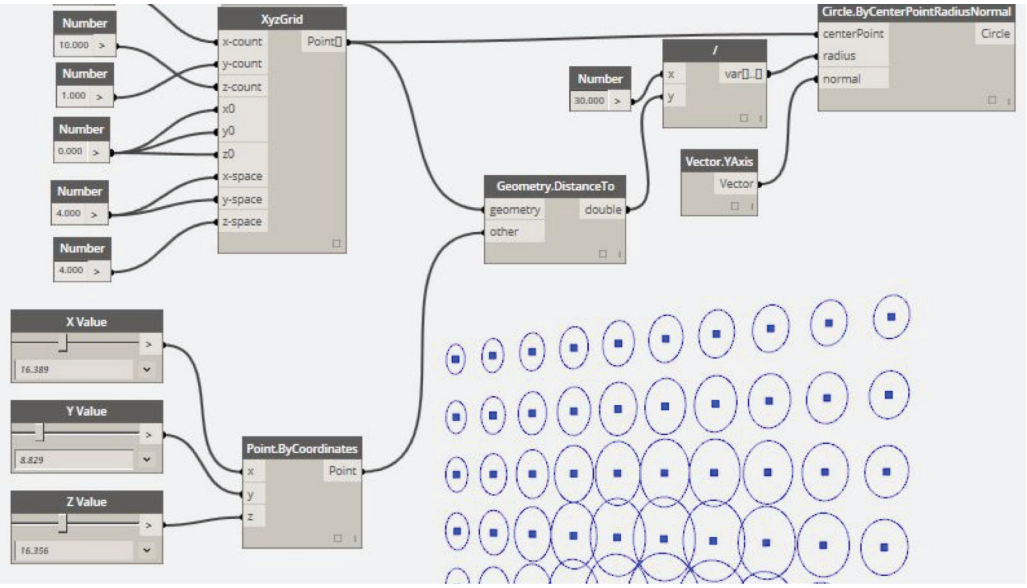
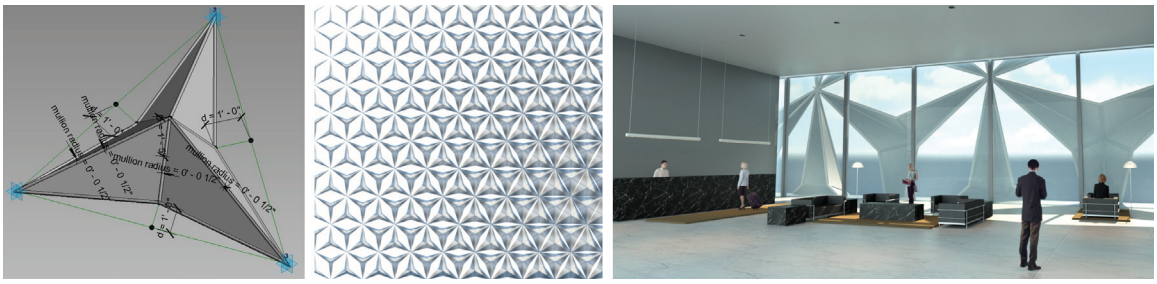


FIGURE 13: Parametric panel responding to the position of the attractor. (images courtesy of USC students Toldo and Chow)



The same technique was used for changing the parameters of Revit families based on the angle between shading device and the sun. After accessing the sun's position in Revit, an attractor script modified custom parameters on façade panels and shading devices based on the sun's position in the sky (Fig. 14).

FIGURE 14: Sun shading device responding to solar position by getting deeper when the sun has a lower altitude. (images courtesy of USC students Dharane and Aljammaz)

The final example not only has the values on the window coffer change dimensions, but also the color of the panel based on the solar altitude (Fig. 15, Fig. 16).

FIGURE 15: Sun shading device parametric family (left) and opening size responding to solar position (right). (images courtesy of USC students Ilyassov and Tazhibayev)

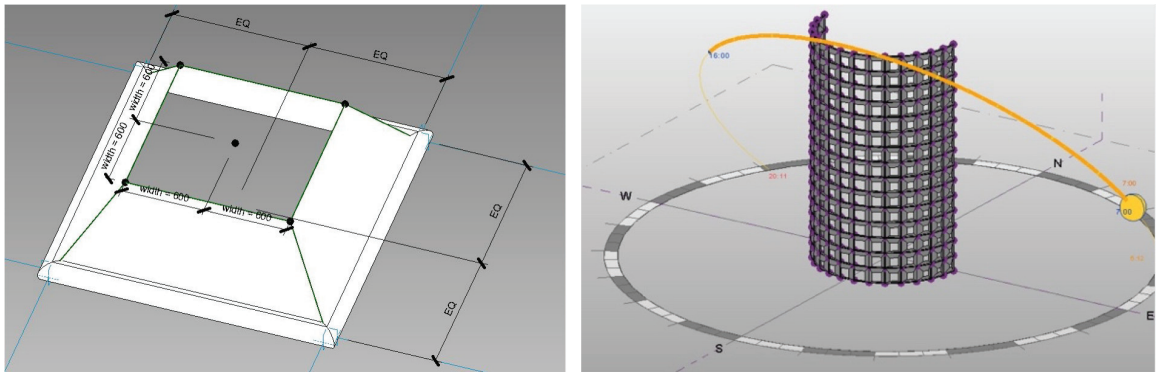
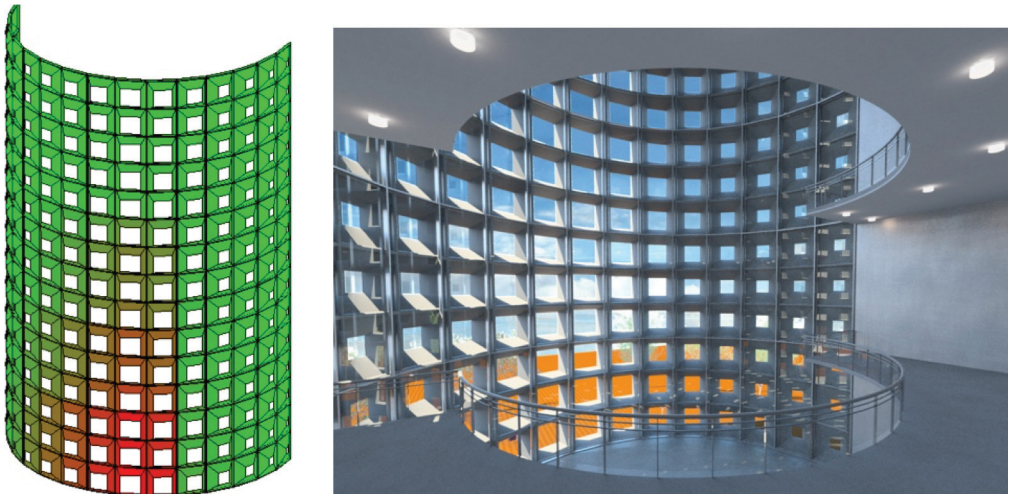


FIGURE 16: Panel changing color in response to solar angle (left) and interior rendering (right). (images courtesy of USC students Ilyassov and Tazhibayev)



Aksamija and colleagues also used parameters for dimension values for more complex solar shading devices based on solar radiation values. Their process involved exporting the model from Revit, calculating solar angles and radiation values in Ecotect, collecting the data in Excel, and using a custom WhiteFeet plug-in to export the data to the solar shades' parameters in Revit (Aksamija et al. 2011). This process could be accomplished within Revit with the use of Dynamo.

DISCUSSION

In the first case study with the "Energy Analysis for Dynamo" package, there were several downsides with using Dynamo especially in comparison with similar workflows completed with Rhino, Grasshopper, and its ecosystem of packages (Konis and Kensek 2015):

- Dynamo only links with Green Building Studio (GBS) and the user still has to go into GBS for any complex changes and visualization of results.

- It only analyzes conceptual masses, not the detailed building model.
- Dynamo and the energy package are not very stable yet.
- There is a comparatively small user base and teaching material.
- There is a lack of optimization tools.

However, a few critical advantages were noticed:

- After Dynamo is set up, the user can easily swap out different building models.
- It works within a BIM authoring tool and conceivably (not tested) could work within other software as it is an OpenSource software program.
- One can easily add sliders for values that are important to a study so that specific parametrics can be controlled during the design process.

In case studies two and three when working directly with family parameters, the biggest problems were

- the lack of consistent nomenclature between Revit and Dynamo about families and parameters.
- stability problems with Dynamo.
- missing nodes and features.
- slow update times. "Interactive programming (also known as live programming) seeks to remove any latency between writing and running code" (Davis 2013). The difficulty faced is that the updating of the Revit families themselves takes time.

Overall, however, the workflow for interacting with a building information model through the use of parametric updates proved to be a feasible method for testing of sustainable design alternatives. Visual scripting offers a potentially more flexible workflow than traditional methods. "The 3D model is one input into the simulation (the weather file would often be a second input). The simulation program is a component node in the overall workflow. Conceptually it is very different from the other workflows conceptualizing the simulation program as a node.

- "there is explicit movement of the data
- a specific parameter can be easily changed to note its effect in the overall simulation results
- the output of the component can be more easily channeled into another simulation node
- potentially, there is a future ability to design more holistically as the tools are visible and more may be used at one time
- optimization algorithms can be applied as part of the workflow
- more tools can be developed that increase the functionally of the workflow as the apps are easier to create than full blown software programs
- there is an expectation that advanced users can create their own nodes
- complex workflows developed for one building can easily be used to study other buildings" (Konis and Kensek 2015)

CONCLUSION

Maleki et al. contend that "the recent history of parametric CAD (at least in building design) can be told as a tale of improving interfaces to the parametric engine, largely through interactions with the dataflow graph," and that "liveness" is critical for improved software (Maleki

et al. 2014). Visual programming, if done well, can provide the immediate reactions that are necessary by providing a link between the geometry and the data. A visual programming language can also provide a bi-directional link between the design tool / model operator and the building performance simulations environment / calculation model (Negendahl 2015). Negendahl concludes that this type of integrated dynamic model can better support the design during the early stages of design.

The use of visual programming languages in AEC extends beyond the focus on energy and shading discussed; applications in generative design development (Miller 2009), landscape architecture (Briscoe 2014), structural performance (Makris et al. 2013), construction detailing (Zarzycki 2012), fabrication (Agrawal et al. 2014), and other aspects of building design, engineering, and construction have been done. Innovative architecture and engineering firms are currently implementing visual programming languages into their workflows, and active users will continue to expand their applicability and functionality. Incorporating this ability within building information modeling is not only possible, but will become increasingly common in both academia and the building profession.

NOTES

The mentioning of different software programs does not imply endorsement of any product. The software names are trademarked by their respective companies.

ACKNOWLEDGEMENTS

Arduino project –W. Kahn, student researcher

Dynamo bridge and shading devices – students in Architecture 507 class (2014 and 2015)

Rhino 3D / Grasshopper and Revit / Dynamo project – K. Konis (co-PI), A. Gamas, G. Gao, J. Milian, I. Toldo, and M. Hijazi, and J.Staller at TRC Solutions, "Leveraging BIM and Scripting for Passive Design Parametric Analysis." Funding was supplied through TRC Solutions by a SoCal Gas IDEEA CSDP grant to fund university research on passive and low energy strategies to assist the non-residential commercial market in achieving sustainability, zero net energy (ZNE), and thermal comfort.

REFERENCES

Agrawal, H., Jain, R., Humar, P., and Yammiyavar, P. 2014. "FabCode: visual programming environment for digital fabrication," *IDC '14*: Proceedings of the 2014 conference on interaction design and children, pp. 353-356, ACM New York, NY, ISBN: 9781450322720 doi>10.1145/2593968.2610490.

Aksamija, A., Guttman, M., Rangarajan, H. P., and Meador, T. 2011. "Parametric control of BIM elements for sustainable design in Revit: linking design and analytical software applications through customization," *Perkins+Will Research Journal*, 3 (1), p. 32.

Asl, M. R., Bergin, M., Menter, A., and Yan, W. 2014. "BIM-based parametric building energy performance multi-objective optimization, *Fusion*: Proceedings of the 32nd eCAADe Conference - Volume 2, Department of Architecture and Built Environment, Faculty of Engineering and Environment, Newcastle upon Tyne, England, UK, 10-12 September 2014, pp. 455-464.

Briscoe, D. 2014. "Parametric planting: green wall system research + design using BIM," ACADIA 14: *Design Agency*: Proceedings of the 34th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA), ISBN 9781926724478, Los Angeles 23-25 October, 2014), pp. 333-338.

Celani, G. and Vaz, C. 2011. "CAD scripting and visual programming languages for implementing computational design concepts: a comparison from a pedagogical point of view. *International Journal of Architectural Computing (IJAC)* Volume 10, Issue 1, pp. 121- 137.

Davis, D. 2013. "Modelled on software engineering: flexible parametric models in the practice of architecture," PhD dissertation, RMIT University, http://www.danieldavis.com/thesis/, pg. 161.
Gamas, A., Konis, K., and Kensek, K. 2014. "A parametric fenestration design approach for optimizing thermal and daylighting performance in complex urban settings." ASES/Solar 14 Annual Conference, San Francisco, CA, July 2014.
Jabi, W. 2014. "Parametric spatial models for energy analysis in the early design stages," *SimAUD '14*: Proceedings of the Symposium on Simulation for Architecture & Urban Design, Article No. 16, Society for Computer Simulation International, San Diego, CA.
Kensek, K. 2014. "Got macros? Scripting and coding for BIM," *AECbytes Viewpoint #70*, May 22, 2014, http://www.aecbytes.com/viewpoint/2014/issue_70.html. Also published in *AECbytes Magazine*, Q2 2014.
Kensek, K. 2014. "Integration of environmental sensors with BIM: case studies using Arduino, Dynamo, and the Revit API," "Integración de sensores medioambientales con BIM: casos de estudio usando Arduino, Dynamo, y Revit API." *Informes de la Construcción*. Vol. 66, 536, e044 octubre-diciembre 2014 ISSN-L: 0020-0883, pp. 31 – 39. doi: http://dx.doi.org/10.3989/ic.13.151.
Kensek, K. and Henkhaus, A. 2013. "Solar access zoning + building information modeling," ASES National Solar Conference, Baltimore, MD, April 2013.
Kensek, K. and Knowles, R. 1997. "Solar access zoning: computer generation of the solar envelope," ACSA Southwest Regional Meeting Proceedings, October 1997.
Konis, K. and Kensek, K. 2015. "Leveraging BIM and scripting for passive design parametric analysis," SoCal Gas Commercial Sustainable Development Program Innovative Design for Energy Efficiency Activities (IDEEA) grant. Prepared for Jeff Staller, CEM, LEED AP, Associate Technical Director, Building Science, TRC Solutions. Technical report.
Kron, Z. 2014. The bridge project was based on Kron's bridge developed in Dynamo, https://www.youtube.com/watch?v=pgPLlFSqI6Y, published on June 6, 2014, last accessed May 26, 2015.
Leitão, A. and Santos, L. 2011. "Programming languages for generative design: visual or textual?," *Respecting Fragile Places*, 29th eCAADe Conference Proceedings / ISBN 978-9-4912070-1-3], University of Ljubljana, Faculty of Architecture (Slovenia) 21-24 September 2011, pp. 549-557.
Makris, M., Gerber, D., Carlson, A., and Noble, D. 2013. "Informing Design through parametric integrated structural simulation: iterative structural feedback for design decision support of complex trusses," *eCAADe 2013*: Computation and Performance – Proceedings of the 31st International Conference on Education and Research in Computer Aided Architectural Design in Europe, Delft, The Netherlands, September 18-20, 2013, ISBN: 978-94-91207-04-4 (vol. 1) · 978-94-91207-05-1 (vol. 2).
Maleki, M., Woodbury, R., and Neustaedter, C. 2014. "Liveness, localization and look-ahead: interaction elements for parametric design," DIS '14: Proceedings of the 2014 Conference on Designing Interactive Systems, Vancouver, BC, Canada, pp. 805-814, ACM New York, NY, ISBN: 978-1-4503-2902-6 doi>10.1145/2598510.2598554.
Miller, N. 2009. "Parametric strategies in civic architecture design," Proceedings from the 29th Annual ACADIA Conference. pp. 144-152.
Miller, N. 2014. Post on Miller's blog, "Introducing RHYNAMO: Apply for BETA Testing!" (Friday, August 22, 2014). Last accessed May 28, 2015.
Negendahl, K. 2015. "Building performance simulation in the early design stage: an introduction to integrated dynamic models," *Automation in Construction*, Volume 54, June 2015, pp. 39-53.
Ogueta, C. 2012. "User innovation in digital design and construction: dialectical relations between standard BIM tools and specific user requirements," thesis, MS Architecture Studies, MIT, June 2012.
Wurzer, G. and Pak, B. 2012. "Lawnmower: Designing a web-based visual programming environment that generates code to help students learn textual programming," *Digital Physicality*: Proceedings of the 30th eCAADe Conference - Volume 1 / ISBN 978-9-4912070-2-0, Czech Technical University in Prague, Faculty of Architecture (Czech Republic) 12-14 September 2012, pp. 655-663.
Zarzycki, A. 2012. "Parametric BIM as a generative design tool," *Digital Aptitudes*: ACSA Conference, Boston, MA.