

# MATERIAL LOGIC

Amanda S. Bruot

A thesis  
submitted in partial fulfillment of the  
requirements for the degree of

Master of Science in Architecture

University of Washington

2013

Committee:

Kimo Griggs

Brian R. Johnson

Program Authorized to Offer Degree:

Department of Architecture

© Copyright 2013

Amanda S. Bruot

University of Washington

**ABSTRACT**

Material Logic

Amanda S. Bruot

Chair of the Supervisory Committee:

Professor Kimo Griggs

Department of Architecture

The practice of architecture requires that materials be considered throughout the design process. Considerations range in complexity from aesthetic decisions, to those dependent upon physical properties. Although materials are integral to the design process, 3D modeling programs are limited in their ability to address the complexities of materials. Instead, their utility lies in the description of geometry, while relying upon the user's experience to inform material considerations. This places architects in the challenging position of designing physical artifacts, while working in an environment that permits the creation of geometry that may lack an analogue in the physical world.

This thesis seeks to expand the potential of 3D modeling through the incorporation of material information. Material Logic is an online database tool that facilitates the construction of material aware models in Grasshopper, a plug-in for Rhinoceros. Through this tool, material information can be saved, and then assembled into custom material components for download, providing drag and drop access to shared material information for the digital design environment.



## TABLE OF CONTENTS

List of Figures .....	ii
Acknowledgments .....	iii
Introduction .....	01
Abstraction .....	03
Geometry + Materials .....	07
Parametric Design .....	11
Case Studies .....	13
Material Logic .....	17
Custom Material Components .....	19
Material Logic Website .....	29
Examples .....	35
Conclusion .....	43
Bibliography .....	46
Image Credits .....	49
Appendix A .....	50

## LIST OF FIGURES

Figure 01	Geometric abstraction used to represent an interior partition .....	05
Figure 02	Assembly of the partition .....	05
Figure 03	Factors necessary for beam sizing .....	09
Figure 04	Curved-crease paper sculptures .....	09
Figure 05	Robert Corser's Wrap Chair .....	14
Figure 06	Stress analysis (FEA) of Wrap Chair .....	14
Figure 07	Material ConneXion database .....	16
Figure 08	MatWeb database .....	16
Figure 09	Custom material component .....	18
Figure 10	Five "chunks" of the GHX document & VB component editing process .....	21
Figure 11	Modification to the "Container" elements .....	22
Figure 12	"Parameter Data" additions and modifications .....	23
Figure 13	Modifications to "OutputParam" element .....	24
Figure 14	Sections of PHP script .....	27
Figure 15	Material Logic Contribute page .....	31
Figure 16	Material Logic Material page .....	31
Figure 17	Material Logic Discussion page .....	33
Figure 18	Results of deflection check using Hem-Fir #2 .....	36
Figure 19	Sized beam versus reference geometry .....	37
Figure 20	Custom cast acrylic component .....	38
Figure 21	Connector length versus number of connector links .....	39
Figure 22	Live hinge in acrylic .....	41
Figure 23	Physical test at 100% yield strength .....	41

## ACKNOWLEDGEMENTS

I would like to express my gratitude to my thesis committee members, Kimo Griggs and Brian R. Johnson, for their useful comments, remarks, and guidance throughout the thesis process and my graduate education. I would also like to thank my friends, classmates, and family for all their encouragement and support.





## CHAPTER 01 / Introduction

The practice of architecture requires that materials be considered throughout design. These considerations range in complexity from aesthetic decisions to those dependent upon physical properties. In current architectural practice these considerations are often not addressed until after geometry has been established. However, materials are inherently linked to the development and realization of architectural form.

In part, the ambiguous role of materials in geometric development is due to the limited ability of digital design programs to address the complexities and performance of materials. Instead, their strength lies in the description of geometry, while relying upon the user to understand the materiality of their design. This places architects in the challenging position of designing physical artifacts, while working in an environment that permits the creation of geometry that cannot be realized in the physical world. Although materials have been minimized in the current architectural design process, new software and paradigms for design are emerging that could alter this trend. Through the use of computational and parametric design tools, information

can be integrated into the digital design environment.

This data can then be used to establish parameters and relationships that can influence early design decisions and the development of geometry.

This thesis explores the relationship between geometry and materials, and seeks to expand the potential of digital design through the incorporation of material information.

Material Logic also promotes the growth of a material-aware design culture through development of an online community resource.

## CHAPTER 02 / Abstraction

Due to the complexities and scale of architectural design, abstractions, such as drawings and models, are used by architects to represent and communicate design intent. Instead of developing a concept as an actual instance, geometry is used to represent the materials, assemblies, and details that govern construction. Because these abstractions simplify the complexities of architecture, design concepts can be explored without being overwhelmed by the constraints of the actual built environment. This is important for the development and understanding of the design's formal characteristics and spatial relationships. However, this approach relies on the architect to understand and address the realities of the design that have been removed for simplification.

Take for example the drawing or modeling of an interior partition. Instead of constructing the partition as an assembly, a rectangular box is used to represent the wall. This level of abstraction is useful for exploring geometric relationships within a space. However, if the physical construction and materiality has not been addressed, there is no guarantee

that the drawn geometry will function as intended once constructed. Furthermore, the primitive rectangular geometry lacks the ability to carry information such as acoustic, thermal, or other performative characteristics that are necessary to evaluate the design in a comprehensive manner. All of these considerations are left to the architect, and must be made based on their previous knowledge and experience with materials and assemblies.

Over time, digital design tools have evolved to assist in handling the complexities of the architectural design process. Most recently, simulation tools have been developed that allow for the evaluation of a design while remaining in the digital environment. These computational tools apply information to a digital model to simulate various performative aspects of a given design. The architect must then decide, based on feedback from the tool, how the design must be modified to meet the intended performance criteria.

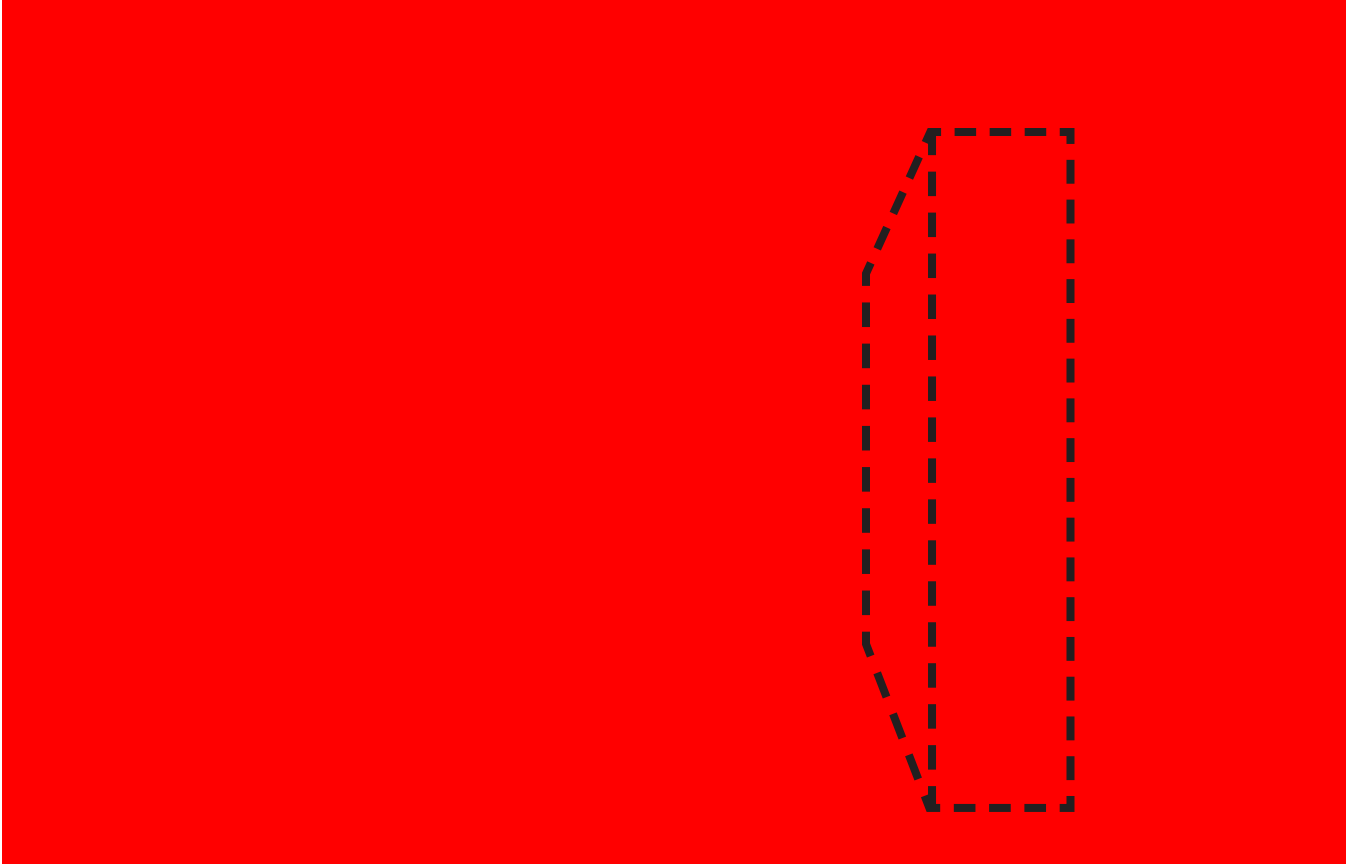
Although powerful, the accuracy of simulation software is dependent upon the precision of the modeled geometry and the information provided. As a result, simulation is generally more appropriate for the late stages of design. While these tools do assist architects in addressing the realities of a design, seldom are material decisions developed at the same time as the geometry. Instead, primacy is given to the creation of geometry, and "material is rarely examined beyond its aesthetic or technological capabilities to act as a servant to form" (Thomas, 2007).

## FIGURE 01

Geometric abstraction used to represent an interior partition

## FIGURE 02

Assembly of the partition





## CHAPTER 03 / Geometry + Materials

Although precedence is given to geometry in the architectural design process, geometry and materials are inherently linked. Every window, wall, floor, and ceiling in the built environment is constructed from materials that once existed as a geometric abstraction on paper. The translation from drawn to built only works if there is an analog for the drawn geometry. Take for example a wall that is curving in plan. The wall can easily be drawn or modeled as a curved geometric representation. However, without specific knowledge of the intended construction, there is no guarantee that the curved geometry is achievable, especially when flat sheet goods are desired for use in construction. Realization of the drawn geometry instead relies upon the architects understanding of material properties and how they will react when bent into the curvilinear form.

Just as material knowledge is necessary for understanding how geometry will function, geometry is a critical element in understanding the materiality of a design. As previously mentioned, digital design tools use geometric abstractions to convey design intent. As a result, digital drawings and models already contain all the geometric data required for the assessment of materials.

For example, in the process of sizing a wooden floor beam, geometry is necessary to determine the appropriate beam size for the design. Geometric information, including the span of the framing member and the area it is required to support, can be extracted from either a framing plan or a three-dimensional model. Once this data is obtained, and design loads are applied, all that is missing to evaluate the beam design is material information. Unfortunately, this information is not readily available in most design software, and as a result, the beam sizing cannot be performed.

Unlike architects who often work through abstraction, artists and craftspeople tend to work directly with materials. This allows the relationship between a material and the resulting form to be more fluent, as feedback is provided directly to the maker while working. Instead of establishing a form and then applying materials, this approach allows the material and its properties to influence geometric development. Examples of this can be seen in the curved-creased paper sculptures created by students in Josep Albers introductory design class taught at the Bauhaus. Albers taught the class entirely through the creation of paper models, in an effort to teach students about the inherent relationship between materials and geometry. In figure 04, the circular shape of the paper, along with the alternating concentric circle folding pattern, works in conjunction with paper's inherent malleability to automatically achieve its resulting saddle curve form (Demaine et al., 2011).

Due to the scale of architecture, architects do not usually have the option to work directly with the materials of their design. Instead, geometric abstractions are necessary to simplify the scale and complexities of architecture for design development. Over time, this simplification has caused a rift in the relationship between geometry and materials. In practice,

**FIGURE 03**

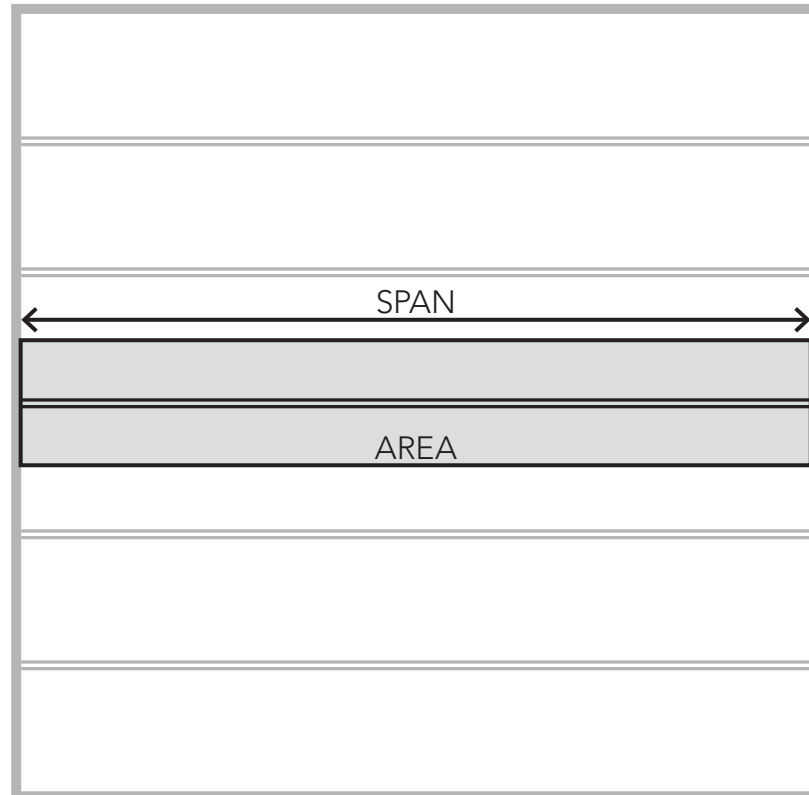
Factors necessary  
for beam sizing

**FIGURE 04**

Curved-crease  
paper sculptures



# MATERIAL?



$$\text{TOTAL LOAD} = \text{DEAD LOAD} + \text{LIVE LOAD} \times \text{AREA}$$



precedence has been given to geometry, and the importance of materials has been minimized. In order for the material/geometry relationship to be explored, as it is in the art and craft worlds, materials should be incorporated into digital design tools as data.

## CHAPTER 04 / Parametric Design

Technological advancements have expanded digital design capabilities beyond serving solely as a means of representation. Computation, or the use of mathematical or logical problem solving methods, can now be used to enhance the digital design process, leveraging the designer's ability to define dependency and causal relationships within the design. Computation allows designers to effectively organize and utilize vast amounts of data to inform design decisions and evaluate iterations.

Computation also permits the integration of information processing into the digital design tools commonly used in the early stages of design. This is most frequently done through the use of programs that employ parametric design methodologies. Here, geometry is constructed based on a series of relationships and constraints that provide a framework for the design. While this framework is most frequently defined by spatial relationships, other information such as material properties can be used.

For example, to design the curved wall discussed in the previous section, the minimum-bending radius of gypsum

board could be established as a constraint in the parametric model. Doing so would ensure that the drawn geometry is achievable when constructed in the physical world.

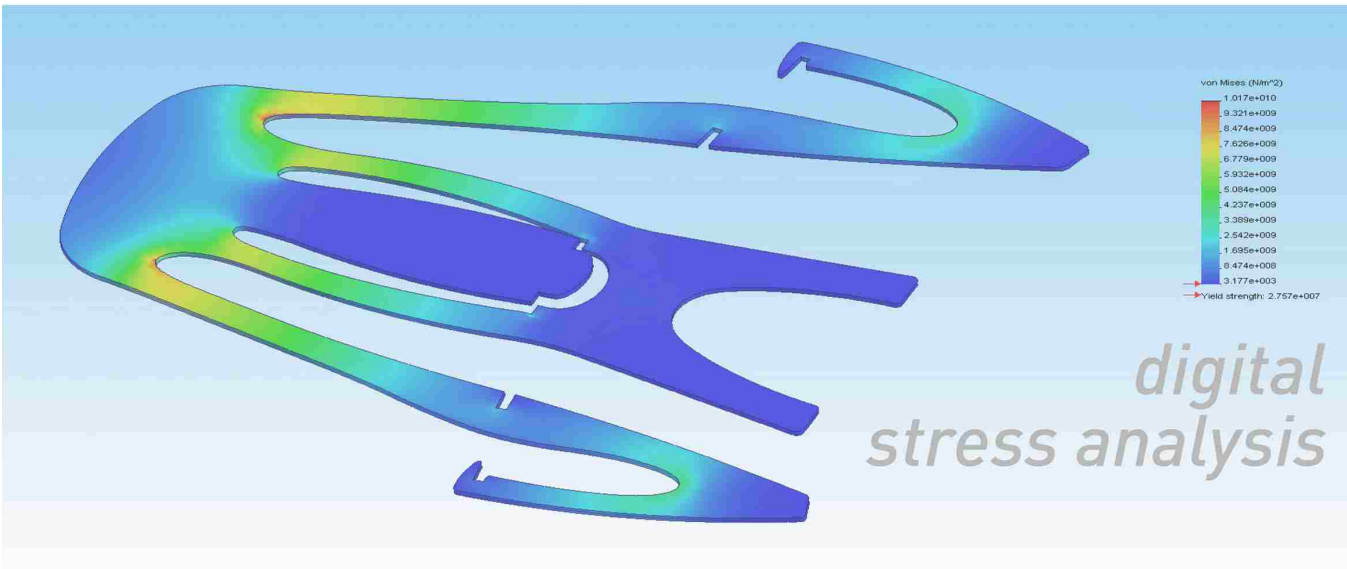
Furthermore, since a parametric relationship will remain intact as the design changes or evolves, the evaluation of the geometry will persist, confirming the integrity of the design.

Many digital design tools either incorporate or interface with software that provides parametric modeling capabilities.

If designers apply these parametric tools from the initial stages of design, materials and geometry can be developed simultaneously.

## CHAPTER 05 / Case Studies

Although architectural practice is predominately based upon a design approach that prioritizes the creation of form, architects have recently been exploring the potentials that materiality can have on the outcome of design. Computational tools and parametric design methodologies have enabled much of the research in this area. For example, finite element analysis (FEA), predominately used by engineers, has gained purchase in the architectural practice for the evaluation of components and assemblies. In the *Induced-Stress Joinery Project*, Robert Corser used FEA to evaluate surface stresses in his furniture (figure 06). His designs explore methods for bending flat sheet goods into stable forms. To avoid material failure it was critical to understand how the stresses propagate through the material when bent into its final position. Through a combination of physical prototyping and FEA, Corser was able to establish a parametric framework for geometry development. In locations where large stresses were generated, the furniture shape was modified to accommodate or redistribute the loads, thus affecting the final geometric outcome (Corser, 2011).



**FIGURE 05**

Robert Corser's  
Wrap Chair

In addition to applied research, efforts have also been made to catalogue material information for architects and designers. Material ConneXion, is a subscription based materials library, that specializes in innovative sustainable materials. For subscribers there are both physical and online libraries that contain, images, descriptions, usage characteristics, and manufacturing information. While Material ConneXion's collection of sustainable materials is well curated, the descriptions are not detailed enough to provide users with information to perform any calculations. Instead, materials are simply rated in a comparative manner relative to other materials in the database, e.g. low, medium, or high level of thermal conductivity ("Material ConneXion", n.d.).


MatWeb is another online library of materials and material properties. Instead of using a rating system to describe materials like Material ConneXion, the MatWeb library features detailed material property information for over 98,000 materials. Data can be accessed by anyone via their website, however with the paid subscription users can export data sheets to a variety of CAD/FEA programs, such as SolidWorks and ANSYS ("Online Materials Information Resource - MatWeb," n.d.). Unfortunately, these programs are not commonly used by architects, especially not in the early stages of design where material information could be used to influence the development of geometry.

**FIGURE 06**

Stress analysis (FEA)  
of Wrap Chair

FUSE product page  
 dwell.materialconnexion.com/product2.aspx?ID=3721

Material Connexion® Enter Search Term Here recently viewed your tags Questions? Contact Us



**FUSE**  
 MC# 6236-01 Category Naturals

Engineered hardwood flooring. The flooring features a three-ply engineered wood containing a minimum of 75 percent recycled content with a veneer and backing separated by a sturdy core. It has high structural stability, acoustical insulation and offers installation directly over concrete. A natural oil finish derived from all-natural plant products rather than typical polyurethane coating makes damage to flooring less noticeable and easily restored through simple sanding or re-oiling. The company's floor-print program matches each square foot of flooring sold with an equal area of reforestation in Panama. The flooring is formaldehyde- and VOC-free, FSC-certified in some cases, and eligible for up to six points toward LEED certification. Applications are for commercial and residential hardwood floors. Both Daniel Libeskind (Royal Ontario Museum) and Philippe Stark (condominium development) recently specified this flooring for projects.

**Processing**  
 Injection Molding: No  
 Extrusion: No  
 Cold Pressing/Deep Drawing: No  
 Blow Molding: No  
 Thermoforming: No  
 Lamination: No  
 Printable: No  
 Stitchable: No  
 Rotomolding: No  
 Weldable: No  
 Wood Working Tools: Yes  
 Die cut: No  
 Metal Working Tools: Yes  
 Castable: No

**Usage Properties**  
 Cradle to Cradle: N/A  
 Fire resistance: High  
 Usage temperature: Low  
 Colorfastness: High  
 Wear Resistance: High  
 Water Resistance: High  
 Acoustics: Sound absorbing  
 Chemical Resistance: Medium  
 UV resistance: Medium  
 Scratch resistance: High  
 Outdoor use: No  
 Tear Resistance: N/A  
 Reflectivity: Light absorbing  
 Stain Resistance: High  
 Thermal Conductivity: Low

**Sustainability**  
 Biodegradable  
 Certified or stewardship sources  
 Reclaimed  
 Recycled (pre- or post-consumer)  
 Renewable  
 Waste materials

**Physical Properties**  
 Stiffness: Stiff  
 Structure: Closed  
 Impact Resistance: Good  
 Surface/Texture: Matte  
 Transparency: Opaque  
 Surface Hardness: Hard

**Dwell Materialism May 2010**  
 Return to MaterialConnexion.com/Dwell

← Previous Next →

This is a LIMITED DEMO of our online materials database. Get full access to the database and discover thousands more materials—Join Now!

**Contact Information**  
 Manufacturer: Relative Space  
 United States of America  
 Company Relative Space  
 Address 2 Bond Street  
 New York New York  
 10012  
 United States of America  
 Email tyler@relative-space.com  
 Phone +1 212 353 3370  
 Fax +1 212 353 3376  
 Website www.relative-space.com

**Sales**  
 Contact Tyler Greenberg  
 Email tyler@relative-space.com  
 Phone +1 212 353 3370  
 Fax +1 212 353 3376

© Material Connexion 2010. All Rights Reserved Services | Tools | Work | News | MATTER | About Us | Contact Us

Overview of materials for x  
 www.matweb.com/search/DataSheet.aspx?MatGUID=a5e93a1f1ff43bcba56ca51b8981f&ckck=1

MatWeb MATERIAL PROPERTY DATA Data sheets for over 98,000 metals, plastics, ceramics, and composites. Advertise with MatWeb REGISTER NOW

HOME • SEARCH • TOOLS • SUPPLIERS • FOLDERS • ABOUT US • FAQ • LOG IN

Searches: Advanced | Category | Property | Metals | Trade Name | Manufacturer | Recently Viewed Materials cast acrylic SEARCH

**CONNECTICUT PLASTICS** Custom Machined Acrylic Parts Click here to request a quote (203) 265-3299

**Overview of materials for Acrylic, Cast**

Categories: Polymer; Thermoplastic; Acrylic; Acrylic\_Cast

**Material Notes:** This property data is a summary of similar materials in the MatWeb database for the category "Acrylic, Cast". Each property range of values reported is minimum and maximum values of appropriate MatWeb entries. The comments report the average value, and number of data points used to calculate the average. The values are not necessarily typical of any specific grade, especially less common values and those that can be most affected by additives or processing methods.

**Vendors:** Lucite International: From standard to tailor-made acrylic bead grades, Lucite International helps you accelerate your new product development. Learn more about Lucite International acrylic services on Omnexus.  
 Click here to view all available suppliers for this material.  
 Please click here if you are a supplier and would like information on how to add your listing to this material.

Printer friendly version Download as PDF Download to Excel (requires Excel and Windows) Export data to your CAD/FEA program Add to Folder My Folder 0/0

Physical Properties	Metric	English	Comments
Density	1.18 - 1.19 g/cc	0.0426 - 0.0430 lb/in³	Average value: 1.19 g/cc Grade Count:10
Water Absorption	0.130 - 0.480 %	0.130 - 0.480 %	Average value: 0.296 % Grade Count:8
Water Absorption at Saturation	1.10 - 2.10 %	1.10 - 2.10 %	Average value: 1.90 % Grade Count:5
Moisture Expansion	0.500 %	0.500 %	Average value: 0.500 % Grade Count:3
Moisture Vapor Transmission	55.0 cc-mm/m²·24hr-atm	140 cc-mil/100 in²·24hr-atm	Average value: 55.0 cc-mm/m²·24hr-atm Grade Count:3

Mechanical Properties	Metric	English	Comments
Hardness, Rockwell M	94.0 - 102	94.0 - 102	Average value: 98.2 Grade Count:3
Ball Indentation Hardness	175 MPa	25400 psi	Average value: 175 MPa Grade Count:3
Tensile Strength, Ultimate	62.0 - 83.0 MPa	8990 - 12000 psi	Average value: 77.0 MPa Grade Count:10
	40.0 - 110 MPa	5800 - 16000 psi	Average value: 75.0 MPa Grade Count:3
	@Temperature=40.0-70.0 °C	@Temperature=40.0-158 °F	
Elongation at Break	4.00 - 5.50 %	4.00 - 5.50 %	Average value: 5.02 % Grade Count:9
Modulus of Elasticity	2.80 - 3.30 GPa	406 - 479 ksi	Average value: 3.17 GPa Grade Count:4
Flexural Yield Strength	105 - 116 MPa	15200 - 16800 psi	Average value: 113 MPa Grade Count:6
Flexural Modulus	2.96 - 3.30 GPa	429 - 479 ksi	Average value: 3.16 GPa Grade Count:3
Compressive Yield Strength	110 - 124 MPa	16000 - 18000 psi	Average value: 114 MPa Grade Count:4
Poissons Ratio	0.370	0.370	Average value: 0.370 Grade Count:3
Shear Modulus	1.70 GPa	247 ksi	Average value: 1.70 GPa Grade Count:3
Izod Impact, Notched (ISO)	1.60 kJ/m²	0.761 ft-lb/in²	Average value: 1.60 kJ/m² Grade Count:3
Charpy Impact Unnotched	1.20 - 2.17 J/cm²	5.71 - 10.3 ft-lb/in²	Average value: 1.57 J/cm² Grade Count:5
Coefficient of Friction	0.450 - 0.800	0.450 - 0.800	Average value: 0.583 Grade Count:3



**FIGURE 07**

Material  
ConneXion  
database

**CHAPTER 06 / Material Logic**

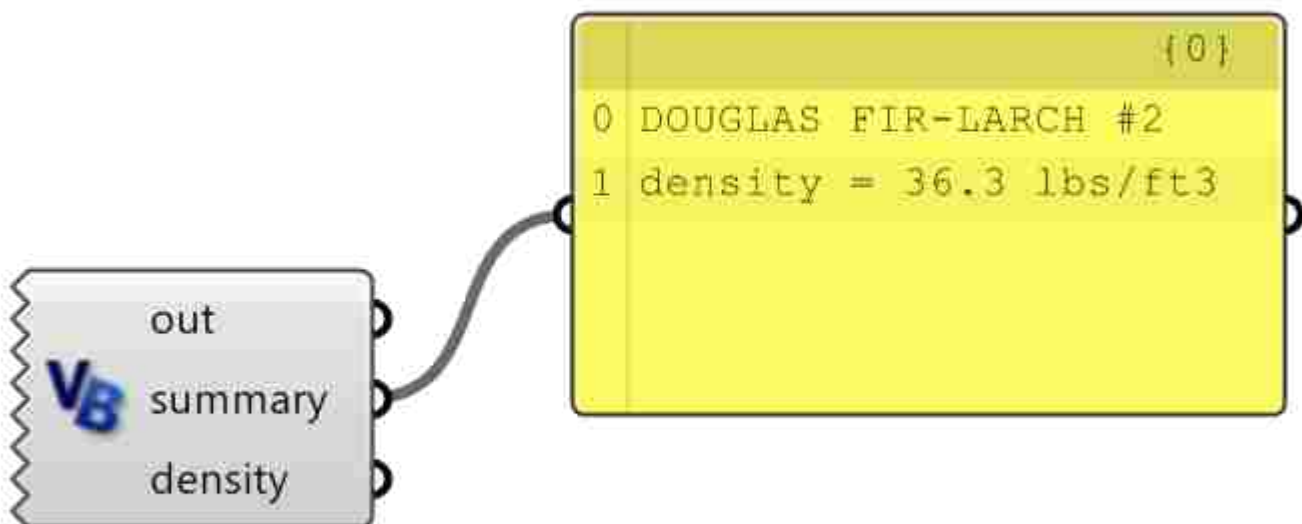
Material Logic seeks to expand the potential of existing three-dimensional modeling software through the incorporation of material information. It does so by assembling user contributed material data and creating a conduit for its incorporation into the digital tools used by architects. Material data includes any property or characteristic that describes a material and/or its behavior. Current three-dimensional modeling programs are limited in their ability to address this information beyond appearance properties (color, visual texture, etc.). Instead, their utility lies in the creation and description of geometry, while relying on users to understand the materiality of their designs.

Material Logic is an online database tool that facilitates the construction of material aware models in Grasshopper, a widely used parametric design plug-in for Rhinoceros, a three-dimensional modeling environment from Robert McNeel and Associates. Through the Material Logic database, material information can be saved, and then assembled into custom material components for download, providing Grasshopper users with drag and drop access to shared material information.

**FIGURE 08**

MatWeb  
database

The development of Material Logic is outlined in the following three chapters. First, the process of creating custom material components is addressed. This was achieved by (1) taking advantage of the fact that one Grasshopper file format (GHX) uses text-based XML files, and (2) utilizing Grasshopper's pre-existing Visual Basic (VB) script component as an "entry point" for material data. This chapter outlines how the GHX document was modified to create the component, as well as the PHP script used to automate this process. The next chapter focuses on the Material Logic website, its development, and motivations behind its creation and features. Finally, the last chapter outlines two examples that illustrate the use of custom material components and the Material Logic website.



## CHAPTER 07 / Custom Material Components

Grasshopper definition files can be saved as XML (Extensible Markup Language). Grasshopper XML documents (GHX) are human-readable text files that store definition data in a structured format. Like all XML files, GHX documents only store information. They require Grasshopper to be displayed as a definition. However, their contents can be read as text and manipulated in a text editor. For the purpose of this thesis, a VB scripting component, in its XML form, was manipulated to create a custom component “container” that can provide material information to a Grasshopper definition. This section outlines the process of editing GHX documents to create custom material components, as well as the author’s development of a PHP script used to automate this process.

### Editing GHX Documents

GHX documents organize the data they contain in a tree structure common to all XML files. The document starts with a root element, which is the parent of all other elements, and branches to contain any number of child and sub-child elements. Any element within the tree can contain data, as well as attributes that describe the element.

#### FIGURE 09

Custom material component

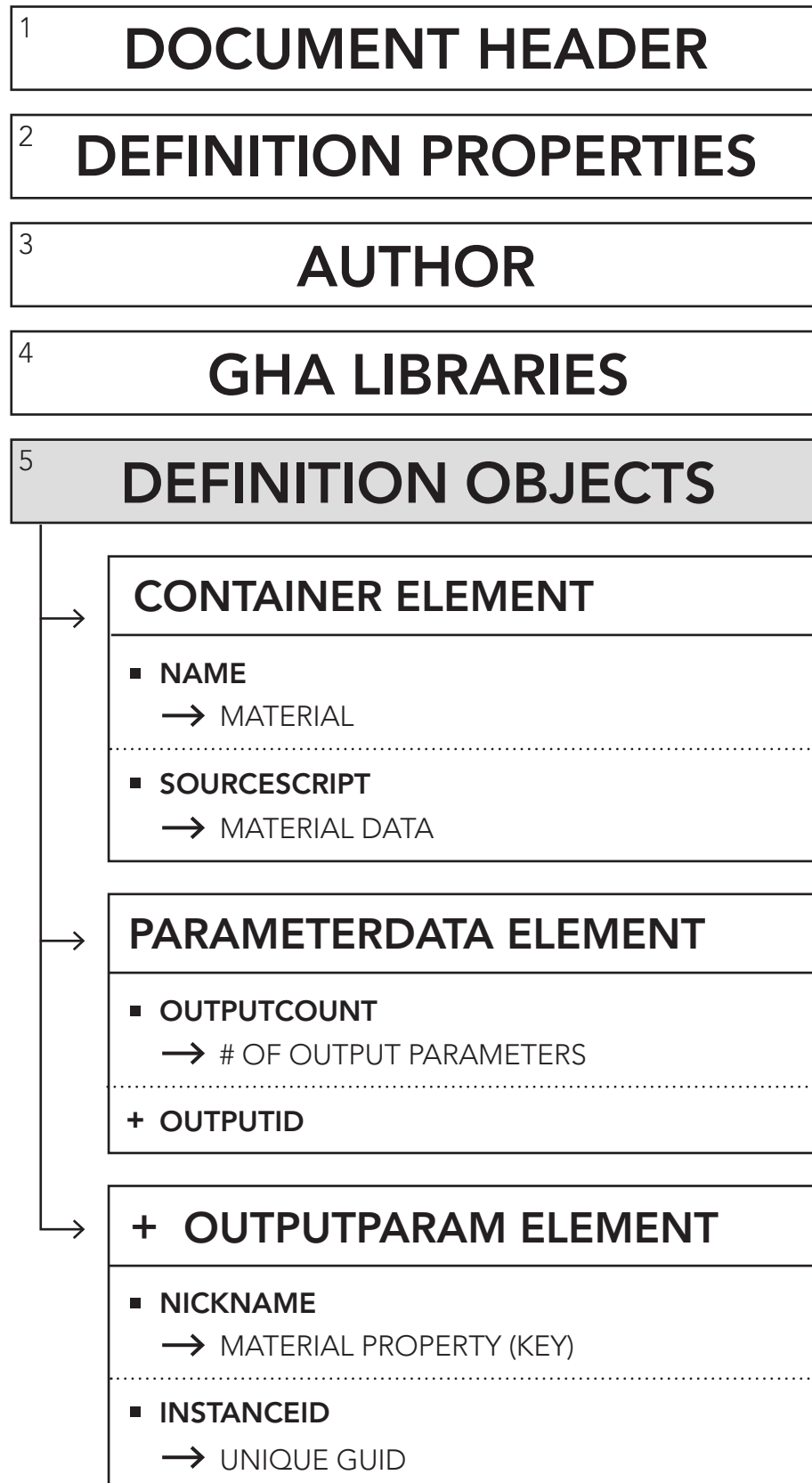
In the case of GHX documents, the root element is named "archive", which refers to the Grasshopper dictionary archive defined by the GH\_IO.dll. Grasshopper uses the GH\_IO.dll to write all of its files, including the GHX file format. From the root element, the rest of the GHX document is built in blocks called "chunks". The body of the document contains five "chunks", all of which contain "items", and/or additional "sub-chunks". The first four "chunks" of data set up the GHX document, while the last one, named "definition objects", contains the actual definition. This is the only "chunk" of XML that requires editing in order to customize an existing component or edit the data it holds.

In order to incorporate information into an existing Grasshopper component, it was necessary to start with one that was originally designed to hold data. Multiple components could have been used, including any of the scripting components, a panel, or even a value list; however in this thesis a VB scripting component was chosen. Scripting components proved to be the most useful for two reasons. First, they do not require any inputs. Since the goal was to create a component that carries information, rather than inheriting it from another component, there was no use for an input parameter. Second, scripting components can expand to contain any number of outputs. This was critical for ease of use, as it allowed for information contained within the component to exist as separate, clearly identifiable streams of data, as opposed to being identified by their position in a list.

Three elements define each VB scripting component. These elements are specified as the "Container", "ParameterData", and "OutputParam". (Note: Typically a VB scripting component also has an "InputParam" element. However, since no input parameters are required for the custom material

## FIGURE 10

Five "chunks"  
of the GHX  
document &  
VB component  
editing process



components, this element has been omitted.) Data within each of these sections needs to be modified in order to customize the scripting component. The following description outlines the process used in this thesis to create custom components containing material information.

Information in the "Container" element defines the component and the content it holds. To create a custom material component, the component's name and content need to be modified in this section. The name is changed to identify the material, and the content is modified to hold information describing the material. To change the component's name, first identify the child element of "Container" called as "Name". Remove the existing content (the text between the opening and closing tags) and insert a name of your choosing. In order to change the content of the VB component to include our material data, the data must be written as the "script" the component holds. The script is located in the element named "SourceScript". All data must be written as "key = value" pairs and separated by a space.

```
<chunk name="Container">
  <items count="6">
    <item name="ScriptSource" type_name="gh_string" type_code="10">
      density=31.8

      Dim A As New List (Of String)
      A.add("HEM-FIR #2")
      A.add("density = 31.8 lbs/ft3 ")
      summary = A
    </item>
    <item name="ReferenceCount" type_name="gh_int32" type_code="3">0</item>
    <item name="Name" type_name="gh_string" type_code="10">HEM-FIR #2</item>
    <item name="NickName" type_name="gh_string" type_code="10">VB</item>
    <item name="Description" type_name="gh_string" type_code="10">
      A VB.NET scriptable component
    </item>
    <item name="InstanceGuid" type_name="gh_guid" type_code="9">
      e88bd518-1b24-455a-b841-9fee79ac0a3b
    </item>
  </items>
  [...]
</chunk>
```

**FIGURE 12**

"Parameter Data"  
additions and  
modifications

```

<chunk name="ParameterData">
  <items count="4">
    <item name="InputCount" type_name="gh_int32" type_code="3">0</item>
    <item name="OutputCount" type_name="gh_int32" type_code="3">3</item>
    <item name="OutputId" index="0" type_name="gh_guid" type_code="9">
      3ede854e-c753-40eb-84cb-b48008f14fd4
    </item>
    <item name="OutputId" index="1" type_name="gh_guid" type_code="9">
      8ec86459-bf01-4409-baee-174d0d2b13d0
    </item>
    <item name="OutputId" index="2" type_name="gh_guid" type_code="9">
      8ec86459-bf01-4409-baee-174d0d2b13d0
    </item>
  </items>
  [...]
</chunk>

```

Once the content of the component has been altered to hold material data, the output parameters of the component must be modified to accommodate each piece of data written in the script. A component's output parameters are established in the element named "ParameterData". This is where the count of input and output parameters is defined, and where each parameter is assigned a globally unique identifier, or GUID.

The count of output parameters must be modified to match, the data that component now holds. This is done by changing the number in "OutputCount" to your total number of data items. Then, an "OutputId" element, and the GUID it contains, must be added for each output parameter.

Although the output parameters are established in the previous section, the formal properties of each parameter have yet to be defined. This is done in the "OutputParam" element. The GHX document must contain an "OutputParam" element for each output in the component. Two items within this element also need to be modified. In the first step of editing the VB component, material data was written as "key = value" in the component's script. The component's

## FIGURE 11

Modification to the "Container" elements

output parameters correspond to these key/value sets of material data. In order for the component to output each data item, the name of the output MUST match the name of the key (material property) specified in the script. The name of each output can be changed in "Nickname", within the "OutputParm" element. The final step in the editing process is to assign each output parameter a GUID in "InstanceId". No other output parameter can have this GUID. It MUST be unique within the VB script component.

```
<chunk name="OutputParam" index="2">
  <items count="6">
    <item name="Name" gh_string="gh_string" type_code="10">density</item>
    <item name="Nickname" gh_string="gh_string" type_code="10">
      density
    </item>
    <item name="Description" gh_string="gh_string" type_code="10">
      Output_parameter_density
    </item>
    <item name="InstanceGuid" gh_string="gh_guid" type_code="9">
      9beb07fe-ad43-40a6-ab45-b591cd82d65d
    </item>
    <item name="Optional" gh_string="gh_bool" type_code="1">>false</item>
    <item name="SourceCount" gh_string="gh_int32" type_code="3">0</item>
  </items>
  <chunks count="1">
    <chunk name="Attributes">
      <items count="3">
        <item name="Pivot" type_name="gh_drawing_pointf" type_code="31">
          <X>10</X>
          <Y>10</Y>
        </item>
        <item name="Bounds" type_name="gh_drawing_rectanglef" type_
          code="35">
          <X>10</X>
          <Y>10</Y>
          <W>10</W>
          <H>10</H>
        </item>
      </items>
    </chunk>
  </chunks>
</chunk>
```

Once these steps have been completed, the GHX document can be saved and opened in Grasshopper. The file should



contain the customized VB component. The component and the material information it contains can now be accessed and used in any definition. Further, GHX files may be dragged from the desktop and dropped into existing definitions in order to merge their components into a new document.

### Automating the Editing Process

The process of creating custom material components as outlined above was automated using the web-based scripting language PHP: Hypertext Preprocessor (PHP) ("PHP: Hypertext Preprocessor", n.d.). This was necessary to realize the larger goals of the project. The objective was not just to create custom material components, but to do so on the fly, to quickly provide drag and drop access to material information for the Grasshopper environment. Manually performing all of the edits took too long, and the results were too inconsistent for the method to meet the objectives of this thesis.

The script was written in PHP for two reasons. First, PHP has an extension called SimpleXML that is designed to use and manipulate XML data. The extension includes all the pre-defined functions needed to edit the GHX document to create the custom components. Secondly, PHP is commonly used to interact with MySQL. MySQL is an open source database management system ("MySQL Cluster 7.3 GA", n.d.), which is commonly used to create and maintain online data. A database was the ideal location to organize and store the material information the custom components would hold. PHP can connect to and query a MySQL database through a set of functions. These functions allow information contained within the database to be selected, manipulated, or even deleted. Additionally, one can add information to the database using a PHP function.

#### FIGURE 13

Modifications to  
"OutputParam"  
element

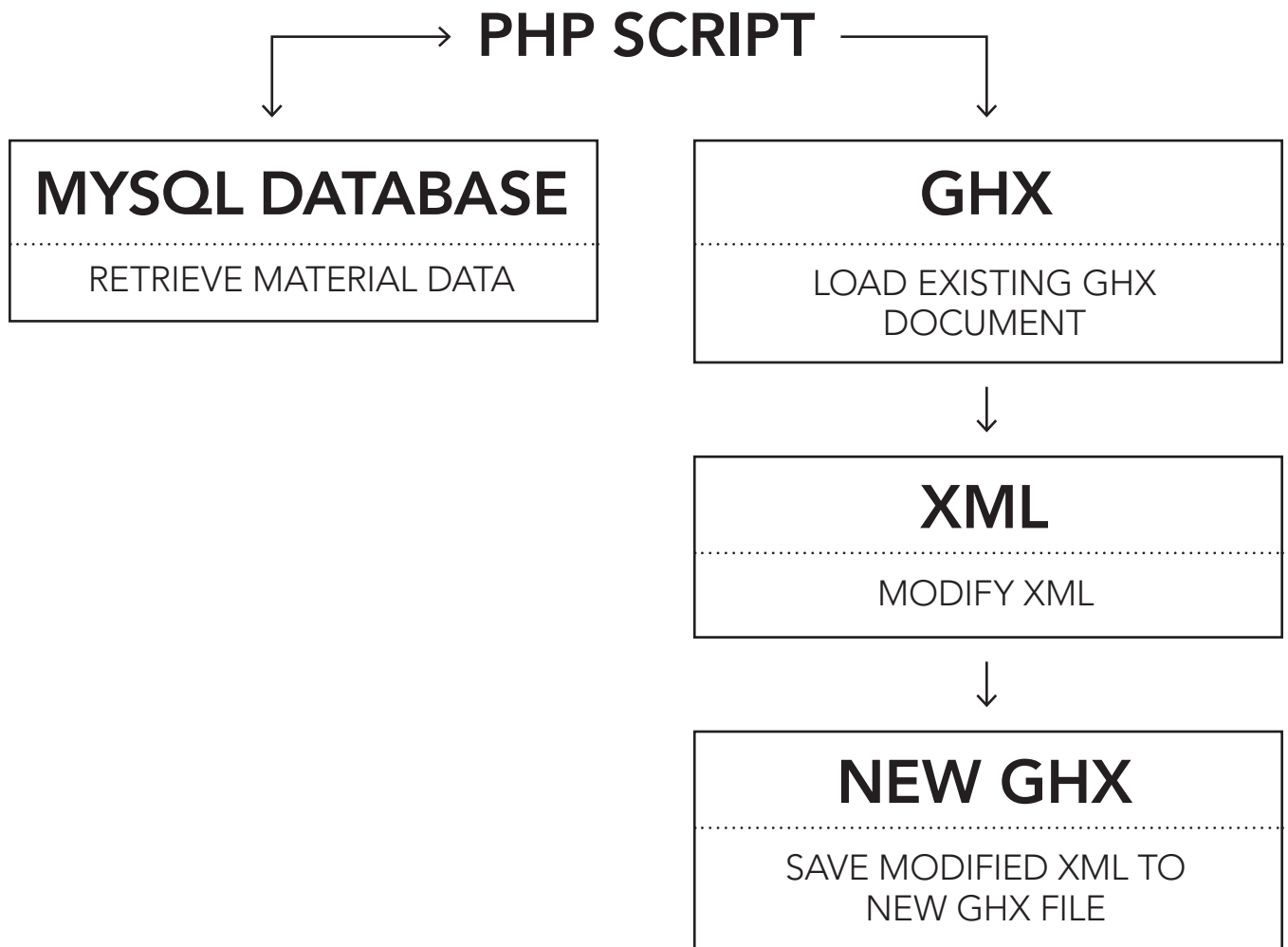
The database of material information created for this thesis contains 5 tables: "mat\_id", "cat\_id", "prop\_id", "unit\_id", and "material\_db". For organizational purposes, each entry to the database required a material, property, unit of measure, and material category to be defined. These fields were assigned an identification number in their respective tables. If an entry already existed for a data item, an identification number was not assigned, nor was a new entry created. Instead, its existing id number was used for identification. Once the id numbers were assigned, the entry was added to the "material\_db" table. Each row in this table contains only the identification numbers of the entry's material, property, units, and category. This database table can easily be searched to find all the entries for a specific material, all the materials that have information on a specific property, or even all the entries in the database for a specific material category.

The full PHP script used to create the custom material components can be found in Appendix A, however a summary of the script, and its vital elements, will be discussed in the subsequent text.

The script can be broken into four sections, starting with the retrieval of data from the MySQL database. The user selects the specific data (material properties) each custom component holds via an HTML form on the Material Logic website (outlined in following chapter). A component can hold any number of material properties desired by the user.

In the next section of the script, a GHX file is loaded using the SimpleXML function `simple_load_file()`. This file contains only the VB scripting component template that will be modified to hold material information along with a panel component to display a summary of the data. Once the file is loaded,

**FIGURE 14**  
Sections of PHP  
script



the XML needs to be edited. The script follows the steps outlined in the previous section for editing the “Container”, “ParameterData”, and “OutputParam” elements, however it does so using SimpleXML. For example, to change the name of the component, the value of the item “Name” within “Container” needs to be modified. Using SimpleXML, this is done with the following line of code:

```
$xml->chunks[0]->chunk->chunks->chunk[4]->chunks->chunk->chunks->chunk->items->item[2] = $material;
```

The variable “\$xml” holds the GHX document previously loaded using the `simple_load_file()` function. Any element in this document can be accessed by specifying a path to its location. In this example, “`chunks[0]->chunk->chunks->chunk[4]->chunks->chunk->chunks->chunk->items->item[2]`” is the (admittedly illegible) path to the item “Name” within “Container”. Once the path to the element has been specified, a new value can be assigned to the element. In this case, the new component name, held in the variable “\$material”, is assigned. The SimpleXML functions `addChild()` and `addAttribute()` were also used in the editing process to add all the necessary child elements and attributes for the custom material component.

After the all the XML modifications have been made, the script saves the XML as a new GHX file. This file can then be downloaded from the Material Logic website. The following chapter discusses the development of the website.

## CHAPTER 08 / Material Logic Website

The Material Logic website (<http://depts.washington.edu/aeworks/MaterialLogic/index.php>) was developed to be a community resource for designers looking to incorporate material information into their digital design process. From the website, material information can be selected and downloaded as a custom material component for use in Grasshopper. While the current database only contains data for a few materials, it is expected that additional material information will be generated by the Material Logic community. Registered users can contribute any information that describes a material and/or its behavior that they feel may be useful to others. This information is stored in the MySQL database (see previous chapter), and can easily be searched via the website. Additionally, the website acts as a platform for both sharing knowledge about the application of materials, and collecting feedback on specific properties. Users can discuss the innovative ways they are using materials and the custom material components in the discussion forum. Here, they can share both images and definition files to explain their process. By leveraging the knowledge and experience of many users, it is expected that the information contained in the forum can be a powerful resource of material applications.

Material information and forum discussions are available for anyone to access on the Material Logic website. However, only registered members can contribute information to the database and post to the discussion forum. The website is meant to be a resource to all, but the provenance of community-generated data can be problematic. By restricting data contributions to registered members, it is the author's hope that the information provided will be accurate. If not, incorrect or fabricated information can easily be tracked to the contributor and removed from the database.

The Material Logic website is divided into three sections: contribute, download, and discuss.

Material information can be added to the Material Logic database using the Contribute page. Only one material can be added at a time, but the number of property entries is expandable. For organizational purposes, all fields must be filled in before submitting the entry. Fields include category, material, property, value and units. These categories relate to the database tables specified in the previous chapter.

Material information is organized in the following hierarchy: (1) material, (2) property, and (3) value / units. As a result, mild steel is different from stainless steel, and hem-fir #1 is different from hem-fir #2. Once submitted, the user is directed to the Material page for the entry.

Every material entry has its own page within the website. From the Material page, information can be selected to create a custom material component for use in Grasshopper. These pages also serve as a useful library of material information for anyone to access. The structure of each Material page is identical; however, the information it contains is custom, as the database entries for each material are unique. Entries are

**FIGURE 15**

Material Logic  
Contribute page

**FIGURE 16**

Material Logic  
Material page

Material Logic

depts.washington.edu/aeworks/MaterialLogic/contribute.php

## MATERIAL LOGIC

AMANDA BRUOT  
SEATTLE, WA  
UNITED STATES  
UNIVERSITY OF WASHINGTON

LOG OUT

To contribute to the database please enter information in the form provided. Only one material can be added at a time, but the number of property entries is expandable. For organizational purposes, we do ask that all fields be filled in. Thanks for your contribution!

Note: We understand there may be slight discrepancies in values, however we reserve the right to remove false or fabricated information.

DISCUSSIONS | NEW POST | **CONTRIBUTE** | Material Search... >

**CONTRIBUTE TO MATERIAL DATABASE**

MATERIAL:  CATEGORY:

PROPERTY:  VALUE:  UNITS:

ADD  ADDITIONAL PROPERTIES

Material Logic

depts.washington.edu/aeworks/MaterialLogic/material.php?id=23

## MATERIAL LOGIC

AMANDA BRUOT  
SEATTLE, WA  
UNITED STATES  
UNIVERSITY OF WASHINGTON

LOG OUT

Select the properties you would like to use on the right and click submit. Once the information has been submitted, a custom material component containing the selected information will be created for you. Download the file and open in Grasshopper.

Disclaimer: This is a community-generated database of material information. We hope that our members are supplying accurate information, however please use at your own risk. If you suspect material information has been fabricated please contact us.

DISCUSSIONS | NEW POST | CONTRIBUTE | Material Search... >

**DOUGLAS FIR-LARCH #2**

PROPERTY	VALUE	UNITS	SELECT
density	36.3	lbs/ft3	<input checked="" type="checkbox"/>
modulus of elasticity	1700000	psi	<input checked="" type="checkbox"/>
allowable bending stress	1450	psi	<input checked="" type="checkbox"/>
allowable shear stress	95	psi	<input type="checkbox"/>

Select properties for download.

displayed in a table that is divided into property, value, and unit categories. Any number of properties can be selected from the table of available information. Once submitted, a custom material component containing the selected material properties will be created using the PHP script outlined in the previous chapter. A link for downloading the GHX file containing the custom component will then be displayed.

Users can share how they are using materials and the custom material components in the discussion forum. Each discussion has its own page, however the main forum contains a list of all forum posts, starting with the most recent. In the Discussion pages, users can describe their process, as well as share images and Grasshopper definition files. The forum is meant to provide users a place to learn, collaborate, and help others through the material prototyping process. Anything from questions to full explanations of a design process can be posted.

Two examples, using custom material components are outlined in the following chapter. A discussion post, including example Grasshopper definition files for each, can be found online.

**FIGURE 17**

Material Logic  
Discussion page



Material Logic

depts.washington.edu/aeworks/MaterialLogic/discussion\_topic.php?id=49

## MATERIAL LOGIC

The discussion forum is a place for users to share the creative ways they are using materials and the material logic database. Material knowledge is often learned through experience or through other people. It is our goal to provide an online space for users to share their knowledge and help others through the material prototyping process.

CONTACT

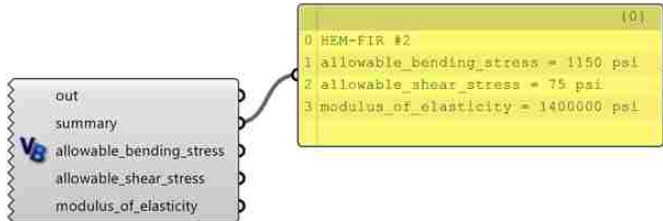
DISCUSSIONS REGISTER LOG IN Material Search... >

### BEAM CALCULATOR

by abruot  
on Jun 01, 2013 @ 2:02 pm

Beam sizing is a necessary process in architectural design. Although there are load and span charts available in design reference books, this calculation could be done during the digital design process using a computational tool like Grasshopper. For basic sizing all you need to know is the beam span, the imposed loads, and the beam's material characteristics.

I've created a Grasshopper definition that runs through the basic structural calculations for sizing a wood beam that is carrying a uniformly distributed load. This is by no means a replacement for an engineer, however it is great a way to more accurately represent beam geometry in your 3D models. The definition relies on knowing the beam material's allowable shear stress, allowable bending stress and modulus of elasticity. I've added these properties to the database for a variety of common wood species.



```

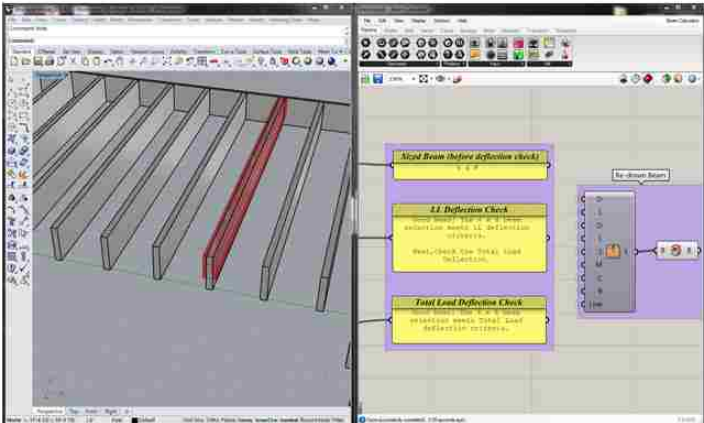
(0)
0 HEM-FIR #2
1 allowable_bending_stress = 1150 psi
2 allowable_shear_stress = 75 psi
3 modulus_of_elasticity = 1400000 psi

```

out  
summary  
allowable\_bending\_stress  
allowable\_shear\_stress  
modulus\_of\_elasticity

To use the definition, first download a custom material component containing the appropriate material properties, and drop it into the beam calculator. I've left an existing material component in the definition for reference as to how to properly feed the material information into the calculation. You must also input your load information. (Note: you must calculate your uniformly distributed live and dead loads.)

Then, starting from a BREP, the definition sizes the beam based on shear and moment calculations. Using that "sized beam" it performs a deflection check based on a selected deflection criteria. If the initial section does not meet the criteria, the next largest section that does, is chosen. The final sized beam is redrawn for you in the appropriate position.



This definition only works for uniformly distributed loads, however it could be pretty easily expanded to include point loads. Enjoy!

ATTACHED FILES:  
Beam Calculator.gh

REPLY

Choose Files No file chosen

Post



## CHAPTER 09 / Examples

### Wood Beam Calculator (Uniformly Distributed Load)

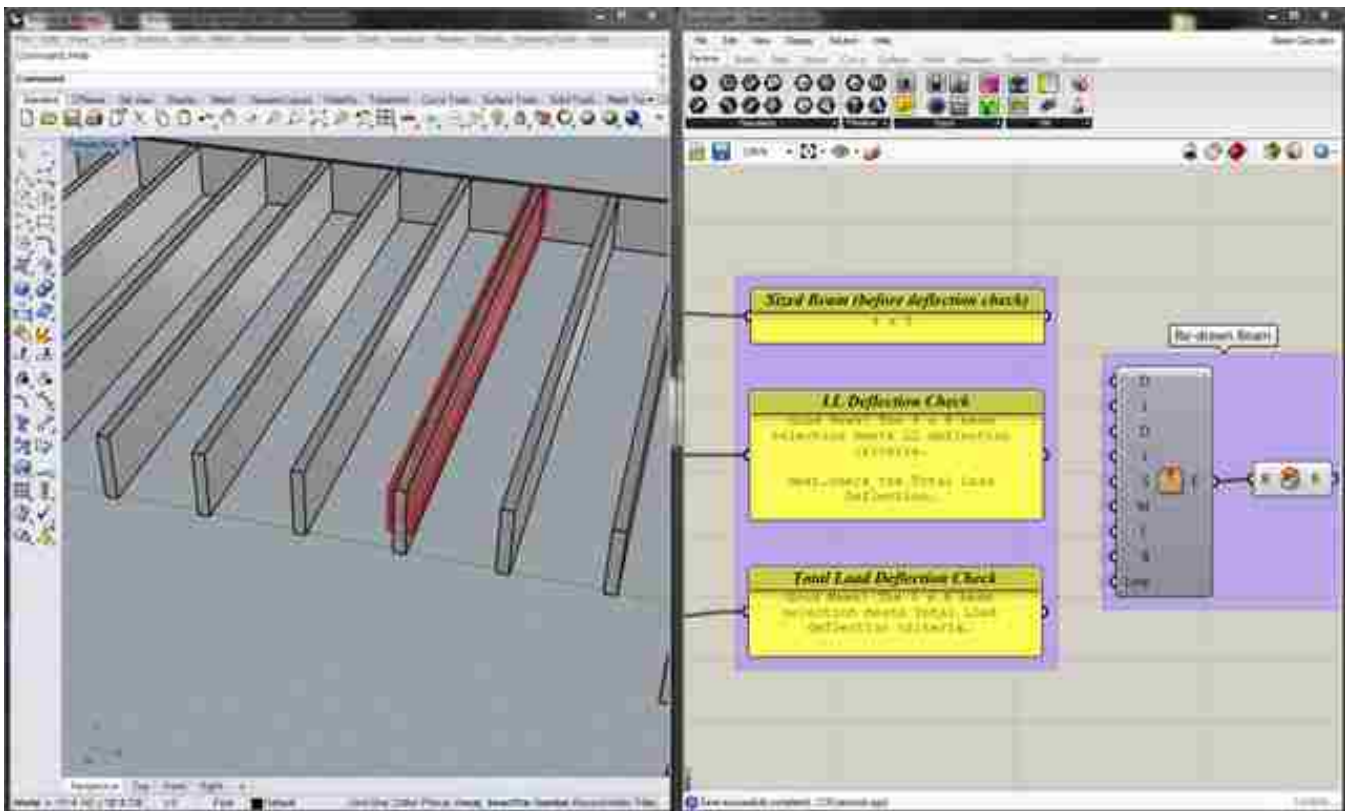
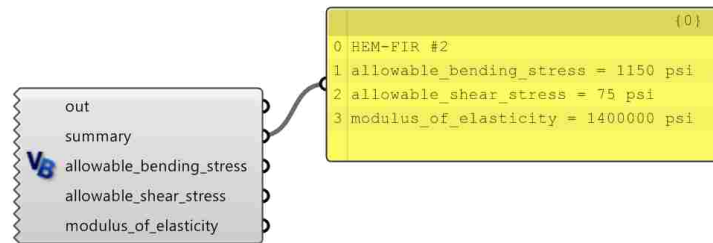
Beam sizing is a necessary process in architectural design. Although there are load and span charts available in design reference books, this calculation could be done during the digital design process using a computational tool like Grasshopper. For basic sizing, the only information required is the beam span, the imposed loads, and the material characteristics of the beam.

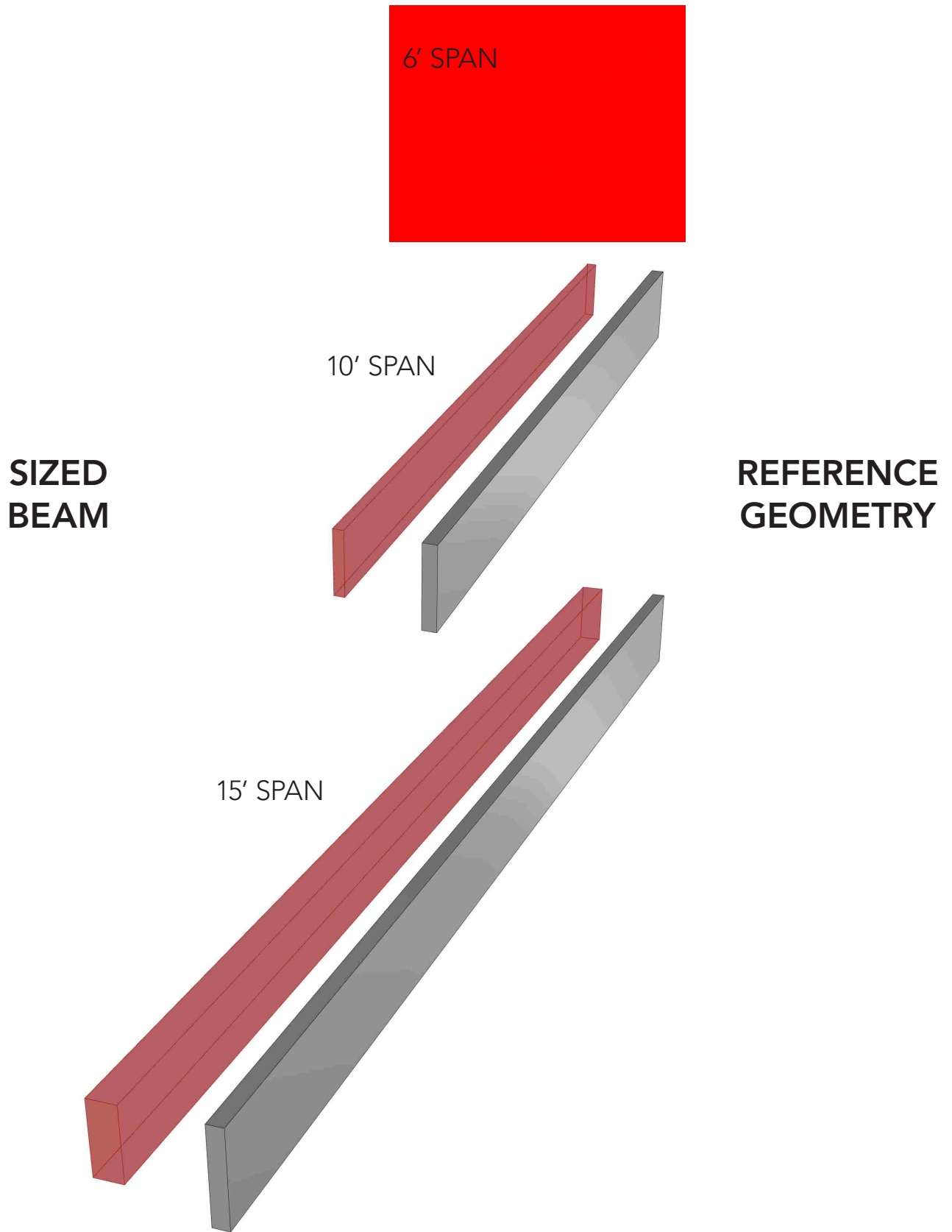
A Grasshopper definition was created that runs through the basic structural calculations for sizing a simply supported wood beam that is carrying a uniformly distributed load. This is by no means a replacement for a structural engineer, however it is a way to aid in the material selection process and more accurately represent floor thickness/beam geometry in three-dimensional models. The definition relies on knowing the beam material's allowable shear stress, allowable bending stress, and modulus of elasticity. Using the Material Logic website, these properties were added to the database for a variety of commonly used wood species, and a custom material component was then downloaded for use in the definition.

The definition is enabled by selecting geometry from Rhino that is a stand in for the beam to be sized. Then, using the geometry and material information supplied from the custom material component, the definition sizes the beam for shear and bending. Once the appropriate cross section has been determined, the sized beam is evaluated for deflection based on criteria chosen by the user (figure 18). If the initial cross-section does not meet the criteria, the next largest section that does is chosen. The final sized beam is re-drawn in the appropriate position. As geometry or load criteria changes the beam size will automatically update providing a more realistic depiction of the actual beam geometry.

**FIGURE 19**

Sized beam versus reference geometry





**FIGURE 18**  
Results of  
deflection check  
using Hem-Fir #2

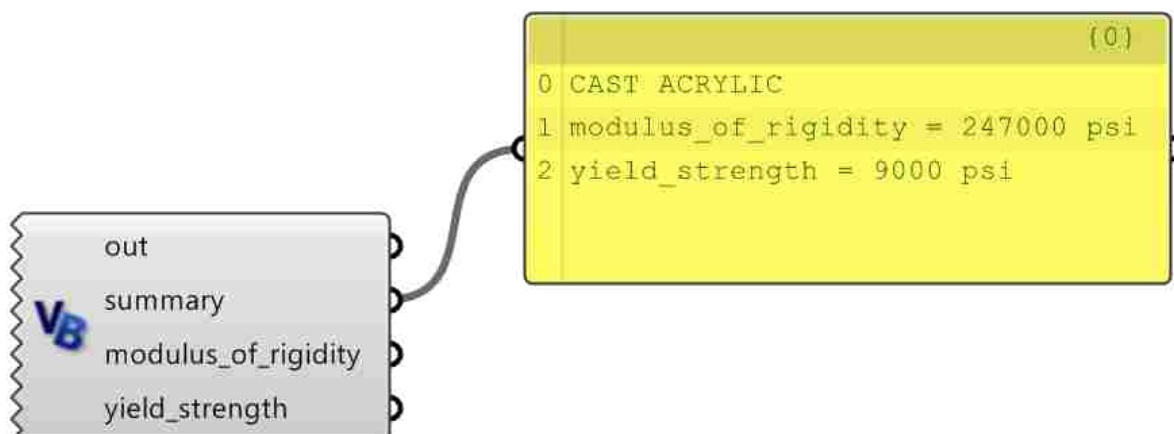
## Live Hinge Calculator

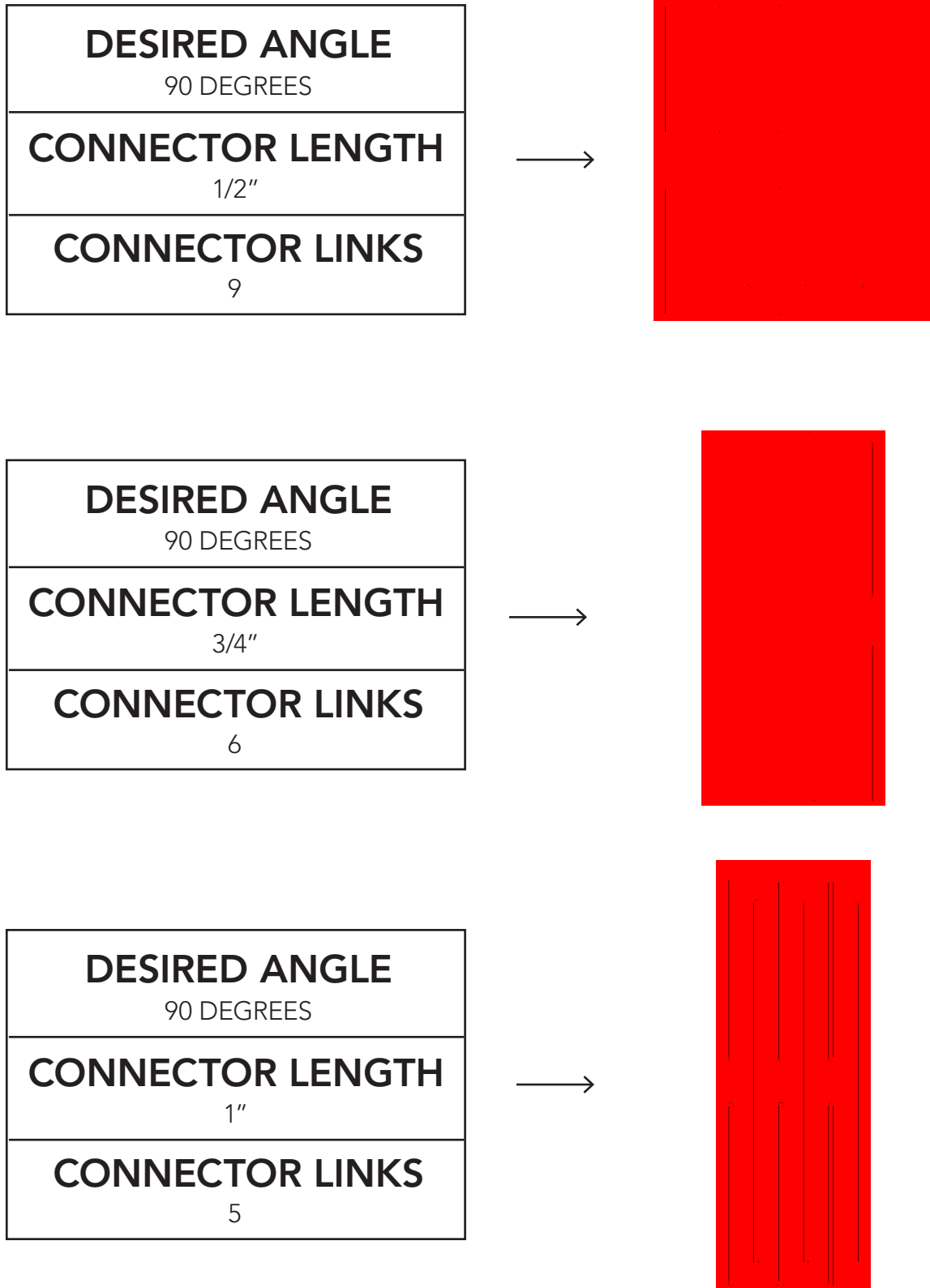
A live hinge is a flexible hinge made of the same material as the two rigid pieces it connects. In sheet goods, a live hinge can be achieved by cutting a series of offset lines. Instead of relying on a material's ability to bend, this method of hinging works primarily through the torsion or twisting that occurs in the slender members (links) between cut lines.

Although this is a relatively simple pattern to draw, without considering the characteristics of the material being used there is no guarantee the hinge will function as intended. Since each link is in torsion, it is possible to calculate the torsional stress in each member to help design a hinge that will not fracture the material. To perform this calculation, you must know a material's modulus of rigidity as well as its yield strength. Using its yield strength as the max allowable stress ensures that only elastic deformation will occur in the links when bending. A Grasshopper definition was created that uses this information to calculate the number of links needed in a live hinge to achieve a specific angle of bend. Using the Material Logic site, both yield strength and modulus of rigidity were download as a custom material component to design a live hinge in cast acrylic.

**FIGURE 21**

Connector length  
versus number of  
connector links



**FIGURE 20**

Custom cast  
acrylic component

In the definition, the geometric constraints for the hinge, specifically the angle of bend, the link length, and the height / width of the link are entered. Then, using the modulus of rigidity and the yield strength, the torsional stress in each link is calculated and the number of links required to achieve the angle of bend is determined. Offset clearance is then calculated to ensure that when in torsion, each link has enough clearance to rotate freely. As the geometric constraints change, the number of links needed increases or decreases. The typical result in this material exploration indicated that as the link length decreased, an increase in the link count was required to achieve the desired angle of bend.

Physical tests were conducted to ensure the accuracy of the definition. Initial tests were conducted using 100% of the material's yield strength. At this percentage, the material failed almost exactly at the specified angle of bend. A safety factor was then added to the definition to guarantee that the desired angle of bend could be achieved repeatedly without the risk of failure. Results from the physical tests were then added to the Material Logic database, as an achievable radius of bend with a specific hinge pattern and sizing. This extends the notion of material information beyond the original uniform sample to include that of a manipulated material. Other examples of material manipulations could include kerf cutting or even the characteristics of a material assembly.

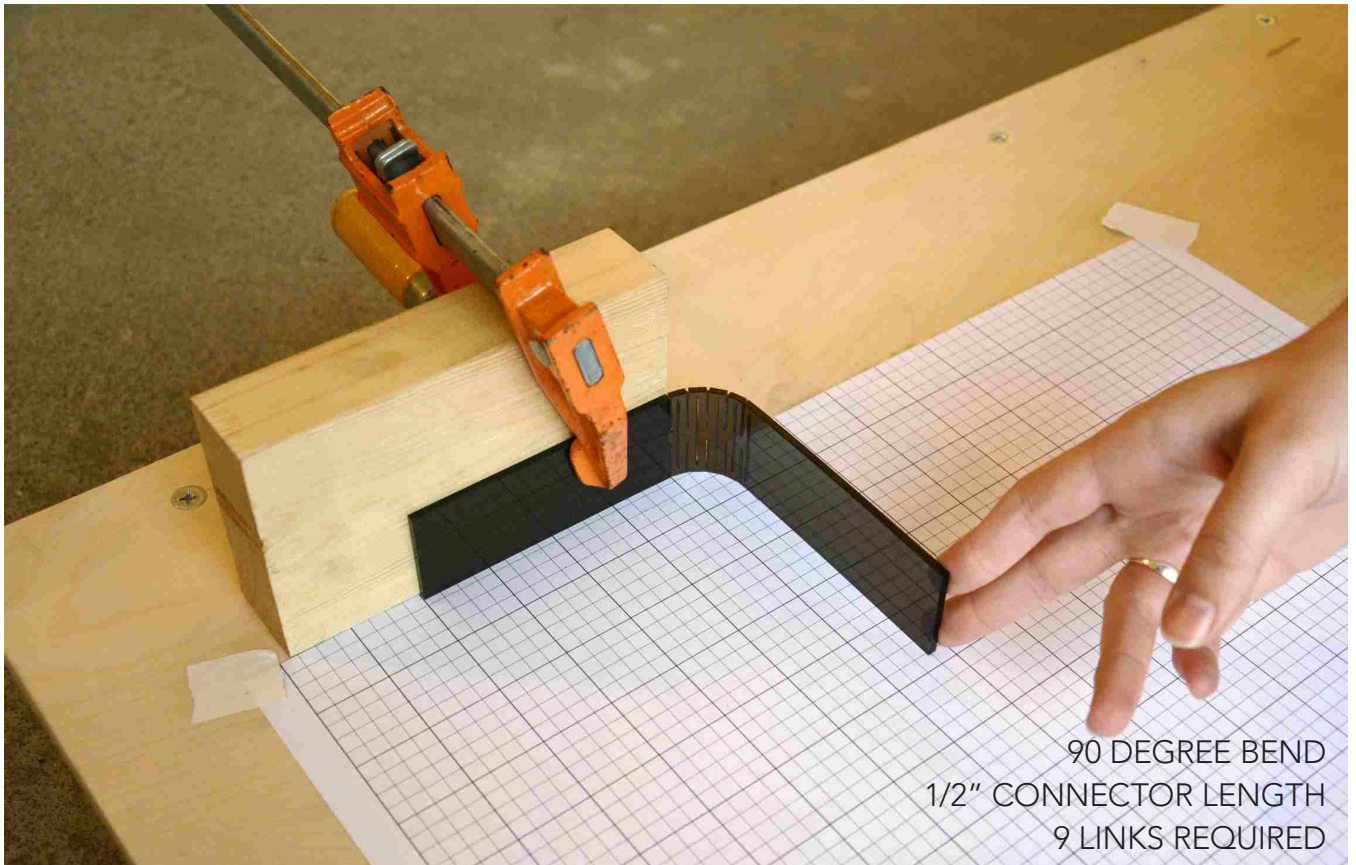
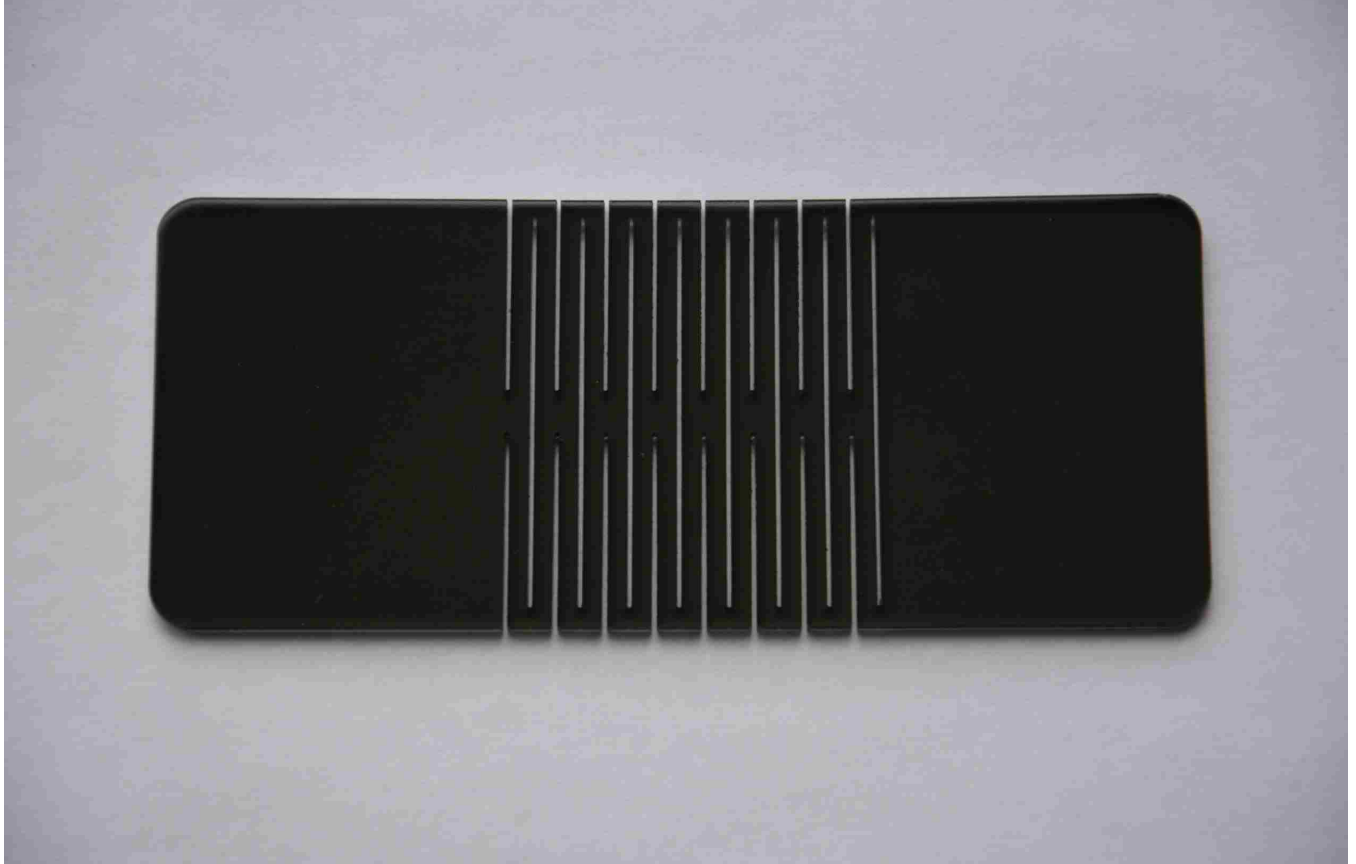
**FIGURE 22**

Live hinge in acrylic

**FIGURE 23**

Physical test at 100% yield strength







## CHAPTER 10 / Conclusion

Although materials are important for the realization of architectural form, they are seldom chosen in conjunction with geometric design. This is due, in part, to the limitations of existing digital tools. This thesis demonstrated one way to expand the potential of digital design tools through the incorporation of material information. Through the development of the PHP script, custom Grasshopper components were created that contain user selected material information. Information packaged in these custom components can be selected and assembled on the Material Logic website, providing users with drag and drop access to shared material information for the digital design environment.

Additionally, the Material Logic website acts as a platform for sharing knowledge about the application of materials. The forum provides a venue for the discussion of how users have integrated custom material components into their parametric definitions and design process. By leveraging the knowledge and experience of many users, the information contained in the forum can be a powerful resource of material applications.

While this thesis has addressed a solution that is specific to Rhinoceros and Grasshopper, several of the other design tools used by architects currently lack the ability to integrate material information. Future work could expand the PHP script to deliver information from the Material Logic database in formats acceptable for other digital design tools.



## BIBLIOGRAPHY

- Booth, P. (2009). Digital materiality: Emergent computational fabrication. *Performative Ecologies in the Built Environment: Sustainability Research Across Disciplines: 43rd Annual Conference of the Australian and New Zealand Architectural Science Association*, Retrieved from <http://cumincad.scix.net/cgi-bin/works/Show?4f1b>
- Corser, R. (2011). Stress-crafting - interweaving digital dexterity and manual intelligence. Where do you stand: *Proceedings from the 99th ACSA Annual Meeting* (pp.438-445). Washington D.C.: ASCA Press.
- Demaine, E. D., Demaine, M. L., Koschitz, D., & Tachi, T. (2011). Curved crease folding: a review on art, design and mathematics. *Proceedings of the IABSE-IASS Symposium: Taller, Longer, Lighter*: London, England.
- Fleischmann, M., Lienhard, J., & Menges, A. (2007). Computational design synthesis: Embedding material behavior in generative computational processes. *Respecting Fragile Places: 29th eCAADe Conference Proceedings*, 759-767. Retrieved from [http://cumincad.scix.net/cgi-bin/works/Show?ecaade2011\\_013](http://cumincad.scix.net/cgi-bin/works/Show?ecaade2011_013)
- Kalay, Y. E. (2004). *Architecture's new media: principles, theories, and methods of computer-aided design*. Cambridge, MA: MIT Press.
- Kolarevic, B. (Ed.) (2005). *Architecture in the digital age: design and manufacturing*. Abingdon, Oxford, UK: Taylor & Francis.
- Kolarevic, B. & K. Klinger (Eds.) (2008). *Manufacturing material effects: Rethinking design and making in architecture*. New York, NY: Routledge.
- Leach, N. (2009). Digital morphogenesis. *Architectural Design*, 79(1), 32-27.
- Marble, S. (2010). Imagining risk. In P. Deamer & P. G. Bernstein (Eds.), *Building (in) the future: Recasting labor in architecture* (pp. 38-43). New York, NY: Princeton Architectural Press.

- Material ConneXion. (n.d.). *Material ConneXion*. Retrieved June 2013, from <http://www.materialconnexion.com/>
- McCullough, M. (1996). *Abstract craft: the practiced digital hand*. Cambridge, MA: The MIT Press.
- Menges, A. (2010). Material information: Integrating material characteristics and behavior in computational design for performative wood construction. *ACADIA 10: LIFE in:formation, On Responsive Information and Variations in Architecture: Proceedings of the 30th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA)*, 151-158. Retrieved from [http://cumincad.scix.net/cgi-bin/works/Show?acadia10\\_151](http://cumincad.scix.net/cgi-bin/works/Show?acadia10_151)
- MySQL Cluster 7.3 GA. (n.d.). *MySQL Cluster 7.3 GA*. Retrieved April 2013, from <http://www.mysql.com/>
- Online Materials Information Resource - MatWeb. (n.d.). *Online Materials Information Resource - MatWeb*. Retrieved May 2013, from <http://www.matweb.com/>
- Oxman, N. (2007). Digital craft: Fabrication-based design in the age of digital production. *Workshop Proceedings for Ubicomp 2007: International Conference on Ubiquitous Computing*, 534-538. Retrieved from [ambient.media.mit.edu/transitive/ubicomp07papers/oxman.pdf](http://ambient.media.mit.edu/transitive/ubicomp07papers/oxman.pdf)
- PHP: Hypertext Preprocessor. (n.d.). *PHP: Hypertext Preprocessor*. Retrieved April 2012, from <http://www.php.net/>
- Pye, D. (1968). *The nature and art of workmanship*. Cambridge, MA: Cambridge University Press.
- Reiser, J., & Umemoto, N. (2006). *Atlas of novel tectonics*. New York, New York: Princeton Architectural Press.

- Stanton, C. (2010). Material feedback in digital design tools. *Proceeding of the 15th International Conference on Computer-Aided Design Research in Asia/ Hong Kong*, 7-10 April 2010, 555-564. Retrieved from [http://cumincad.scix.net/cgi-bin/works/Show?caadria2010\\_051](http://cumincad.scix.net/cgi-bin/works/Show?caadria2010_051)
- Thomas, K. L. (Ed.) (2007). *Material matters: architecture and material practice*. Abingdon, Oxford, UK: Routledge.



## IMAGE CREDITS

All images not listed below were produced by the author.

**Figure 04** *The 'Albers Effect'* [Photograph]. (2011). Retrieved May 30, 2013. from: <http://bryantyeewordpress.com/tag/paper-engineering-concentric-circles-curved-folding/> (page 09).

**Figure 05** *Nola Chair* [Photograph]. (2008). Retrieved June 01, 2013. from: <http://www.onegoodchair.com/competition/2008/winners/> (page 14).

**Figure 06** *Nola Chair* [Graphic]. (2008). Retrieved June 01, 2013. from: <http://www.onegoodchair.com/competition/2008/winners/> (page 14).

**Figure 07** *Material ConneXion Homepage* [webpage]. (n.d.) Retrieved June 01, 2013. from: <http://www.materialconnexion.com/> (page16).

**Figure 08** *MatWeb Homepage* [webpage]. (n.d.) Retrieved May 01, 2013. from: <http://www.matweb.com/> (page16).

## APPENDIX A

```

<?php session_start();

mysql_connect("host", "username", "password") or die("can't connect");
mysql_select_db("material_logic") or die ("can't select database");

// ----- DECLARE GLOBAL VARIABLES -----

if (isset($_POST['submit']));
    $props = $_POST['prop'];
    $count = count($props);
    $material = $_POST['material'];
    $outputid = "8ec86459-bf01-4409-baee-174d0d2b13d0";

    $guid = array ("9beb07fe-ad43-40a6-ab45-b591cd82d65d",
        "920d54b8-0bfd-4d51-9761-6faf4a7c52ed",
        "8253a6a8-b8e5-470d-b731-4adb8a11cac4",
        "fcc3a431-2dc4-4901-a7f6-efca926a2b58",
        "77f35305-5b33-4a15-8a8d-f190569c7ac6",
        "860fbc65-d54a-4f1f-9fd3-a12a72431c61",
        "c2ab7366-0c9f-48bd-a166-bfbbb25e0360",
        "86a431d8-9d8d-4967-a9c5-3cb28e665c66",
        "e86f8968-3b2f-406f-87c2-fc6a9cb1f795",
        "83b729a3-d30b-49ed-a71a-87afd767a44e",
        "87098d74-301f-4503-9fc1-2fdd20860676",
        "e4dad78f-e85d-426c-b15a-8dda3cf12e39");

    $item_string = "";

    for ($m=0; $m<$count; $m++){
        $xpld_props = explode(',', $props[$m]);
        $out_keys = $xpld_props[0];

        $item_string .= $out_keys. "," . $out_keys. "," . "Output
parameter" . $out_keys. "," . $guid[$m]. "," . "false". "," . "0".
"|";
    }

    $item_array = array_filter(explode("|", $item_string));

// ----- XML LOAD -----

$file = "../gh/empty.ghx";

if (file_exists($file))
    {
    $xml = simplexml_load_file($file);
    }
else
    {
    exit('file does not exist');
    }

// ----- EDIT XML -----

if((isset($props)) && (!empty($props)))
    {

```

```

// ----- EDIT COMPONENT NAME -----

if ((isset($material)) && (!empty($material)))
    {
    $xml->chunks[0]->chunk->chunks->chunk[4]->chunks->chunk->chunks->chunk-
>items->item[2] = $material;
    }
else
    {
    echo "error";
    }

// ----- EDIT VB SCRIPT INFO -----
$string = "";
$sum_string = "A.add(\"" . $material."\" )\n";

$out_dim = "Dim A As New List (Of String)" . "\n";
$out_assign = "summary = A";

for ($m=0; $m<$count; $m++){
    $xpld_props = explode(' ', $props[$m]);
    $out_keys = str_replace(' ', '_', $xpld_props[0]);
    $out_vals = $xpld_props[1];
    $out_units = $xpld_props[2];

    $string .= $out_keys. "=" . $out_vals. "\n";
    $sum_string .= "A.add(\"" . $out_keys. " = " . $out_vals. "
    ".$out_units."\" )\n";
}

$xml->chunks[0]->chunk->chunks->chunk[4]->chunks->chunk->chunks-
>chunk->items->item[0]= $string. "\n" . $out_dim. "\n" . $sum_
string. "\n". $out_assign. "\n";

// ----- EDIT PARAMETER DATA -----

$xml->chunks[0]->chunk->chunks->chunk[4]->chunks->chunk->chunks->chunk-
>chunks->chunk[1]->items->item[1] = ($count + 2);

for($i=0; $i<$count; $i++){
    $xml->chunks[0]->chunk->chunks->chunk[4]->chunks->chunk->chunks->chunk-
>chunks->chunk[1]->items->addChild('item', $outputid);
    $out_attname=array("name", "index", "type_name", "type_code");
    $out_attval=array("OutputId", ($i+2), "gh_guid", "9");

    for ($j=0;$j<count($out_attname);$j++)
        {
        $xml->chunks[0]->chunk->chunks->chunk[4]->chunks->chunk->chunks-
>chunk->chunks->chunk[1]->items->item[($i+4)]->addAttribute($out_
attname[$j], $out_attval[$j]);
        }
}

// ----- ADD OUTPUT PARAMATER CHUNKS -----

for ($x=0; $x<($count); $x++)
    {

```

```

$xml->chunks[0]->chunk->chunks->chunk[4]->chunks->chunk->chunks->chunk->
>chunks->chunk[1]->chunks->addChild('chunk');

$chunk_attname=array("name", "index");
$chunk_attval=array("OutputParam", ($x+2));

for ($y=0; $y<count($chunk_attname); $y++)
{
    $xml->chunks[0]->chunk->chunks->chunk[4]->chunks->chunk->
    >chunks->chunk->chunks->chunk[1]->chunks->chunk[$x+2]->
    >addAttribute($chunk_attname[$y],$chunk_attval[$y]);
}

$xml->chunks[0]->chunk->chunks->chunk[4]->chunks->chunk->chunks->chunk->
>chunks->chunk[1]->chunks->chunk[$x+2]->addChild('items');

$xml->chunks[0]->chunk->chunks->chunk[4]->chunks->chunk->chunks->chunk->
>chunks->chunk[1]->chunks->chunk[$x+2]->items->addAttribute('count',
'6');

$item_array2 = str_replace(' ','_',(explode(",", $item_array[$x])));

for ($y=0; $y<6; $y++)
{
    $xml->chunks[0]->chunk->chunks->chunk[4]->chunks->chunk->chunks->
    >chunk->chunks->chunk[1]->chunks->chunk[$x+2]->items->
    >addChild('item', $item_array2[$y]);

    $item_attname=array("name", "gh_string", "type_code");

    $item_attval=array(array("Name", "NickName", "Description",
    "InstanceGuid", "Optional", "SourceCount"), array("gh_string",
    "gh_string", "gh_string", "gh_guid", "gh_bool", "gh_int32"),
    array("10", "10", "10", "9", "1", "3"));

    for($z=0;$z<count($item_attname);$z++)
    {
        $xml->chunks[0]->chunk->chunks->chunk[4]->chunks->
        >chunk->chunks->chunk->chunks->chunk[1]->chunks->
        >chunk[$x+2]->items->item[$y]->addAttribute($item_
        attname[$z],$item_attval[$z][$y]);
    }
}

$location = array("X","Y");
$val = 10;
$coordinates = array ("X","Y","W","H");

$chnks_data = array("", "");
$chnks_attname = array("name","type_name", "type_code");
$chnks_attval=array(array("Pivot","Bounds"), array("gh_drawing_pointf",
"gh_drawing_rectanglef"), array("31","35"));

$xml->chunks[0]->chunk->chunks->chunk[4]->chunks->chunk->chunks->chunk->
>chunks->chunk[1]->chunks->chunk[$x+2]->addChild('chunks');

```

```

$xml->chunks[0]->chunk->chunks->chunk[4]->chunks->chunk->chunks->chunk-
>chunks->chunk[1]->chunks->chunk[$x+2]->chunks->addAttribute('count',
'1');

$xml->chunks[0]->chunk->chunks->chunk[4]->chunks->chunk->chunks->chunk-
>chunks->chunk[1]->chunks->chunk[$x+2]->chunks->addChild('chunk');

$xml->chunks[0]->chunk->chunks->chunk[4]->chunks->chunk->chunks-
>chunk->chunks->chunk[1]->chunks->chunk[$x+2]->chunks->chunk-
>addAttribute('name', 'Attributes');

$xml->chunks[0]->chunk->chunks->chunk[4]->chunks->chunk->chunks->chunk-
>chunks->chunk[1]->chunks->chunk[$x+2]->chunks->chunk-
>addChild('items');

$xml->chunks[0]->chunk->chunks->chunk[4]->chunks->chunk->chunks->chunk-
>chunks->chunk[1]->chunks->chunk[$x+2]->chunks->chunk->items-
>addAttribute('count', '3');

for ($a=0; $a<2; $a++)
{
$xml->chunks[0]->chunk->chunks->chunk[4]->chunks->chunk->chunks-
>chunk->chunks->chunk[1]->chunks->chunk[$x+2]->chunks->chunk->items-
>addChild('item', $chnks_data[$a]);

    for ($b=0; $b<count($chnks_attname); $b++)
    {
        $xml->chunks[0]->chunk->chunks->chunk[4]->chunks->chunk-
        >chunks->chunk->chunks->chunk[1]->chunks->chunk[$x+2]-
        >chunks->chunk->items->item[$a]->addAttribute($chnks_
        attname[$b], $chnks_attval[$b][$a]);
    }
}
for ($b=0; $b<count($location); $b++)
{
$xml->chunks[0]->chunk->chunks->chunk[4]->chunks->chunk-
>chunks->chunk->chunks->chunk[1]->chunks->chunk[$x+2]-
>chunks->chunk->items->item[0]->addChild($location[$b],
$val);
}
for ($c=0; $c<count($coordinates); $c++)
{
$xml->chunks[0]->chunk->chunks->chunk[4]->chunks-
>chunk->chunks->chunk->chunks->chunk[1]->chunks-
>chunk[$x+2]->chunks->chunk->items->item[1]-
>addChild($coordinates[$c], $val);
}
}

// ----- ASSEMBLE NEW GHX -----

$xml->asXML('../gh/material_logic.ghx');
$query = mysql_query("SELECT * FROM mat_id WHERE material = '$material'");
$row = mysql_fetch_array($query);
$mat_id = $row['mat_id'];

header("Location: ../material.php?id=$mat_id&download=1");

```

```
    }
    else{
        $query = mysql_query("SELECT * FROM mat_id WHERE material =
'$material'");
        $row = mysql_fetch_array($query);
        $mat_id = $row['mat_id'];

        header("Location: ../material.php?id=$mat_id&error=1");
        die();
    }
?>
```