# gerilla v1.0

grasshopper to energy plus

credits

Michael Marvin
Drew Orvieto
Ben Silverman

Product Architecture
Stevens Institute of Technology

## 1. What is Gerilla?

More than likely you already know the answer to this question otherwise you wouldn't be reading this primer, but Gerilla is a plugin for Grasshopper / Rhino that will allow you to run full building energy simulations on using the EnergyPlus simulation engine. It was born in the Product Architecture Lab at Stevens Institute of Technology in Fall of 2011 and is the result of the collaborative efforts of Michael Marvin, Drew Orvieto, and Ben Silverman. Gerilla has been developed as an open source, free to download plugin for Grasshopper (all you have to do is make sure to give us all the credit when you design your next amazing zero energy building).

Currently, Gerilla is still in it's infancy, and as such we haven't published the code and plugin to the internet yet, but if you are interested in wrestling the Gerilla feel free to contact us at:

**gerilla-for-grasshopper@gmail.com**

Once things are developed a little further it is our intention to publish the plugin, install files, and libraries out to the net for all to download and enjoy.

## 2. System Requirements and Installing Gerilla

If you're at all familiar with energy modeling you'll know that hardware power is a major part of being able to run simulations on complex models. Currently, Gerilla is somewhat limited in terms of what you'll be able to do with it with regard to model complexity so you probably won't run into any problems running Gerilla on just about any machine (yet), but you obviously will need to have enough computer power to run Rhino and EnergyPlus.

You'll need to download **EnergyPlus V6.0** and install it. You can download from[1]:

*http://apps1.eere.energy.gov/buildings/energyplus/*

Gerilla is known to be compatible with **Grasshopper V0.8.0061** and hasn't been tested on any other versions yet. Grasshopper can be downloaded here:

*http://www.grasshopper3d.com/page/next-build*

After you've emailed us and we've sent you the Gerilla.zip file, open it up and copy the Gerilla.gha to your Grasshopper plugins folder here[2]:

*C:\Program Files (x86)\Rhinoceros 4.0\Plug-ins\Grasshopper\Components*

[1] Gerilla was written for EnergyPlus Version 6.0 and hasn't yet been tested on Version 7.0 yet. Check back on our page regularly for new information regarding compatibility with the new versions of EPlus.

[2] This primer was written assuming that you're using Windows 7. If you are on an earlier OS your Grasshopper folder is probably located elsewhere. If you aren't sure where it is check out the help files on www.grasshopper3d.com

*(System Requirements and Installing Gerilla continued…)*

After that's set, copy the folder labeled Gerilla to your C: drive to install all of the Materials, Assemblies, and Batch Files that you'll need to assemble your models and run simulations:

*C:\Gerilla*

## 3. Getting Started Modeling

Since Gerilla is a plugin to allow Rhino / Grasshopper geometry to be sent to EnergyPlus any models produced for Gerilla will need to be built based upon EnergyPlus's modeling rules. We would recommend familiarizing yourself with the basic ins and outs of EnergyPlus if you are going to attempt to really use the simulator. The Department of Energy packed a huge amount of really amazing, super robust tools into EPlus that Gerilla doesn't yet have the ability to access, so understanding how deep this program is will give you a much better idea of what the potential of Gerilla could be, especially if you would like to work with us to develop it further.

One of the most basic rules of geometry creation in Energy Plus is that all spaces must be 100% water-tight closed volumes in order to be able to be transferred to the simulator. This can be accomplished by creating closed boxes in Rhino or by making closed Breps in Grasshopper via whatever modeling methods are the most comfortable to you. This, clearly, seems like a simple concept, but like many things can become a difficult task as you models become more complex. Model carefully.

Here's a brief overview of some of the rules of modeling in Gerilla. For a much more detailed description of all of this please see the EnergyPlus website. Pretty much all of the rules of modeling described there apply to your Gerilla models, and the Department of Energy has excellent documents that delve into much more detail than we're going to here. Since this is the Gerilla Primer, we'll try to stay more in the Grasshopper world and leave the simulator to the experts.

The core of what makes EnergyPlus work is it's IDF file. The IDF is a text-type file that contains all of the information necessary for EnergyPlus to assemble your model and run a simulation on it. Seems easy enough, but the assembly of that file can become extremely complex. Gerilla takes your model and extracts all of the data that the IDF needs – all of the vertices, constructions, assemblies, etc. and then combines all of that data, writes an IDF and sends it to EnergyPlus with the click of one node in Grasshopper. Gerilla can then take the results of your simulation and read the information back into Grasshopper for review and visualization.

Vertices: As is stated above, EnergyPlus requires watertight geometry. This geometry is defined in the IDF file (see below), by inputing the vertices of a Brep's component planar surfaces in a consistent order (Gerilla always defaults to a clockwise input order when describing surfaces). You'll notice that we said **planar** surfaces – EnergyPlus will not accept any curved surfaces, so if you have a curve you will have to break it down to simpler facets in order to bring it into EPlus.[3]

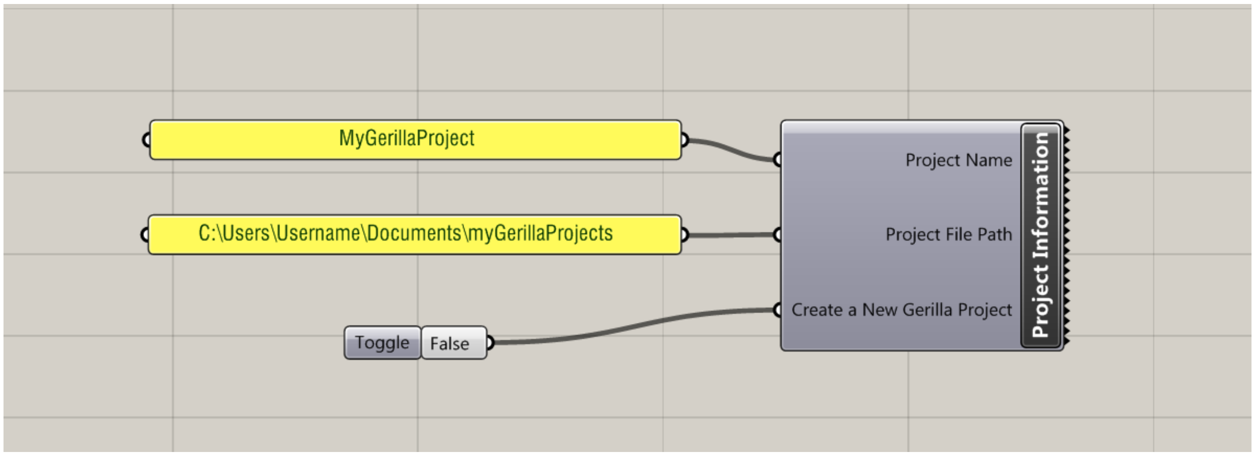
Zones: Currently Gerilla will accept multi-zone input, however each zone must have a precisely matching plane in order for this function to work properly. Each zone should be defined as a separate Brep in Grasshopper, and any coplanar walls must overlap each other and share identical vertices.

[3] We've tested meshing more complex forms for input into EPlus using Gerilla with varying degrees of success. Ultimately, manually simplifying your geometry will probably give you the cleanest results, as well as reduce your simulation time considerably.

## 4. Component Glossary

The Gerilla tab that now lives in your Grasshopper ribbon consists of three panels: Materials, Simulation, and Zone. The number of panels included in Gerilla and their content will no doubt change as updates are made and the tool grows in future releases, but these are the panels that allow you to use Gerilla's basic functions.

### Project Information

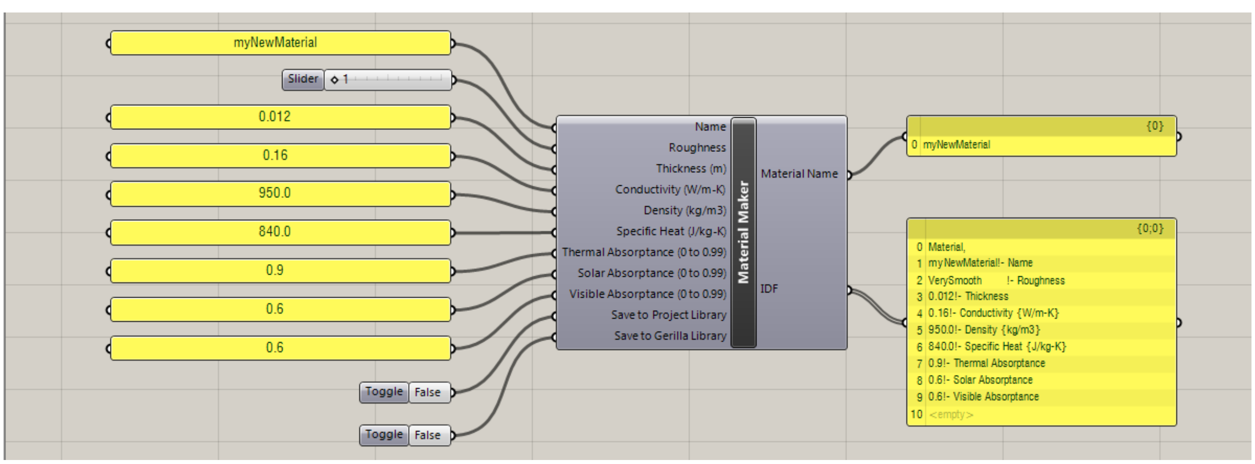



**Project Name:** Input the project name as a string.

**Project File Path:** Input the project file path as a string, or using the Grasshopper "Path" component.

**Create New Gerilla Project:** Input True or False using the Grasshopper "Boolean Toggle" to create a New Gerilla Project.

### Material Maker




**Name:** Input the material name as a string.

**Roughness:** Input the material roughness.

**Thickness:** Input the material thickness (m).

**Conductivity:** Input the material conductivity (W/m-K).

**Density:** Input the material density (kg/m3).

**Specific Heat:** Input the material specific heat (J/kg-K).

**Thermal Absorptance:** Input the material thermal absorptance (0-0.99).

**Solar Absorptance:** Input the material solar absorptance (0-0.99).

**Visible Absorptance:** Input the material visible absorptance (0-0.99).

**Save to Project Database:** Input True or False using the Grasshopper “Boolean Toggle” to save the material to the Project Library.

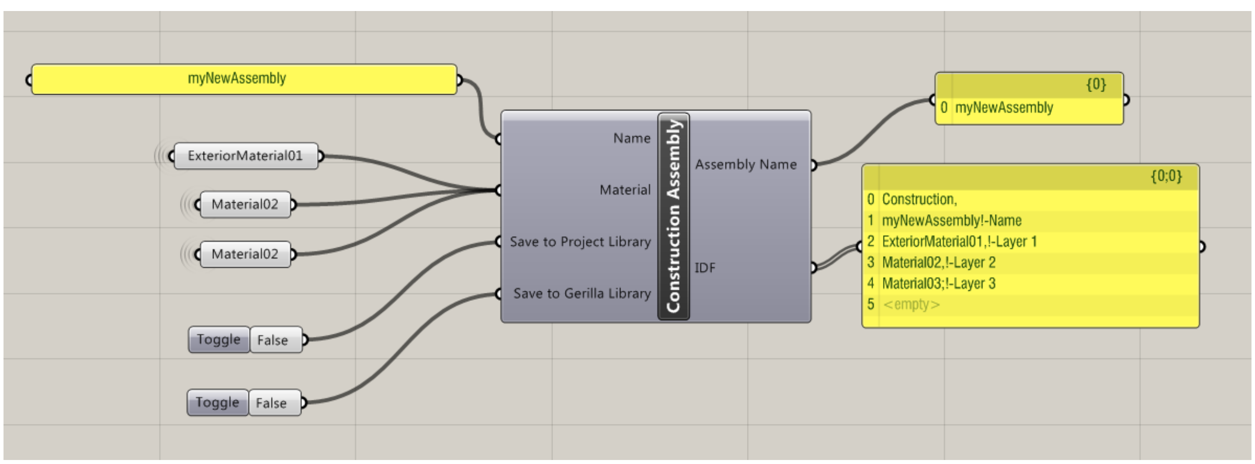
**Save to Gerilla Database:** Input True or False using the Grasshopper “Boolean Toggle” to save the material to the Gerilla Library.

**Outputs:**

**Material Name:** Name of material

**Material:** Material description for IDF

## Construction Assembly




**Name:** Input the construction assembly name as a string.

**Material:** Input the material as a string in the order of the assembly buildup from exterior to interior. This can come from the “Material Library” component, or a series of “Material Maker” components.

**Save to Project Library:** Input True or False using the Grasshopper "Boolean Toggle" to save the assembly to the Project Library.

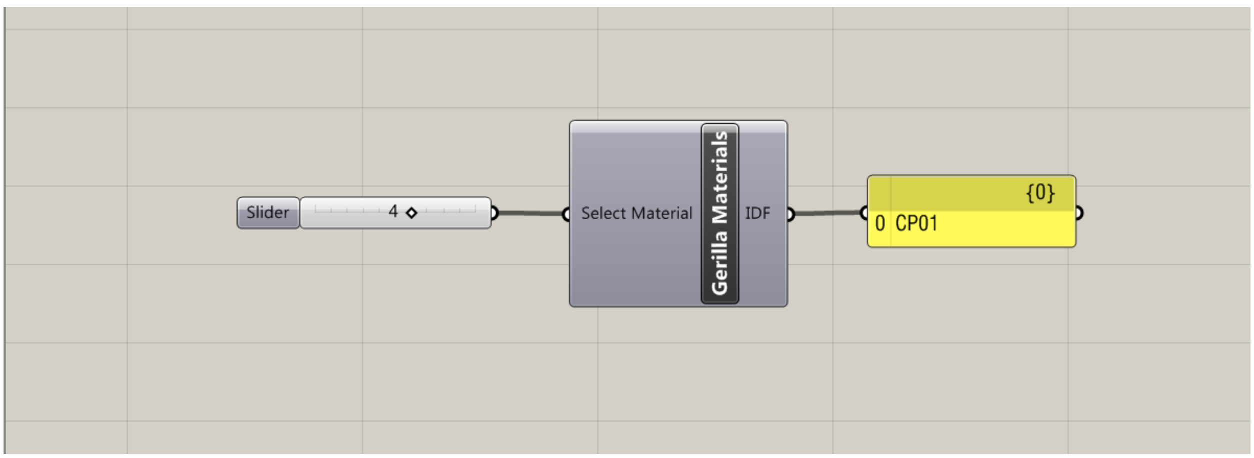
**Save to Gerilla Library:** Input True or False using the Grasshopper "Boolean Toggle" to save the assembly to the Gerilla Library.

**Outputs:**

**Assembly Name:** Name of assembly

**Assembly:** Assembly description for IDF

## Gerilla Materials

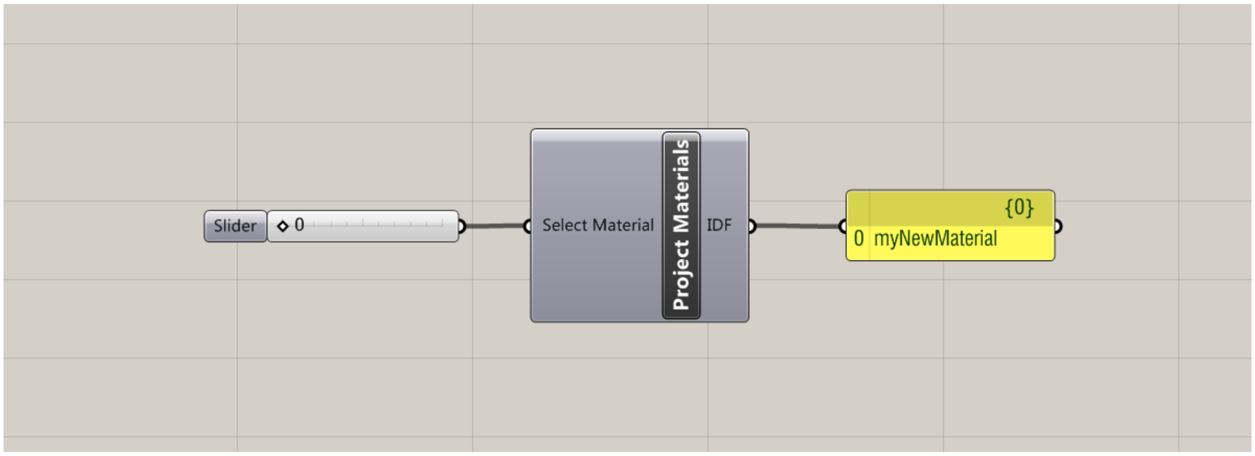



**Select Library:** Input index numbers of Gerilla Library material library as integers in the order of the assembly buildup from exterior to interior.

**Outputs:**

**IDF:** List of selected Gerilla Library materials in the order of the assembly buildup from exterior to interior

**Project Materials**

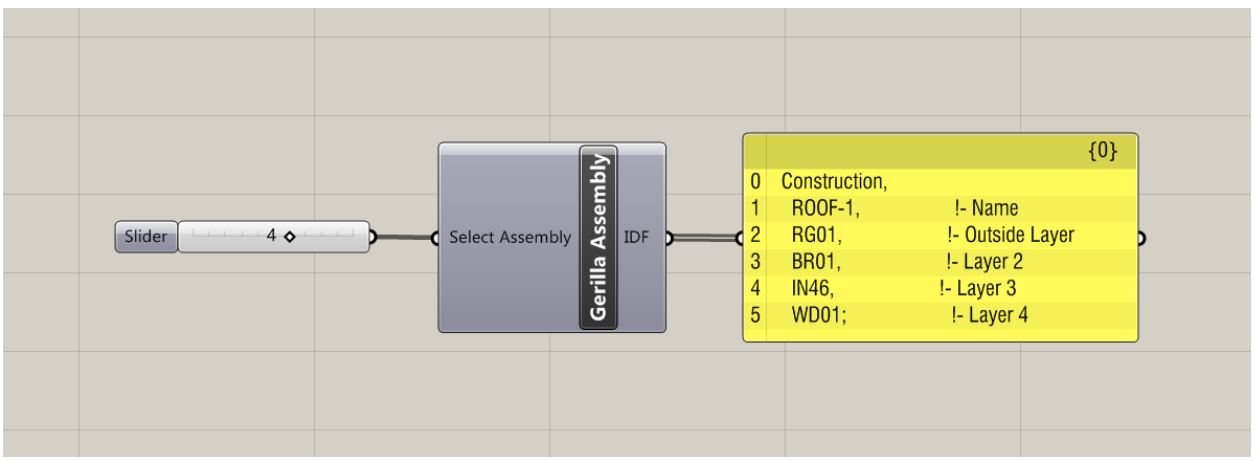



**Select Library:** Input index numbers of Project Library materials as integers in the order of the assembly buildup from exterior to interior for IDF.

**Outputs:**

**IDF:** List of selected Project Library materials in the order of the assembly buildup from exterior to interior for IDF.

**Project Assemblies**



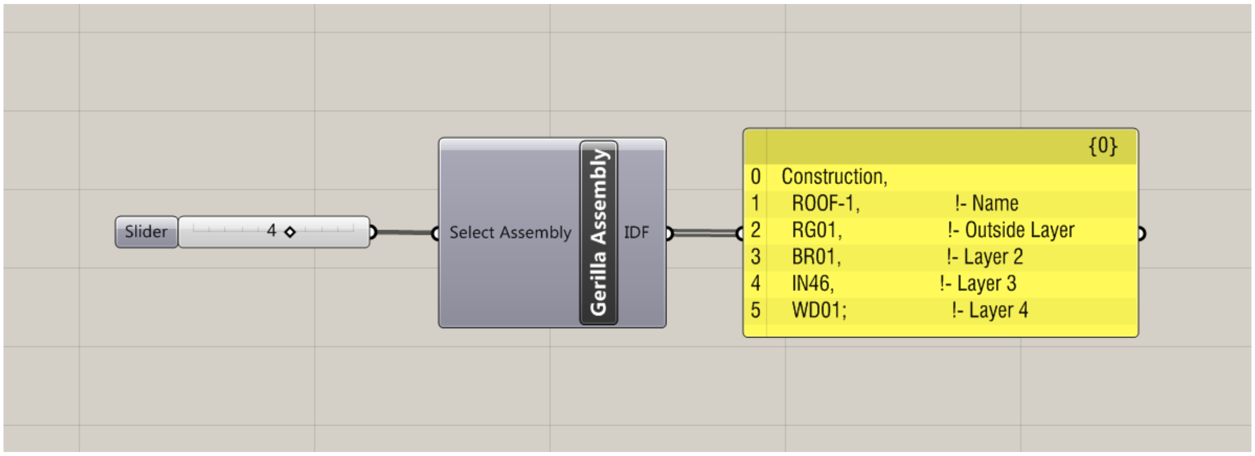

**Inputs:**

**Select Assembly:** Input index numbers of Project Assembly Library assemblies as integers.

**Outputs:**

**Assembly:** List of selected Project Assembly Library assemblies for IDF.

## Gerilla Assemblies

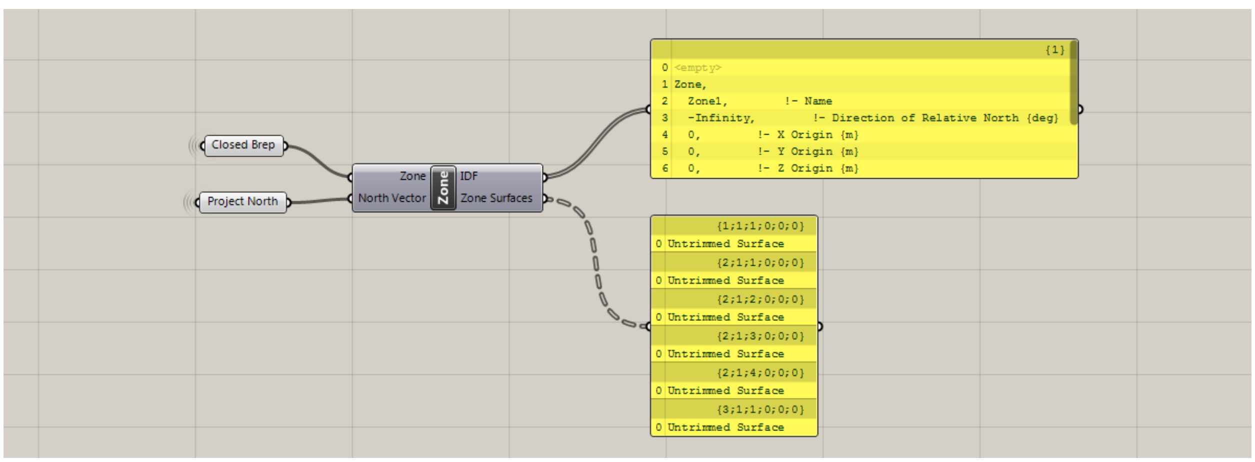



**Select Assembly:** Input index numbers of Gerilla Assembly Library assemblies as integers.

**Outputs:**

**Assembly:** List of selected Gerilla Assembly Library assemblies for IDF.

## Zone




**Inputs:**

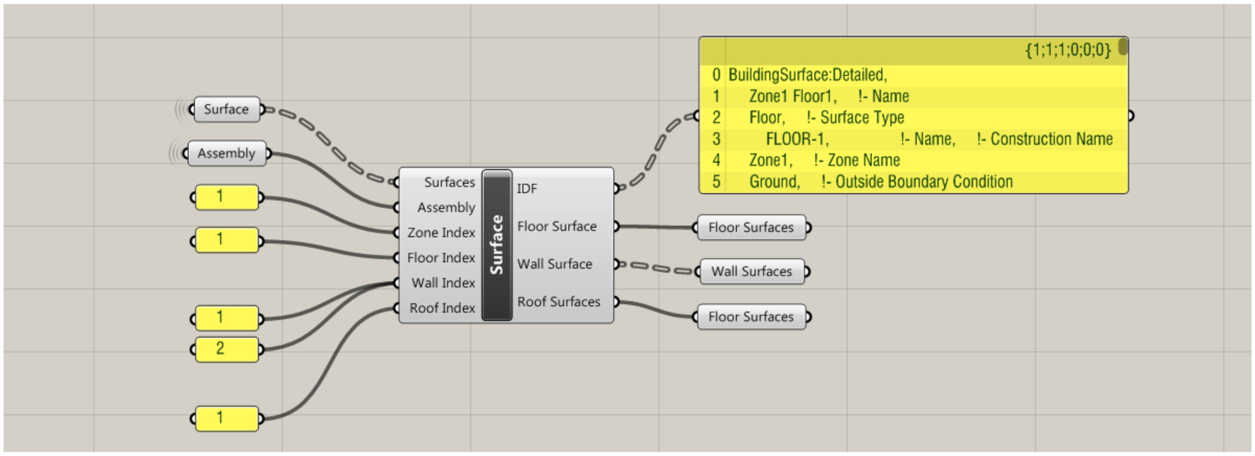
**Zone:** Input Zones to include in simulation as closed Breps.

**North Vector:** Input North Vector as a vector using the Grasshopper “Vector” component.

**Outputs:**

**Surfaces:** Surfaces for “Surfaces” component to assign construction assemblies.

**Surface**



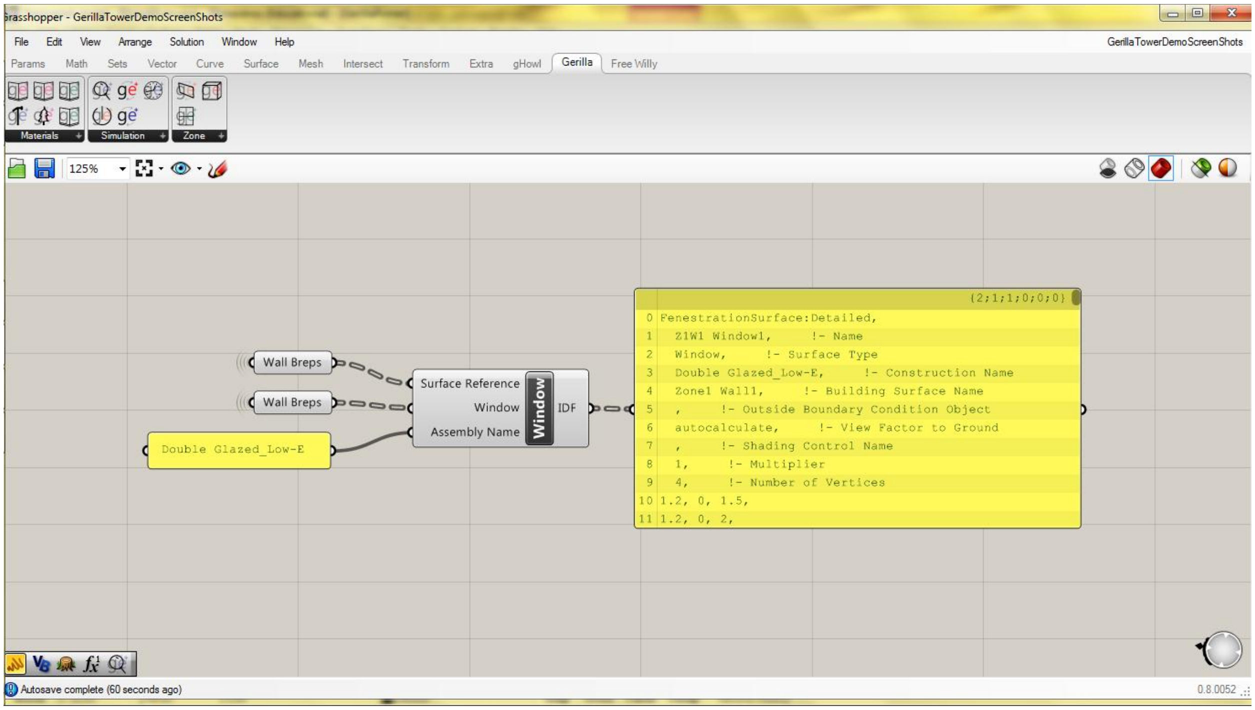

**Inputs:**

| | |
|---|---|
| **Surfaces:** | Input surfaces from “Zone” component. |
| **Construction:** | Input Construction Name using the “Assembly” Component. |
| **Zone Index:** | Input integers for indices of zones in which assemblies will be applied. |
| **Floor Index:** | Input integers for indices of floors in which assemblies will be applied. |
| **Wall Index:** | Input integers for indices of walls in which assemblies will be applied. |
| **Roof Index:** | Input integers for indices of roofs in which assemblies will be applied. |

**Outputs:**

| | |
|---|---|
| **IDF:** | Surface information for IDF. |
| **Floor Surfaces:** | All indexed floor surfaces. |
| **Wall Surfaces:** | All indexed wall surfaces. |
| **Roof Surfaces:** | All indexed roof surfaces. |

## Window




**Inputs:**

**Window Surface Reference:** Input Surface that Contains the Window as a Brep.

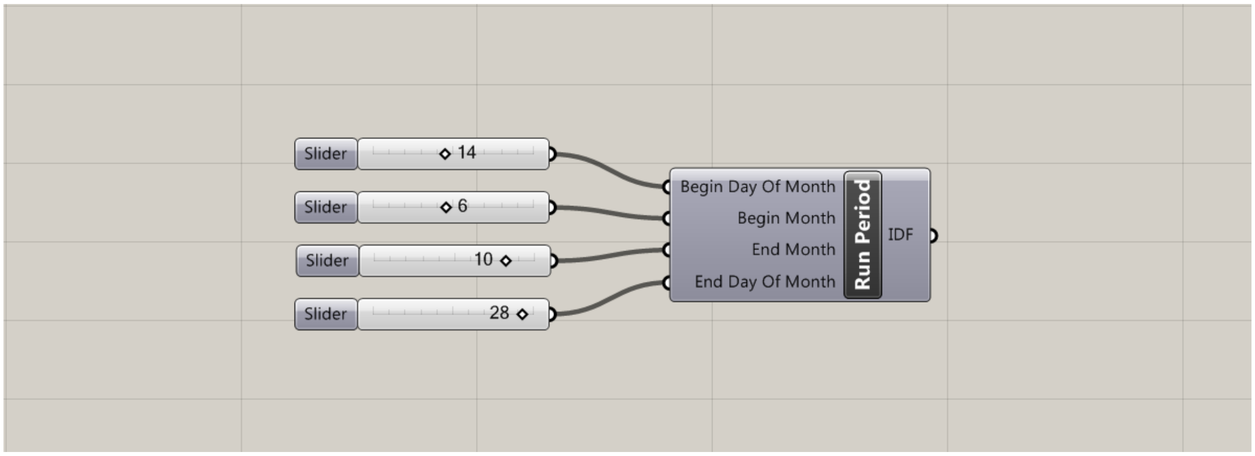
**Window Brep:** Input Window Brep as a Brep.

**Assembly Name:** Input Assembly Name using the "Assembly" Component.

**Outputs:**

**IDF:** Window information for IDF (Surface Input).

## Run Period




**Inputs:**

**Begin Day Of Month:** Input beginning day of month as an integer (1-30).

**Begin Month:** Input beginning month as an integer (1-12).

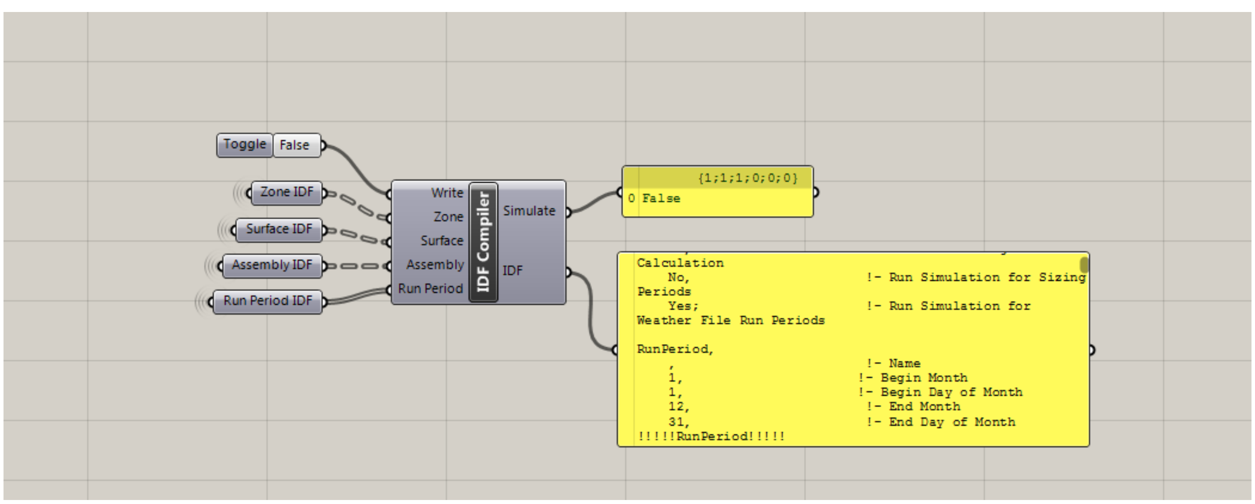
**End Month:** Input end day of month as an integer (1-30).

**End Day Of Month:** Input end month as an integer (1-12).

**Outputs:**

**IDF:** Run Period information for IDF.

## IDF Compiler



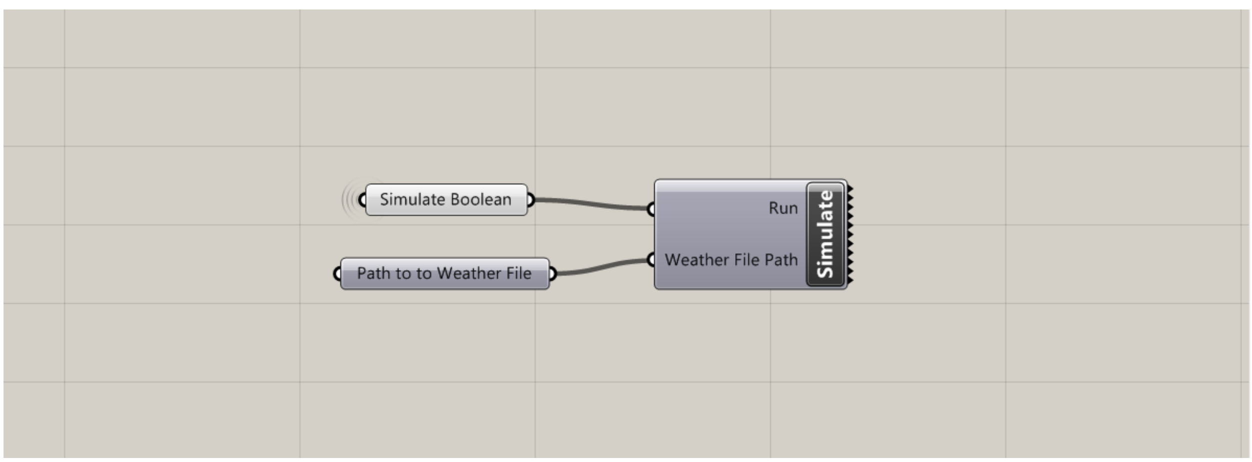
**Inputs:**

| | |
|---|---|
| **Write:** | Input True or False using the Grasshopper "Boolean Toggle" to write IDF file. |
| **Zone:** | Input "IDF" output from "Zone" component. |
| **Surface:** | Input "IDF" output from "Surface" component. |
| **Material:** | Input "IDF" output from "Project Library" component, "Gerilla Library" component, or "Material Maker" component. |
| **Assembly:** | Input "IDF" output from component. |
| **Run Period:** | Input "IDF" output from "Run Period" component. |

**Outputs:**

| | |
|---|---|
| **Simulate:** | Boolean trigger for "Simulation" component. |
| **IDF:** | IDF file to be used in simulation. |

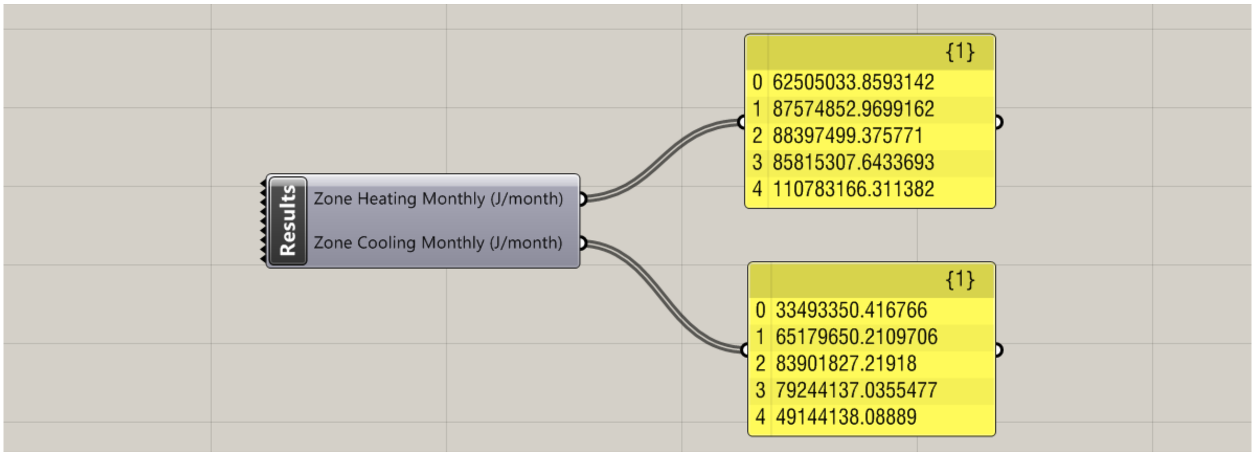
## Simulate




**Inputs:**

| | |
|---|---|
| **Run:** | Input "Simulate" output from "IDF Compiler" component. |
| **Weather File:** | Input weather file path as a string, or using the Grasshopper "Path" component. |

## Results



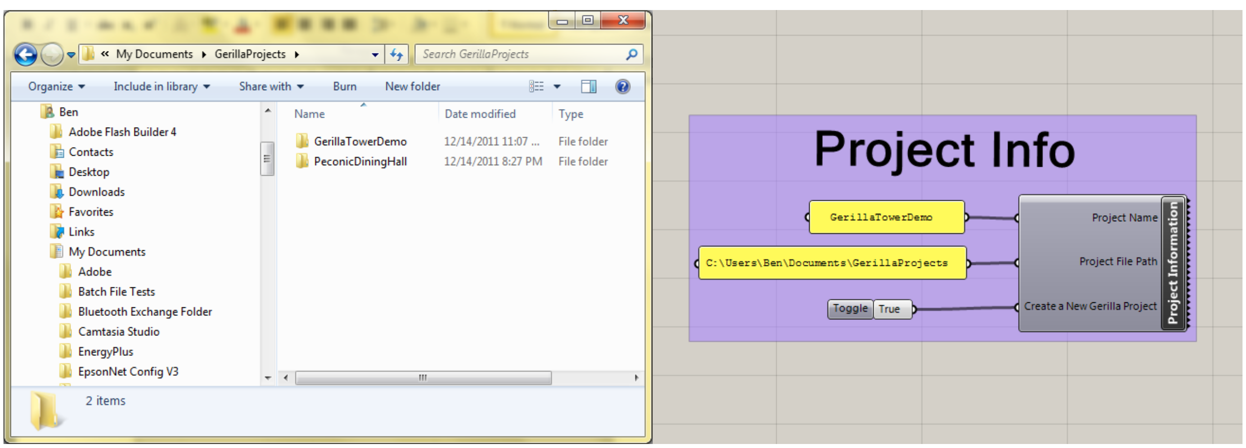

**Outputs:**

**Zone Heating Monthly (J/Month):** List of monthly zone heating values for all zones in simulation.

**Zone Cooling Monthly (J/Month):** List of monthly zone cooling values for all zones in simulation.

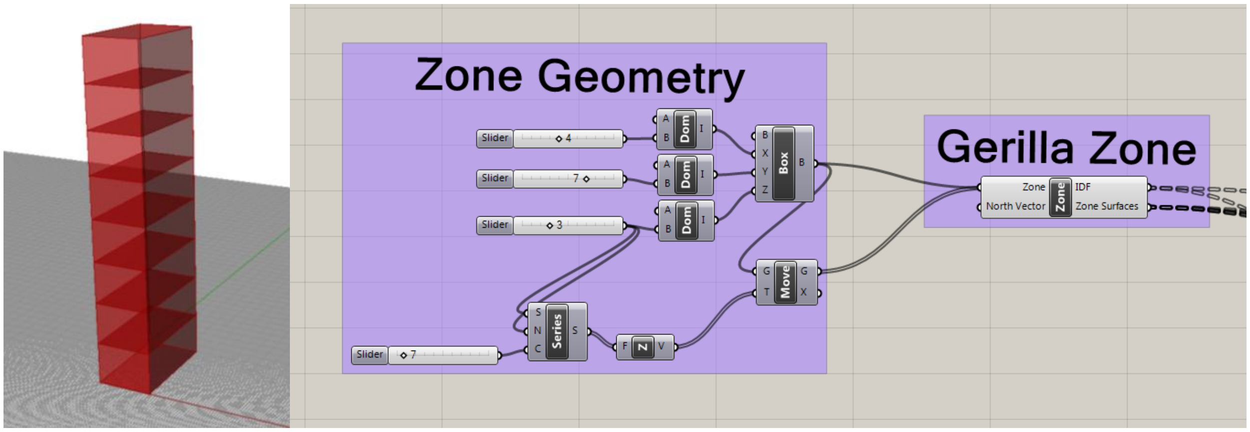
## 5. Example Gerilla Project: Tower

**Step 1:** Project Setup




The first thing to do with any Gerilla project is to use the Gerill Project Information component to setup the project name and folder location. Input a project name and the file path to where you would like to store your Gerilla projects. Toggle the Boolean to true to create the project. You will be warned if the project name you input already exists in that location. In order to overwrite that project you will need to delete the project folder (or move it) from the location. Once you have set the project name and folder location, the rest of the Gerilla components will know the name and folder location as well (no additional wires necessary here).

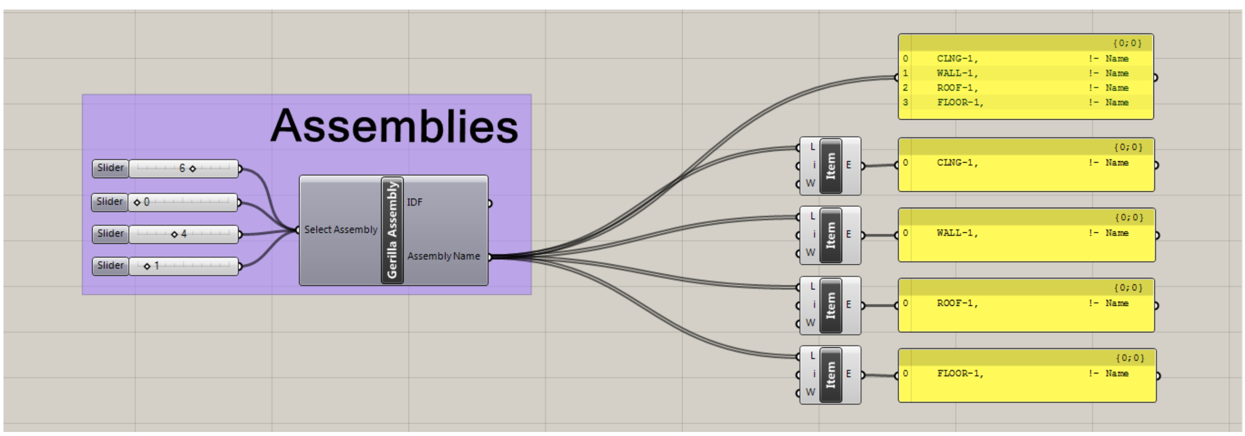
**Step 2:** Zone Geometry Generation and Zone IDF




All EnergyPlus zones must consist of air tight zones made of planar surfaces. In Grasshopper speak, we need closed Breps made of planar surfaces. For this tower demo we will use a parametric tower design using a simple Domain Box and a Series to create several zones stacked on top of each other. Input all of the zones into the Gerilla Zone component. You can also input a vector to specify the North direction for your project. For now we'll use the default North Vector which is the Y-axis. The Gerilla Zone Component will visualize all of the closed Brep zones it receives, so you can turn off the original geometry to error check yourself.

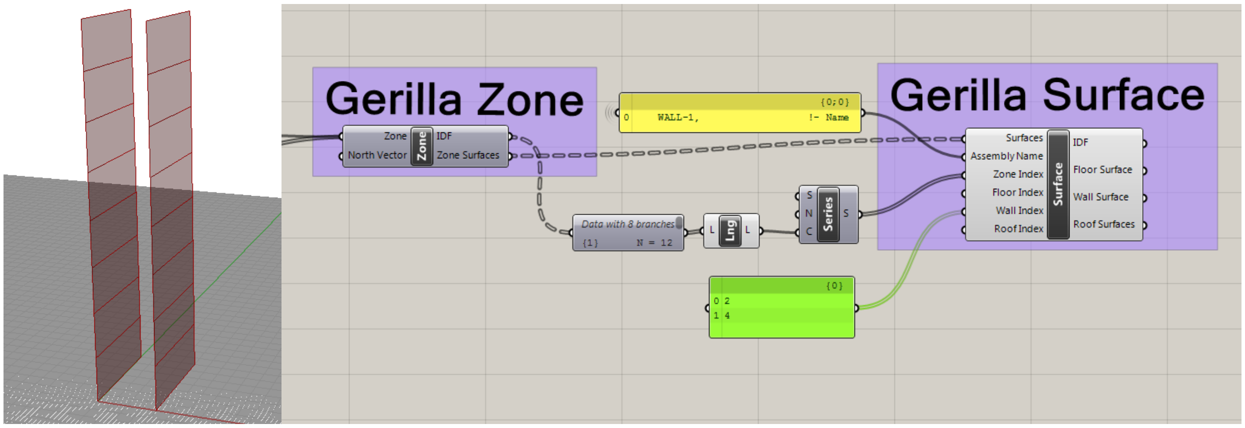
The Gerilla Zone component outputs an IDF string for each zone that will later be input into the IDF Compiler component. It also outputs each zone surface Brep with a unique branch path that will be deciphered by the Gerilla Surface component.

**Step 3:** Assembly Selection and Assembly IDF



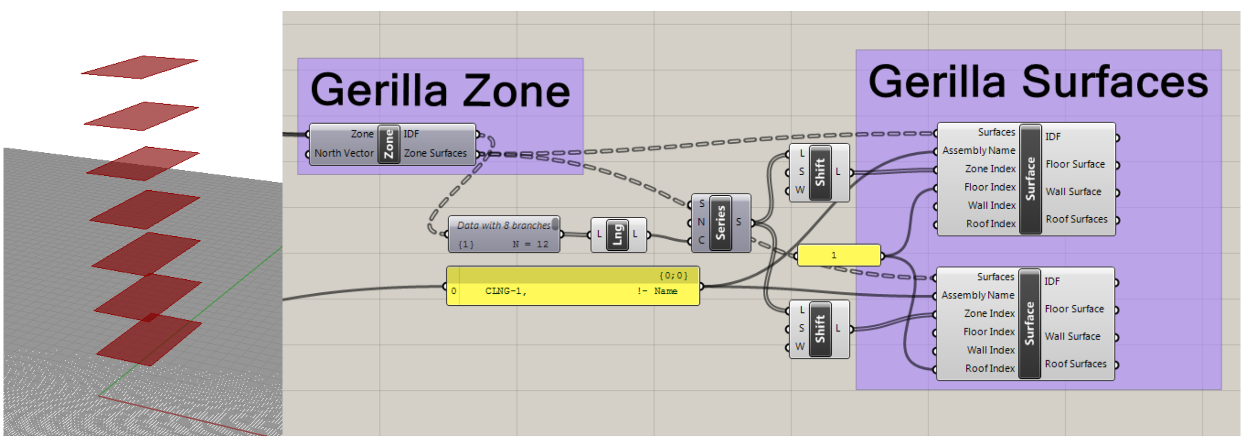

Gerilla comes with the ability to build two different libraries, a project specific library and a global Gerilla library, for both materials and construction assemblies. For now we will work with the default assemblies that are located in your Gerilla Assembly Library. You can use a slider to scroll through your Gerilla Assembly Library component and select each assembly you will be using. For the tower example, we will need a floor, roof, ceiling (inter zone floor slabs), and wall assembly. The Assembly Name for each assembly will then be assigned to the appropriate Gerilla Surface.

**Step 4:** Assembly Application and Surface IDF



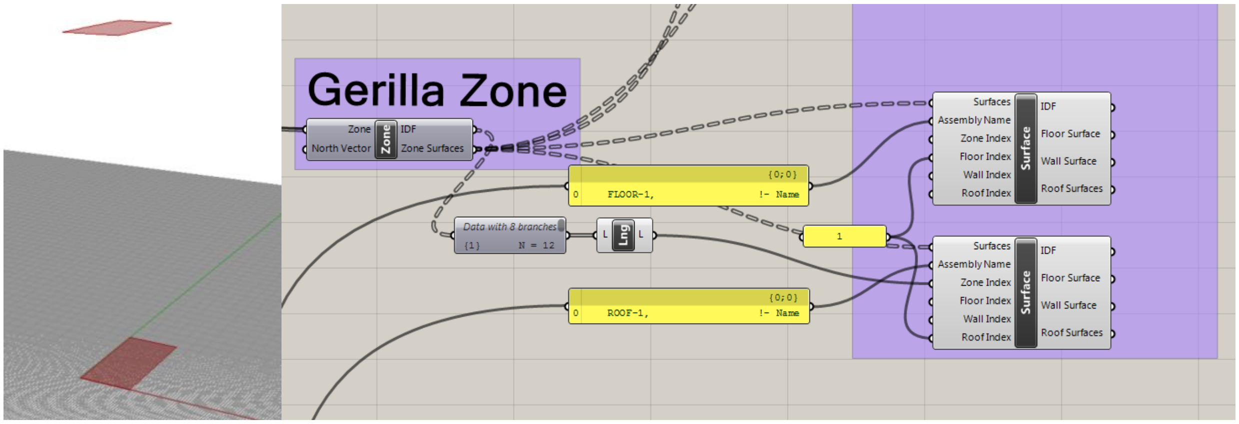

Now that we have sorted zone surfaces from the Gerilla Zone component and construction assemblies from the Gerilla Assembly Library Component we can apply the assemblies to the surfaces. Using a Gerilla Surface component we can select which zone surfaces we would like to assign assemblies to (they will eventually all need assemblies or will receive default assemblies). First thing to do is directly connect the Zone Surfaces output from the Gerilla Zone component to the Surfaces input for the Gerilla Surface component without flattening, grafting or do any other wacky stuff to it. To select the surface(s) you'd like to assign an assembly to, select the zone(s) to which it belongs by using the Zone Index input. In the example above we used a Series component that starts at one, steps by one and creates a series that is as long as the number of zones to generate the index of each zone. Currently we have 8 zones, so the series 1,2,3,4,5,6,7,8 is generated and input into Zone Index. We can then select the indices of the walls we would like to deal with. Above we have selected Wall Index 2 and 4 using a panel. If you turn off the Gerilla Zone preview you can see a Rhino visualization of the walls selected. This helps to determine whether or not you are selecting the intended zone surfaces.

Now we need to apply the wall assemblies for each zone surface. This step takes a bit of set up and requires some good organization, but once complete we'll be ready to simulate!

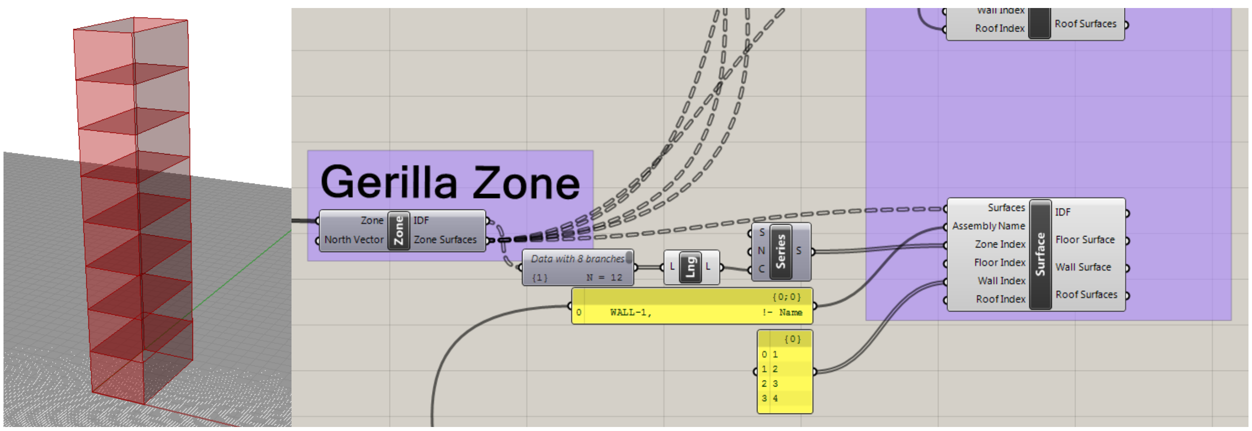



For the adjacent surfaces between the vertical zones we will apply a ceiling assembly. It is very important to note that EnergyPlus requires adjacent surfaces to have the same assembly assignment. Thus we need to select 2-8$^{th}$ floor surface and the 1$^{st}$-7$^{th}$ roof surface. We can quickly do this by taking the Series we set up a second ago and shift it 1 with the Wrap Boolean set to false to get a 2-8 series and shift it -1with the Wrap Boolean set to false to get a 1-7 series. These shifted lists will go into the Zone Index input and since there is only one roof and one floor surface for each zone we only need to input the integer 1 into the Roof Index and Floor Index. To complete our work here we will input our assembly from the

Gerilla Assembly Library component, into both the select floor and roof surfaces. We now have IDF strings to be sent to the Surface input of the IDF Compiler component.

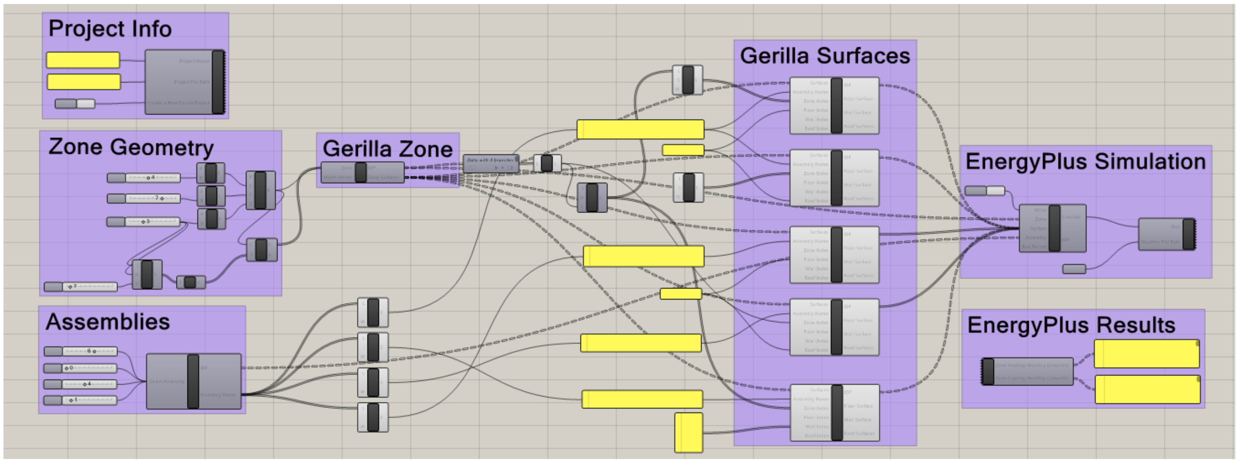



Above we are assigning a floor assembly to the floor of the first zone and a roof assembly to the roof of the last zone selected using a list length component.

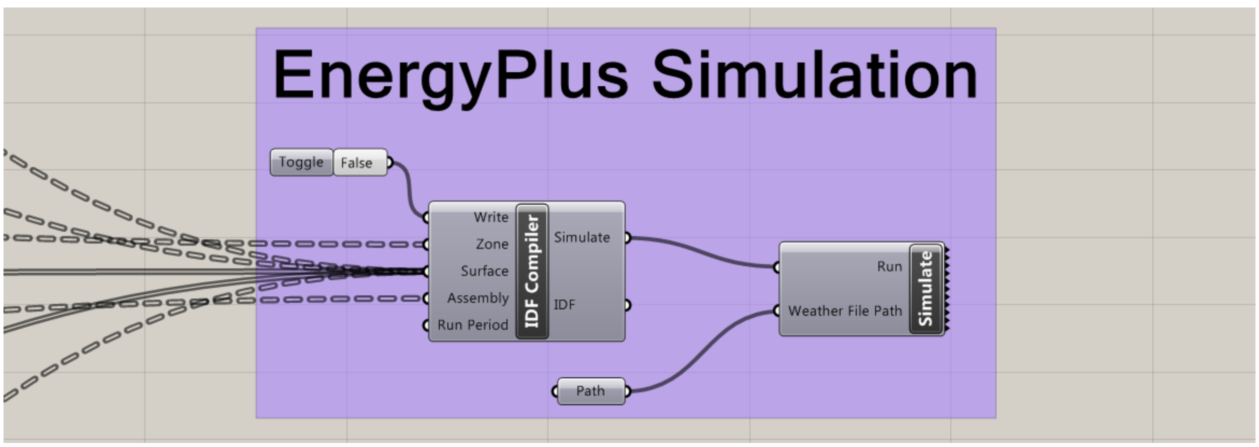



Above we are assigning a wall assembly to every wall surface using the Series component we used previously and selecting Wall Index 1,2,3, and 4 using a panel.

**Step 5:** Compile IDF and Simulate

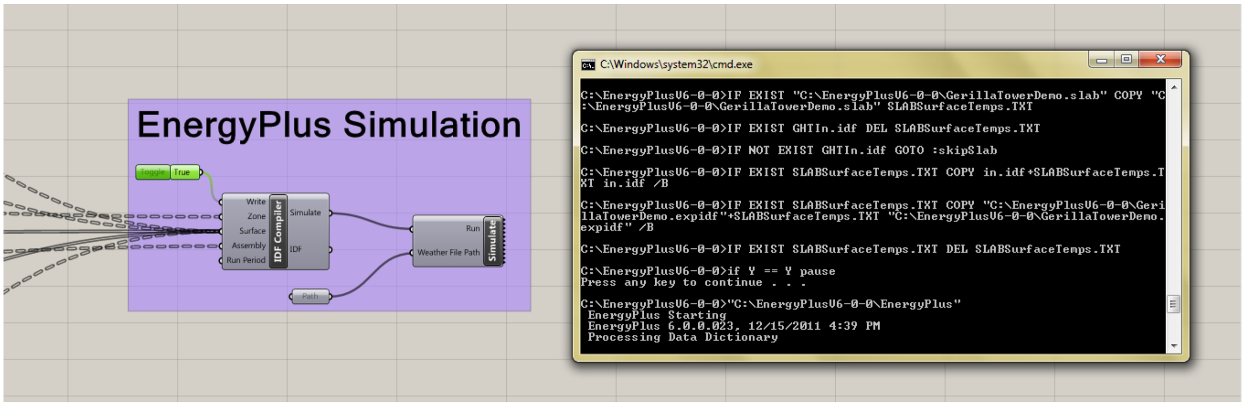



Now we will connect all of the IDF outputs from Gerilla components to the Gerilla IDF Compiler component.



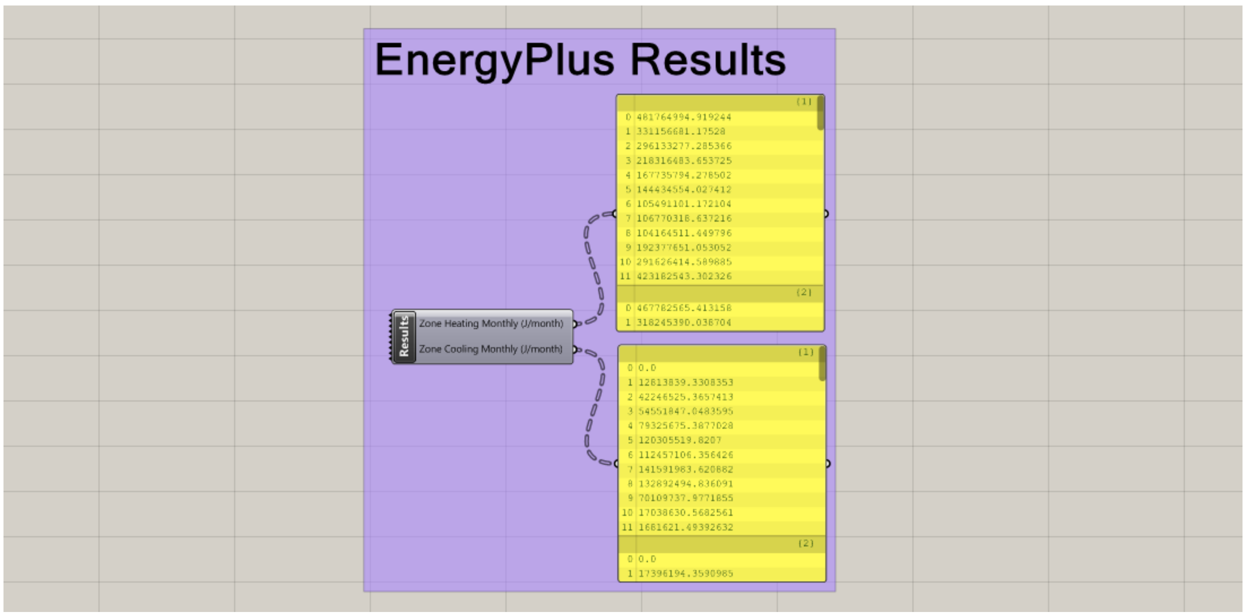

The Gerilla Zone component IDF attaches to the IDF Compiler Zone input. All of the Gerilla Surface IDF's connect to the Surface input. Connect the Gerilla Assembly Component IDF to the Assembly input for the IDF Compiler component. The default Run Period is all year, so we don't need to plug anything in here. Attach a Boolean to the Write input for IDF Compiler component.

When we toggle the Write Boolean to True, the IDF Compiler will write an IDF with all of your Gerilla project data that will be used in the EnergyPlus simulation. You can attach the Simulate output Boolean to a EnergyPlus Simulation component Run input to trigger the simulation. Make sure to first attach a file path to your .epw weather file which can either be selected from the EnergyPlus folder or downloaded from the internet.




When you toggle the Write Boolean to True it will initiate a simulation using your IDF file and the weather file you selected. A pop up window will give you a very quick play-by-play of the simulation.

**Step 6:** Simulate and Results




If you pull down an EnergyPlus Results component you can access two sets of simulation results, either monthly heating or monthly cooling measured in Joules. The results are provided as a DataTree of String with a branch for each Zone (branch paths are the same as the original Gerilla Zone output IDF branch paths) and a list of 12 strings (1 for each month) in each branch.