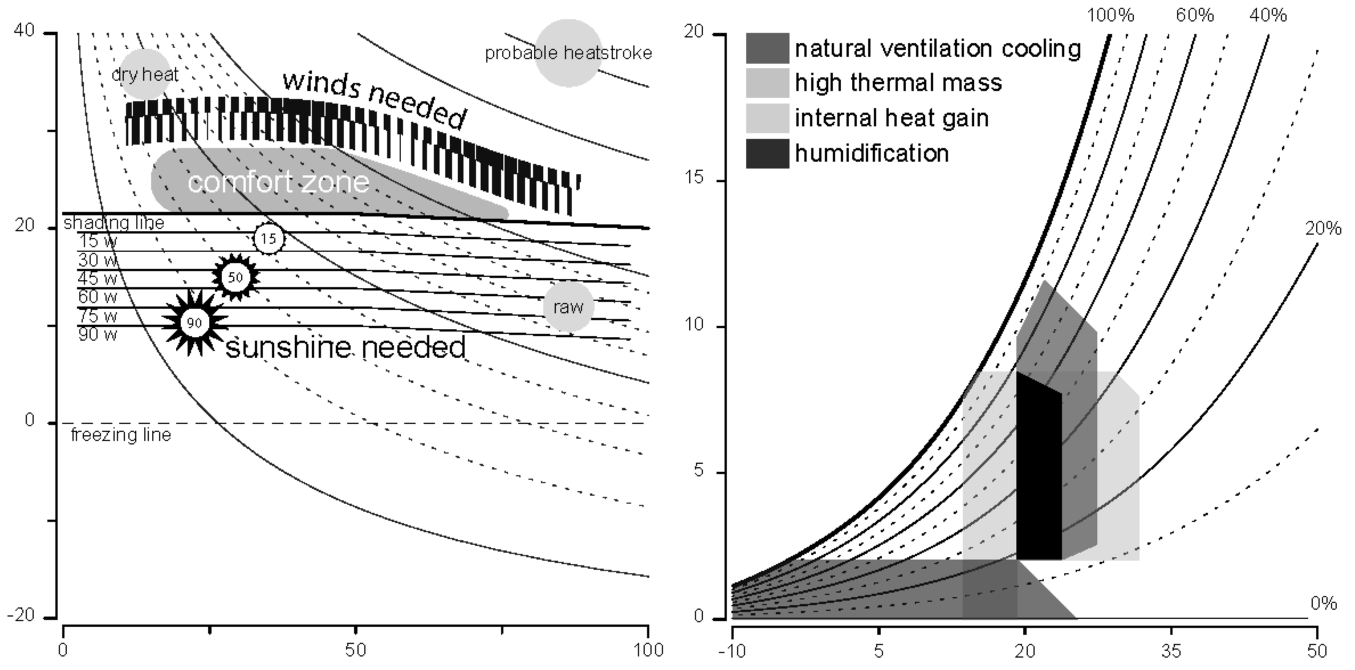
DHOUR: A BIOCLIMATIC INFORMATION DESIGN PROTOTYPING TOOLKIT

Kyle Steinfeld University of California, Berkeley
Brendon Levitt Loisos + Ubbelohde



1 Victor and Aladar Olgyay's Bioclimatic chart (left) and Baruch Givoni's Building Bioclimatic Chart (right), redrawn by author.

ABSTRACT

The qualification of predicted building performance through quantitative methods is as challenging as it is crucial to the meeting of the mandate to design buildings better adapted to their bioclimatic conditions. Methods for the visualization of building performance data that have found success in the past struggle in the contemporary context of large computational data sets. While application of building performance simulation to architectural design is highly context-sensitive, existing approaches to the visualization of simulation results are generalized and provide the designer with a preconfigured battery of visualizations that are, by definition, not calibrated to specific questions or contexts. This paper presents a new prototyping visualization toolkit, developed for the Grasshopper (Rutten 2013) visual programming environment, which enables the situational development of information graphics. By enabling more nuanced and customizable views of complex data, the software described here offers designers an exploratory framework in contrast to the highly directed tools currently available. Two case studies of the application of this toolkit are then presented, the results of which suggest that a more open framework for the production of visualization graphics can more effectively assist in the design of buildings responsive to their bioclimatic environments.

INTRODUCTION

While many approaches to the problem of qualifying predicted building performance through quantitative methods have proven successful in the past, they often rely on graphic methods that build intuitive understanding through visual means. These same approaches struggle in the contemporary context. The integration of performance metrics into contemporary architectural design through the use of data visualization faces the new challenge of integrating bioclimatic data orders of magnitude more vast than it has been available in the past—from advanced thermodynamic simulations to detailed descriptions of climate. The dominant model for simulation and climate data visualization software—the "design tool" approach, does not adequately meet the demands of these new conditions. This paper presents an alternative software, one based on the Situated Bioclimatic Information Design (SBID) approach(Steinfeld, et al, 2010) to information design tools. This software and approach is better suited to the challenges presented by contemporary architectural design.

While the underlying physical principles of building simulation models can be generalized to a wide range of scenarios, the interpretation and visualization of simulation results is highly context-sensitive. The salient relationships at work in any given design situation are idiosyncratic to the qualities and complex dynamics of climate, site, activity, building typology and tectonic-material system. As a consequence, the specific queries of simulation data that a designer may require are difficult to effectively anticipate outside the context of a situated design problem. In spite of this, most existing approaches to the visualization of simulation results take a generalized approach, providing the designer with a preconfigured battery of visualizations appropriate to the broadest possible range of design situations. This disconnect may be summarized as the difference between prototyping and mass-production: where data visualization in the context of an architectural design calls for the agility and diversity akin to prototyping, while existing software offers the uniformity and scalability of mass-production. As a result, existing approaches to visualization impose artificial limits on the way analysts are encouraged to think about simulation results.

After surveying a selection of the most current existing approaches, this paper presents Dhour, a bioclimatic information design toolkit developed to address the gap that exists between the ability to produce building performance data and the ability to visualize it in complex, nuanced and graphically compelling ways. Developed for the Grasshopper visual programming environment, Dhour applies principles of the Situated Bioclimatic Information Design approach to information design tools.

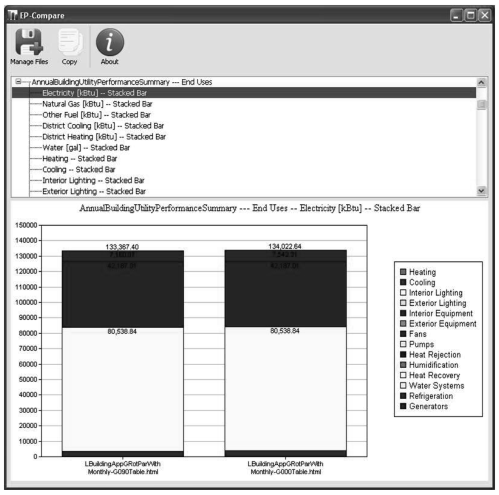
Two case studies of the application of this toolkit are then presented with each developed by students in a building performance visualization course at UC Berkeley led by Kyle Steinfeld and Brendon Levitt in the Spring of 2013. The results of these case studies suggest that a more open framework for the production of visualization graphics will both better align data visualization with other modes of architectural drawing, and more effectively assist in the design of buildings responsive to their bioclimatic environments.

DRAWINGS AND GRAPHS

Historically, drawings have been the primary cognitive instrument of design thinking. As Donald Schon asserts in his seminal work (Schön, 1983) on processes and cultures of design, architectural drawings are valued not only as methods for finding answers, but also as means to frame previously undiscovered questions. While drawing is described as a "reflective loop" of visual reading and responding, the primary utility of drawing lies not in the working out of a problem, but in the stimulation of a conversation. This may take the form of a metaphorical conversation between an architect and the properties of design, or a real conversation between stakeholders in the process of design. Drawing practices are most valuable to this vision of the design process that emphasize exploration over determinism, and encourage the subjectivity necessary to project new forms and organizations into directly authored visual material. Schon's work of thirty years ago is widely accepted as one of the most complete and insightful descriptions of the process of design. While it still resonates with many architects today, its relevance has been challenged by new design practices and technologies.

Contrast Schön's view on the value of architectural drawings with the processes of quantitative analysis supported by simulation software. Unlike architectural drawings, simulations are valued primarily for their ability to generate accurate predictions, and hence the graphs, charts and other visual material that follow carry forward the weight of this prediction's authority (Latour, 1986). The use of building simulation has been described as containing an "aura of objectivity" that requires architects to "abdicate control of their design" (Turkle, 2009). If graphs carry an authority equal in power to, yet conflicting with, the subjectivity of drawings, how can simulation better integrate with a creative design process that allows for the interrogation and synthesis of such data? One possible step toward reconciling these competing models of design thinking is to challenge the immutability of data-driven graphs, bringing them more in line with an exploratory model of drawing.

The task set before architects and engineers to effectively communicate the complex bioclimatic dynamics of heat, air and light has presented a unique challenge. It requires a deep understanding of the interrelationships between these phenomena, and a capacity to translate large amounts of quantitative building performance data into qualitative assessments of an intended spatial experience. The visualization of data can play a crucial role in developing this capacity: clarifying the often complex parametric interrelationships that drive building simulation models, filtering


EP-Compare
Manage Files
Copy
About
AnnualBuildingUtilityPerformanceSummary --- End Uses
Electricity [kBtu] -- Stacked Bar
Natural Gas [kBtu] -- Stacked Bar
Other Fuel [kBtu] -- Stacked Bar
District Cooling [kBtu] -- Stacked Bar
District Heating [kBtu] -- Stacked Bar
Water [gal] -- Stacked Bar
Heating -- Stacked Bar
Cooling -- Stacked Bar
Interior Lighting -- Stacked Bar
Exterior Lighting -- Stacked Bar
AnnualBuildingUtilityPerformanceSummary --- End Uses -- Electricity [kBtu] -- Stacked Bar
150000
140000
130000
120000
110000
100000
90000
80000
70000
60000
50000
40000
30000
20000
10000
0
133,367.40
42,187.01
80,538.84
134,022.64
42,187.01
80,538.84
LBuildingAppGRotParWith Monthly-G090Table.html
LBuildingAppGRotParWith Monthly-G000Table.html
Heating
Cooling
Interior Lighting
Exterior Lighting
Interior Equipment
Exterior Equipment
Fans
Pumps
Heat Rejection
Humidification
Heat Recovery
Water Systems
Refrigeration
Generators

2 View of EP-Compare, a results viewer for EnergyPlus, taken as a screenshot by author.

out the irrelevant, and distilling quantitative results into a qualitative understanding. In this sense, it is possible for graphs to be seen as more "mutable" than they are sometimes perceived to be, and for data visualization to become an interrogative medium analogous to architectural drawing in that it offers designers a means to enter into a dialog with the "materials" of a design situation. Bringing the quantitative and seemingly immutable results of simulation closer to the traditional role that drawings play in the design process will serve to better integrate quantitative analysis into architectural design.

CURRENT APPROACHES TO BIO-CLIMATIC INFORMATION DESIGN

Graphic visualization of data in the service of bio-climatic design predates computer-aided data visualization. Victor and Aladar Olgyay's Bioclimatic chart (Olgyay1963) is one of the earliest examples of information design applied for this purpose, and remains one of the most elegant. The bioclimatic chart synthesizes a format for plotting climate data, a depiction of the range of human comfort, and a set of "needs" that suggest methods for bringing uncomfortable periods into the range of human comfort. Baruch Givoni's Building Bioclimatic Chart (Givoni, 1969) extends the work of Olgyay by depicting the potential impact of a range of passive and active design strategies in terms of an "expansion" of the comfort zone. While the predicted effectiveness of these design strategies relied heavily on heuristics and approximations, the basic approach taken by Givoni anticipates modern computational methods. Due to the restraint enforced by the limits of computing power and availability in the time of Olgay and Givoni (Figure 1), these pre-computational examples demonstrate effective use of averaged data in identifying the most useful metrics, and the judicious application of only the most salient information to produce visualizations with a high data-ink (Tufte 2001) ratio.

In contrast to Olgyay and Givoni, contemporary designers face the challenge of visualizing bio-climatic data with a number of computational tools at their disposal: from high-level techniques designed with this purpose in mind (often built into simulation software) to low-level applications that facilitate the manipulation and visualization of data in general. The following survey presents a selected subset of these tools, discusses the advantages and disadvantages of the approach implied by each, and concludes by identifying the exigencies created by their deficiency in serving the contemporary design context.

SURVEY OF EXISTING BIOCLIMATIC DATA VISUALIZATION SOFTWARE

Bioclimatic data visualization software can be grouped into three types: data generation, data analysis and data visualization software. The characteristics of the graphs that each group provides vary markedly. Each category shows a different approach to the flexibility of changing graph type, aesthetic and content. In addition they vary by their ability to post-process data and integration into a streamlined workflow.

DATA GENERATION SOFTWARE

Data generation software consists of simulation engines and their GUIs, such as EnergyPlus (US Department of Energy, 2013) as seen in Figure 2, EQuest , IES Virtual Environments or TRNSYS. Most data generation software bundle a "results viewer" that allows quick data visualizations. These are typically limited to one or two time-series graph types (line or heatmap) and one or two summary graph types (bar or pie). While these types of software packages allow any internally produced data to be visualized, they do not allow for native post-processing. In addition, the aesthetic is pre-packaged which means line weight, color, font type and size, and labels are not customizable. The strength of this approach is that simulated data can be quickly visualized, but the inability to customize graph type or aesthetic imposes limits on analyses.

DATA ANALYSIS SOFTWARE

Data analysis software takes outputs from simulation engines or climate data and automatically post-processes them into preconfigured battery of graphs. The Ecotect Analysis Weather Tool and UCLA Climate Consultant (Milne 2013) exemplify this type. Data analysis software delivers a suite of pre-packaged time-series graph types—including line, area, scatter and heatmap (Figure 3), as well as summary graph types (bar or radial). For each graph type content is limited to a pre-populated list of variables and the aesthetic is limited to a few basic parameters such as color and axis scale. Data analysis software usually allows more varied visualizations than data generation software, but the user is still limited to a large extent by pre-packaged graph types and aesthetics. Data analysis software also does not accommodate native post-processing and it has the added drawback of not being integrated with the simulation engine.

DATA VISUALIZATION SOFTWARE

Data visualization software, exemplified by Excel, Matlab or R, have increased flexibility in terms of graph type, content and aesthetic. They accommodate the post-processing of complex conditional logic. However, they are not integrated with the simulation engine, and as a result additional time for data input and switching of user interface is required. In addition, graph aesthetic can be difficult to control for all but expert users.

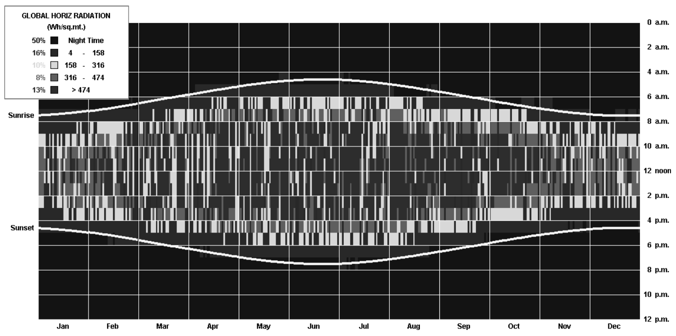
LIMITATIONS OF EXISTING SOFTWARE

In practice, the most widely used visualization software in the analysis of building performance are those bundled with the simulation engine. However, these applications allow little customization of either content or aesthetic. While some analysts supplement built-in visualizations by exporting data to more robust data visualization software such as Matlab, this process is cumbersome and time-intensive. More often, data is presented either directly from the simulation application or as a slightly-customized Excel graph. The limits of the visualization put artificial limits on the way analysts are encouraged to think about the simulation results.

Dhour was developed to address the gap that exists between the ability to produce building performance data and the ability to visualize it in complex, nuanced and graphically compelling ways. Its Rhino/Grasshopper-integration allows for real-time and WYSIWYG use of advanced, and almost infinite, drawing tools. With this comes the potential for full control over all aesthetic considerations from line weight and color palette to complex geometric manipulation. The Grasshopper platform also allows integration with simulation software, such as EnergyPlus and Radiance. In theory, Dhour could visualize data from any software that can be controlled via Grasshopper's library of Python, C+, or Visual Basic scripting languages via a COM (Component Object Model) interface. By the same token, those same languages can be used to post-process results using advanced conditional logic.

PREVIOUS WORK

The software presented here applies the SBID approach to information design tools, and is based upon a re-implementation of previous work developed as a low-level library of methods written in Java. Two concepts from this previous work are further developed in the scope presented here: the "Dhour" data type (Steinfeld, et al 2010) , and the identification of a discrete set of "graphic interfaces" (Steinfeld, Schiavon, Moon 2012) as the most useful for innovative climate-responsive design processes.



3 View of a Heatmap of Global Horizontal Radiation in Climate Consultant, taken as a screenshot by author.

THE DYR AND DHR DATA TYPES

The Dhr (Dynamic Hour) data type is a custom class developed in the Java programming language. This type constitutes the most basic element for importing, manipulating, filtering, and displaying simulation and climate data. The configuration of this data type requires several assumptions regarding the nature of the data under consideration:

- The data are described at a resolution of 1-hour increments, or may be recomposed into 1-hour increments.
- The data associate the same parameter values with every hour of the year.
- The values associated with each hour may be described as floating-point numbers.

Table 1: Dhr Properties and Methods – Java Implementation
data [Hashtable <String, float>]
Stores key-value pairs containing parameter data for this hour.
id [int]
Indicates the position of this hour within the year (0->8760).
val(key) [float]
Returns a stored value associated with a given key from the Hashtable.
put(key) [void]
Stores a new value in the Hashtable, and associates it with the given key. If the key already exists, it is overwritten.

GRAPHIC INTERFACES

In a previous scope of work, a limited set of programmatic graphic interfaces were identified as the most relevant to data visualization in architectural design, and developed in the Java programming language. Figure 4 shows three interfaces previously developed (Heatmap, Solarplot, and Psychrometric Graph) (Figure 4).

METHODS

This section presents the implementation details for a prototyping visualization toolkit developed for the Grasshopper visual programming environment. Discussed here are the rationale and practicalities of developing for a visual programming environment/ parametric modeler, the general data visualization workflow supported by the software, and the details of core functionality. This framework was developed alongside a course at UC Berkeley, led by Kyle Steinfeld and Brendon Levitt.

REIMPLEMENTATION FOR A VISUAL PROGRAMMING ENVIRONMENT

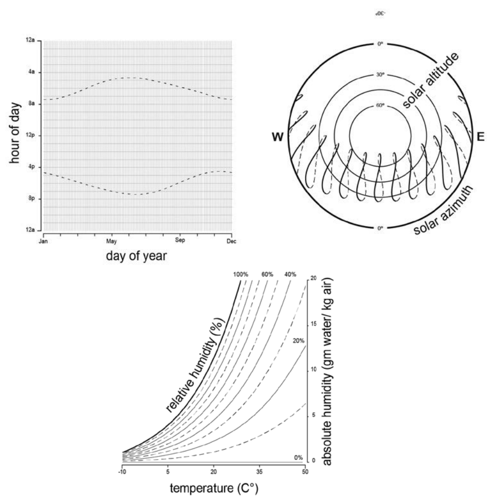
Visual programming environments represent a "half step" between scripting interfaces and GUI-driven software, and offer a lower threshold of entry for novice users while maintaining some of the flexibility of a textual programming environment. This combination of ease of use with low-level access forms an ideal environment to support both an architectural design and building science audience.

In addition, reimplementation in the Grasshopper environment facilitates:

- The ability to supplement built-in functionality with scripted methods using VB, C# and Python components.
- Access to a full-featured geometry library provided by the Rhino CAD environment.

THE DHOUR DATA TYPE

In the current scope of work, the Dhour class was reimplemented in C# as a custom parameter for Grasshopper. Several features were added in order to integrate with the parametric environment, and in response to the need of the additional graphic interfaces described below (Table 2).



4 Three graphic interfaces developed in the Java implementation of DYear

Table 2: Dhour Properties and Methods – C# Implementation
vals [Dictionary <String, float>]
Stores key-value pairs containing parameter data for this hour.
hr [int]
The position of this hour within the year (0->8760).
pos [Point3d]
The position of this hour as plotted in a graphic space
color [Color]
The color of this hour as plotted in a graphic space
is_surrogate [bool]
Indicates that the data contained within this hour describes a range of hours within a given year. For example, the average values for an entire day may be stored within a single Dhour.
val(key) [float]
Returns a stored value associated with a given key from the Hashtable.
put(key) [void]
Stores a new value in the Hashtable, and associates it with the given key. If the key already exists, it is overwritten.

GRAPHIC INTERFACES

The three programmatic graphic interfaces previously developed were generalized in order to account for both climate data and simulation result data, and greatly expanded in order to encompass a more diverse set of graphic spaces. The three interfaces previously developed were each updated for the parametric environment of Grasshopper, and a number of new interfaces were developed. See the Spatialization section, below, for a complete description of each.

DATA VISUALIZATION WORKFLOW

As Dhour is intended to provide an exploratory framework, the anticipation of a precise workflow is presented merely as a guide, and as a way of classifying the more elemental functional units for acquiring, manipulating, and visualizing data. The looseness of the workflow described here is reinforced by the nature of visual programing environment, which prohibits the enforcement of a strict linear sequence.

One may roughly divide the process of data visualization into four stages: data acquisition, data manipulation (including filtering, sorting, and culling), data spatialization and colorization, and the translation of graphic results into a desired format. As diagrammed in Figure 5, Dhour offers functionality that supports each of these stages, including the importing of multiple data types and the exporting of resulting geometry to multiple formats (Figure 5).

CORE FUNCTIONS AND METHODS

The data visualization workflow described above is supported through the following clusters of functionality, each of which is related to a number of Grasshopper components that may be deployed and configured by the user of the software in any sequence required.

Some of these components produce or acquire streams of data. Others manipulate existing streams of data, either by reorganizing the configuration of the data stream or by adding graphical information to individual hours. In every case, the metaphor of the "flow" of data is preserved, with most components requiring a set of Dhours on which to operate, and providing a modified set of Dhours in return.

PRIMITIVE FUNCTIONS

Primitive functions provide direct access to the construction and decomposition of the basic elements in Dhour. It is more common in practice to perform these operations through higher-level means, but access to the elemental properties of hours is often useful for debugging and supporting unanticipated workflows. For example, while one would more likely import data into the Dhour format through some process of data acquisition, Figure 6 demonstrates the elements contained within the Dhour data object in a manner useful for instruction (Figure 6).

DATA ACQUISITION FUNCTIONS

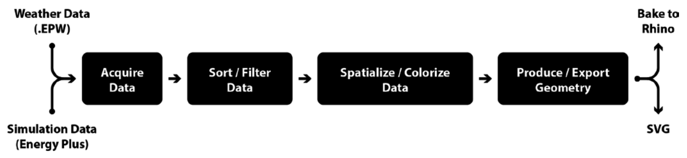
In contrast to a "results viewer" tool, which is tied to a single simulation software, an open data visualization framework must provide mechanisms that allow the acquisition and synthesis of data from multiple sources. Components were developed to import EPW weather file and Energy Plus results files, and to translate the data into the Dhour data type.

FILTERING AND SORTING FUNCTIONS

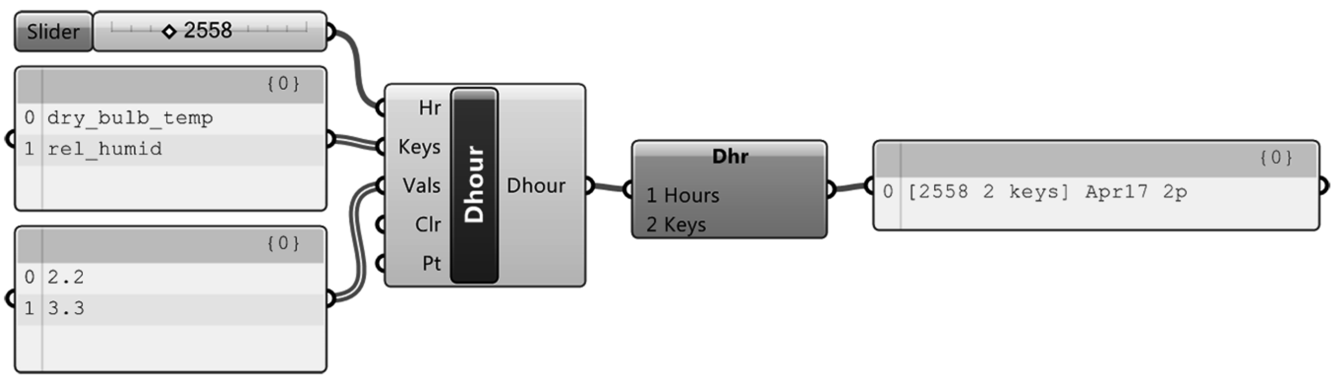
Components developed for this category are diverse, and include those that filter data by given time periods and those that perform standard statistical operations. Some, such as the LimitKeys and MergeHours components, support tasks commonly encountered when working with streams of data from multiple sources. Others are closely related to specific forms of visualization, such as the HourFreq component, which counts and sorts hours by their "binning" within a given range of values. This is a necessary step when producing histograms, windroses, and other summary graphics.

COLORIZATION FUNCTIONS

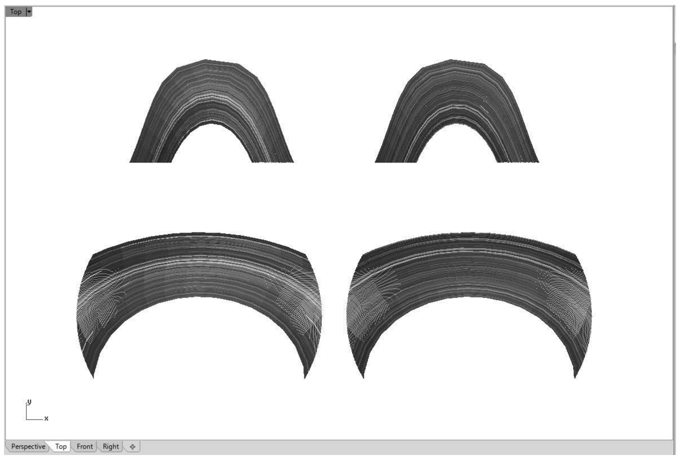
Colorization and Spatialization functions are primarily responsible for determining the graphic quality of visualizations produced with Dhour. While closely related, it was found to be advantageous to separate these two categories of operation in order to reduce redundancy when producing multiple related graphics that required a coordinated approach to color. Additionally, the approach of defining object colors in a separate operation allows for more sophisticated color schemes, such as the two-parameter colorization of a heatmap (Figure 7). Here, the hue of each line maps the two distinct parameters of dry bulb temperature and relative humidity simultaneously.


Weather Data (.EPW)
Simulation Data (Energy Plus)
Acquire Data
Sort / Filter Data
Spatialize / Colorize Data
Produce / Export Geometry
Bake to Rhino
SVG

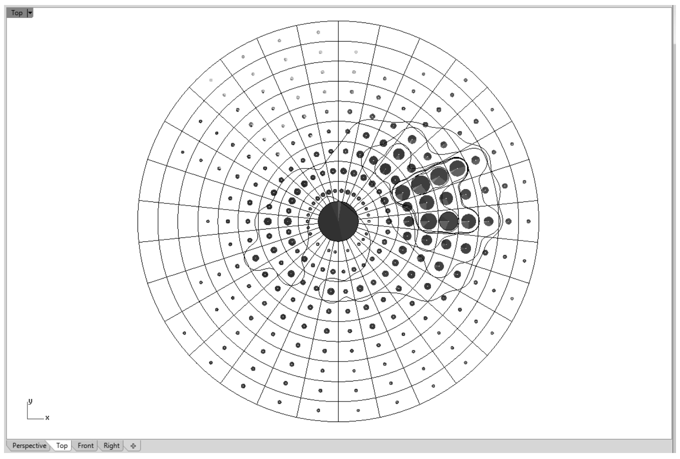
5 Data Visualization Workflow



6 The Basic Construction of a Dhour Data Object



7 Screen-shot of a Preview of a Solar Position Graphic Spatialization in Rhino.



8 Screen-shot of a Preview of a Value-Value Graphic Spatialization in Rhino.

SPATIALIZATION FUNCTIONS

Spatialization functionality is at the utility core of Dhour, as it enables the structuring of data into formats demanded by the most common graphic spaces used in quantitative analysis. Some of these formats are well-known and well-supported by generalized data visualization software, such as time-value graphs (including line and bar graphs), pie charts, and histograms. Other formats are more specific to the domain of bioclimatic data visualization, such as the psychrometric chart, the solar position chart and the diurnal time-value graph.

The remaining formats supported by Dhour are not common in either of these contexts, but rather arose from the observation of the needs of bioclimatic information design in practice. The wind rose, for example, suggests a radial value-value graphic spatialization not unlike a conventional radar graph, but with the added requirement to collect samples into discrete "bins" to form a visual averaging effect without loss of data fidelity (Figure 8).

VALIDATION FUNCTIONS

Data validation is an essential step in the production of any visualization, and is especially important when dealing with the complexities of merging data from multiple sources. Components developed for this category (Figure 9) provide the user with a method for performing quick checks of incoming or processed data to ensure that the format is correct and the data is behaving as expected.

DATA INTERNALIZATION

Dhour is designed to synthesize data from multiple sources, including weather files and simulation results from multiple runs. The process of importing these data files usually involves the creation of a link such that one may update the linked file and see the changes propagate across the visualization. Active links to data files can also cause problems when collaborating with others who may not have access to the same file system, or when working between multiple computers. To mitigate these issues, Dhour provides a data internalization mechanism to embed data directly into a Grasshopper component, and a way of visually differentiating internalized data (Figure 10).

SURROGACY

One core assumption of the current implementation of Dhour is that the data are described at a resolution of one-hour increments. As a consequence, as the Dhour object is the single storage mechanism for data, ambiguity may arise when averaging hours across a given linear or diurnal time period. The Periodic Stats component performs statistical operations on a given set of hours, for example, calculating diurnal monthly averages. While we may expect a stream of "resulting" hours (in this case twelve resulting hours, one for each month) containing this averaged data that may be later visualized in some way, we require a way of indicating that each of these results stands in for some number of source hours. Without this, components that attempt to process these "surrogate" hours may produce unexpected results. To this end, Dhour provides the capability to "tag" each hour as a surrogate, and to graphically label these hours (Figure 10).

CASE STUDIES

The Dhour Grasshopper Plugin was used for building performance data visualization in a graduate-level seminar led by the authors at UC Berkeley in the Spring of 2013. Students used Viper (Reinhart and Jakubiec, 2013), the Grasshopper add-on to DIVA for Rhino, as a GUI for EnergyPlus and results were visualized with Dhour. The class was divided into teams of two to four students each working on the thermal analysis of a zone in an existing building. Two analyses will be presented here to illustrate the application of Dhour to thermal performance data visualization.

CASE STUDY 1: CONDITIONAL LOGIC AND GRAPHIC OVERLAY

The first case study (Figure 11), authored by Joyce Kim and Oscar Josue Diaz, looks at the potential for increasing occupant comfort during the summer through cross ventilation and night flushing of a Studio Art building in New Haven. The resulting drawing is shown in Appendix A. The team searched for a graphical method that would help them to understand the potential for reducing the number of hours of mechanical cooling. They used conditional logic in Python and graphical overlay techniques to arrive at an answer.

First, a wind map was created showing the magnitude of hourly wind speed through each day (Y-Axis) and year (X-Axis). Then, using a post-processing script for adaptive thermal comfort, hours that were too cold were masked out while hours of comfort were muted. Hours warmer than comfort were color-coded to show the magnitude of wind speed. These hours then became the focus of subsequent studies focused on natural ventilation strategies.

The team checked the orientation of the building against the prevailing winds, using Dhour to generate custom wind roses. They used "binning" techniques to post-process the data, grouping it by time of day, time of year, direction, frequency and wind speed. Wind frequency is plotted along the radius with larger wedges indicating higher frequencies. Furthermore, each wedge is a stacked bar chart where color value indicates wind speed. From this, the team could see that there is frequent but slow winds from the northeast on summer nights and the winds shift to the south on summer afternoons.

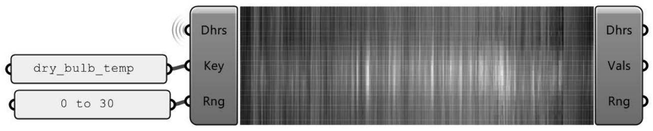
Once the team understood that wind would be available and accessible for cross ventilation and night purge, they simulated and visualized the resultant comfort conditions. These heat-maps chart the degrees from comfort for each hour of the day (Y-Axis) and year (X-Axis). Blue values are degree-hours too cold, yellow-red values are too hot and light gray indicates comfort. A summary histogram reports the number of hours at each degree from comfort. The progression from "Base Case" to "Daytime Cross" to "Night Flush" shows the decrease in overheating hours. Overheating is decreased further by considering evaporation off the skin due to increased air movement when cross ventilation is present. The data was post-processed with a script that increases the upper threshold for comfort by two degrees Celsius for any hour with cross ventilation.

Below each comfort graph is a simple line graph that reports the volumetric airflow rate for the analysis zone. This line graph explicitly shows the increased airflow with cross ventilation and night flush, and importantly, allows for the correlation between airflow and thermal comfort. It also served as a diagnostic check to ensure the simulation was running correctly.

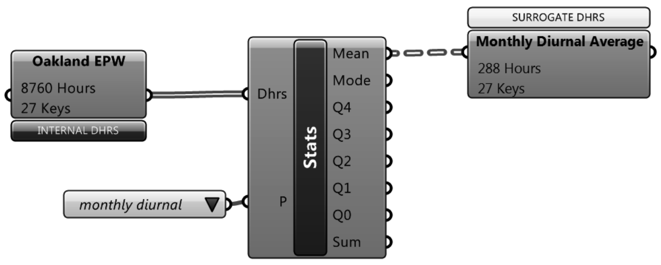
By allowing cross ventilation during occupied hours and night flushing during unoccupied hours, the number of overheated hours was reduced by 78 per cent (from 2,462 to 557 hours). Both the graphical methods and conditional logic enabled by Dhour were key to this exploration. The team was able to mask out extraneous information and then superimpose an overlay rich with the information most pertinent to their exploration. They were also able to post-process the raw data with a conditional logic that expanded the scope and specificity of the energy simulation results. Through a juxtaposition of graphics, both in sequence and in type, the team was able to see direct correlations between a given strategy and the effect on occupant comfort.

CASE STUDY 2: HYBRID GRAPHS

The second case study (Figure 12), authored by Margaret Pigman, Bin Chen and Jonghoon Park, examines the sensitivity of occu-



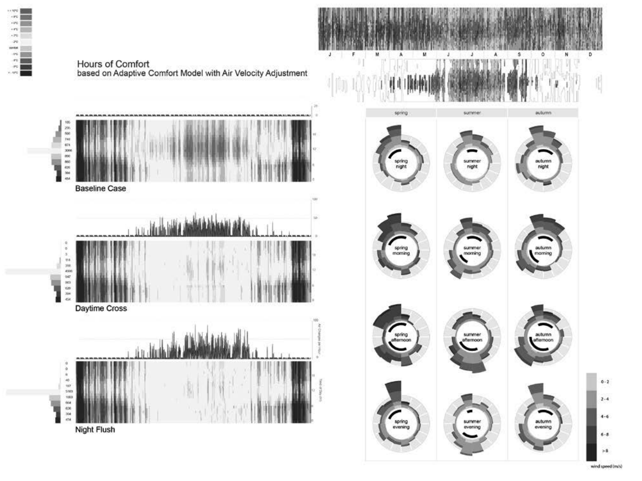
9 Screen-shot of a Heatmap Validation Component.



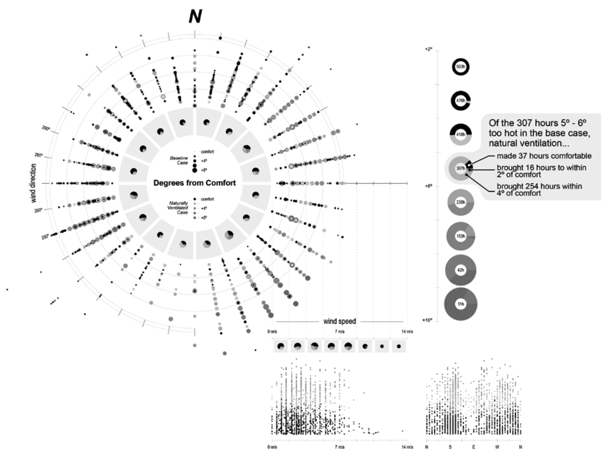
10 Data Internalization and Surrogate Dhours. Note the graphic modifiers applied to the Grasshopper component.

pant comfort to wind direction and speed in a naturally-ventilated house in Japan. The resulting drawing is shown in Appendix B. Using Dhour, the team experimented with a hybrid graph geometry and custom scripting in Python.

The resultant graph combines several traditional graph types into a complex hybrid. Wind direction is signified by thirty-six lines radiating from the center. Wind speed is signified by the distance from center. Degree of overheating without cross ventilation (in the base case) is signified by dot size. Degree of overheating with cross ventilation is signified by hue. For example, a large, red dot indicates an hour that is overheated with or without cross ventilation. A large, black dot indicates an hour that is comfortable with cross ventilation but overheated without ventilation.



11 Potential for increasing occupant comfort during the summer through cross ventilation and night flushing of a Studio Art building in New Haven



12 Sensitivity of occupant comfort to wind direction and speed in a naturally-ventilated house in Japan

By examining the patterns of black dots, the chart is designed to reveal under what wind direction and speed comfort conditions result from cross ventilation. If large black dots were grouped in only one quadrant of the graph, it could be concluded that only wind from that direction would make the house comfortable. If large black dots were grouped around the same radius, then only those wind speeds would result in comfort. Instead, the random distribution of black dots in the graph reveals that comfort conditions do not consistently depend on a particular wind speed or direction.

This conclusion is reinforced by the summary pie graphs. The inner circle shows an even distribution of comfort conditions resulting from natural ventilation regardless of wind direction. The pie charts at bottom right present a summary of comfort conditions at the different wind speeds. Although the three highest wind speeds generally result in greater comfort hours, one can also see that the lowest wind speed results in more comfort hours than the second through fifth wind speeds.

DISCUSSION

The two case study projects presented above illustrate the application of Dhour to thermal performance data visualization, and demonstrate the mutability of data-driven graphic material put in the service of a situated design question. In both, there are several moments of visualization inventiveness that result from work with Dhour that may not have been possible through the use of other data visualization tools. Because of the WYSIWYG interface, color palette and lineweight is carefully calibrated in the resulting graphics. Because of the fine-grained control that Dhour offers, the hierarchy of line and color can also be tuned to the scale of the specific image and information.

Most importantly for building performance modeling, these case studies show that the close coupling of Dhour with Python in the Grasshopper environment allows for the post-processing of complex conditional logic. For professionals, there are several "post-processing" functions that are common in energy analysis but cumbersome owing to software limitations. These include:

- Statistical analysis - including binning, averaging, deviation
- Conditional logic - if/then statements and loops, particularly for controls protocols
- Data aggregation - combining output data to produce derivative metrics (such as adaptive comfort)
- Parametric changes - post-processing independent or derivative variables (such as Return-on-Investment or Carbon Emissions)

Dhour and Python make these types of functions readily accessible for analysis and diagnostics.

CONCLUSION

Dhour is designed to offer a more open framework for the production of visualization graphics that will both more closely align data visualization with other modes of architectural drawing, and more effectively assist in the design of buildings responsive to their bioclimatic environments.

LIMITATIONS OF CURRENT IMPLEMENTATION AND FUTURE WORK

The current implementation of Dhour presents some limitations and opportunities for further research and development:

- Computational Speed - Dhour was developed by non-expert programmers in response to the changing needs of an active user group. As a result, several inefficiencies remain to be resolved, and are currently causing non-trivial lag times when computing large numbers of geometric objects.
- Granularity – One of the core assumptions of Dhour is the elemental unit of the hour, and the default collection of the 8760 hour year. Partial-year simulations or time steps greater than one hour cannot be accommodated in this version.
- Text Integration - We have also found that creating text for legends and labels has been cumbersome. If, however, this software is used primarily for prototyping as opposed to mass-production, this is less of an issue. Manual integration of text on a one-off graph is easily accomplished.
- Host Dependence - DHour was developed specifically as an add-on to Grasshopper. Future versions might be developed for other platforms, such as Sketchup or Revit

WORKS CITED

EP-Compare. 2013. US Department of Energy. Accessed April 16. http://apps1.eere.energy.gov/buildings/energyplus/energyplus_utilities.cfm.

Givoni, Baruch. 1969. *Man, Climate, Architecture.* 1st ed. New York: Elsevier Science Ltd.

Latour, Bruno. 1986. "Visualization and Cognition: Thinking with Eyes and Hands." *Knowledge and Society: Studies in the Sociology of Culture Past and Present* 6: 1–40.

Milne, Murray. 2013. *Climate Consultant* (version 5.4). UCLA Architecture and Urban Design. http://www.energy-design-tools.aud.ucla.edu/.

Olgyay, Victor. 1963. *Design With Climate: Bioclimatic Approach to Architectural Regionalism*. Princeton University Press.

Reinhart, Christoph, and Alstan Jakubiec. *Viper* (version 1.902). Solemma. http://diva4rhino.com/user-guide/grasshopper/thermal.

Rutten, David. 2013. *Grasshopper* (version 0.9.0014). McNeel. http://www.grasshopper3d.com/.

Schön, Donald. 1983. *The Reflective Practitioner: How Professionals Think in Action.* Basic Books.

Steinfeld, Kyle, Pravin Bhiwapurkar, Anna Dyson, and Jason Vollen. 2010. "Situated Bioclimatic Information Design: a New Approach to the Processing and Visualization of Climate Data." In *Proceedings of the ACADIA '10 Conference*. New York, NY: Association for Computer Aided Design in Architecture.

Steinfeld, Kyle, Stefano Schiavon, and Dustin Moon. 2012. "Open Graphic Evaluative Frameworks - A Climate Analysis Tool Based on an Open Web-based Weather Data Visualization Platform." In *Proceedings of the 30th International Conference on Education and Research in Computer Aided Architectural Design in Europe*, 1:675–683. Prague, Czech Republic: eCAADe.

Turkle, Sherry. 2009. *Simulation and Its Discontents*. London, England: The MIT Press.

Visser, Willemien. 2006. *The Cognitive Artifacts of Designing.* Lawrence Erlbaum Associates.

KYLE STEINFELD is an Assistant Professor specializing in digital design technologies in the Department of Architecture at University of California Berkeley, where he teaches undergraduate and graduate design studios, core courses in architectural representation, and advanced seminars in digital modeling and visualization. Professionally, he has worked with and consulted for a number of design firms, including Skidmore Owings and Merrill, Acconci Studio, Kohn Petersen Fox, Howler/Yoon, and Diller Scofidio Renfro. His research interests include collaborative design technology platforms and bioclimatic design visualization, and he holds a Masters of Architecture from MIT and a Bachelor's Degree in Design from the University of Florida.

BRENDON LEVITT is a licensed architect and holds architecture degrees from UC Berkeley and Yale University. He is an associate at Loisos + Ubbelohde, where he serves as project manager, modeler, and designer for a wide range of high-performance buildings. His work and research has centered around thermal comfort and energy modeling as well as daylighting and electric lighting design. In addition, Mr. Levitt writes and lectures on sustainable design and the synthesis of contemporary culture, sensory space, and new technology. Mr. Levitt is faculty at UC Berkeley and California College of the Arts where he teaches building technology and design courses related to sustainable design.