S206E057 -- Lecture 11, 10/13/2023, Grasshopper – a preview

Grasshopper is a "graphical algorithm editor" that requires no knowledge of programming or scripting. It has been included as a part of Rhinoceros component since Rhino 6 for Windows after years of Beta test. While Rhinoceros for Mac was released in June 2015 (version 5), Grasshopper for Mac had not been included at that time. But it's been installed in Rhino 6 since then for Mac. Required by Robert McNeel & Associates, all systems must be installed with the latest release of Rhino 7.

Background

Rhino can't keep track of how the model was constructed after it is completed and saved. Some 3D modeling applications could remember all set of curves that a surface was created; when we change these curves, the surface will change accordingly as it was newly created. This is called History of parametric modeling in some sense. Maya and 3D Max implement the concept of "History". They support the design process by enabling us to go back to earlier steps of the modeling process to change them and change the result as well. Rhino is very limited in this regard. Thus, the concept of "Explicit History" was developed in Rhino, which is the Grasshopper component. Explicit History differs from traditional modeling history in that we explicitly construct the "History" and the set of commands generating objects. Such history of commands could be followed by Grasshopper (GH) shown in Rhino viewports.

The origins of GH can be traced back to the functionality of "Record History" in Rhino3d version 4. It allows users to store modeling procedures implicitly in the background as users making models. Before 2008, David Rutten at Robert McNeel & Associates intended to provide some control over the recorded history to expose the history tree, and to provide users with chances to develop logical sequences. This would be the chance to let users develop their own design algorithms. Thus, Grasshopper was added to Windows version as a plug-in application in March 2008 after series of testing and modifications.

Why Grasshopper?

Why do we want to use GH? Usually, designers would like to use solid modeling to deal with complex models. Sometimes, users want flexibility with the ability to quickly change fundamental attributes of a complicated model and the ability to make complex formations through repeating simple forms. Sometimes users want to use mathematical functions to control shapes. Such flexibilities could be accessed from within Rhino by using scripting languages of Python or Rhinoscript. These scripting tools offer powerful control over Rhino's modeling commands, including some that are not available through the graphic interface. Yet, using scripting language requires in-depth knowledge of computer programming techniques, which users might not have. However, GH combines a graphical approach of working in Rhino with algorithmic techniques used in scripting. Thus, users don't need to have a high level of programming or scripting knowledge experience to begin with. That is why GH is popular.

Installing GH in Rhino 5, 6, & 7

In Rhino 6, GH is installed in the "Standard" folder. In Rhino 5, the GH could be downloaded from http://www.grasshopper3d.com/, or at URL: http://download.rhino3d.com/Grasshopper/1.0/wip/rc/download/, and saved in "<ProgramFiles>/Rhinoceros 5.0/Plug-ins/" folder. Then, initiate the "grasshopper_0.9.76.0.rhi" file, GH should be installed and be run by the "_Grasshopper" command. In Rhino 6, these steps are skipped. GH could be run by clicking the GH button and it is a part of the Rhino 7 system.

Basic concepts of GH and what is algorithm?

Algorithm is a step-by-step procedure of executing functions or actions to get things done. In the fields of math and computer science, algorithms are formula or calculation methods for solving problems. **In computer modeling, it is used as lists of successive steps with well-defined instructions**. These steps are not necessary to be sequential. In computer programming, an algorithm could be explained as a function that has a beginning and an end in coding. Right at the beginning of the function, the format of applied data shall be declared first. Input data is processed inside the function steps by steps and transformed into certain types of data for output, which is the final state of the function.

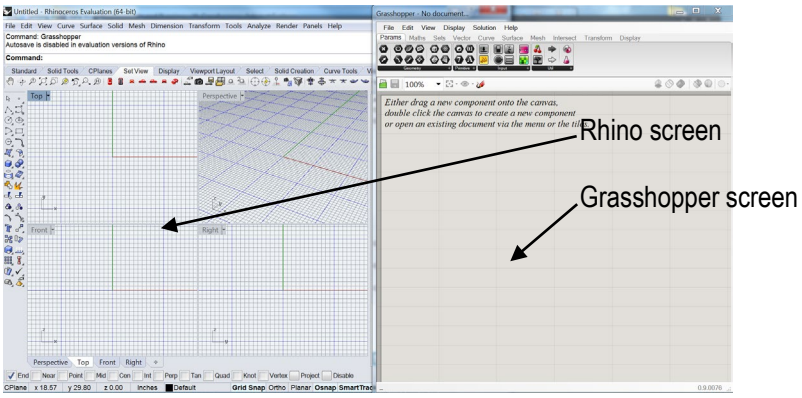
What is data?

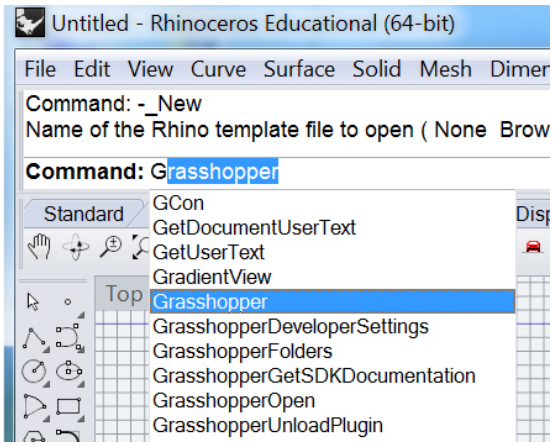
Data is a body of facts. In Information Technology, data is unorganized raw facts with no context or organization. When data is processed, structured, or contextualized; it becomes information. Data can be dimension, color, shape, or size relating to

qualitative in nature, or length, volume, or area of quantitative nature. Inside GH, we could get the geometry data of the model in Rhino, which could be the center point of a circle, the length of a line, the curvature of a surface at a given point, coordinate points of lines, or lines of surfaces, etc. The other kind of data is locally defined data, which could be numbers, names, or certain things defined through variables and input by users.

Initiate GH:

Click the "**Grasshopper**" in the "**Standard"** folder or type the keyword of Grasshopper to start the application. In Rhino 7, there are new tool of GrassopperPlayer, Grasshopper PluginList, and Script Comiler added to the menu.

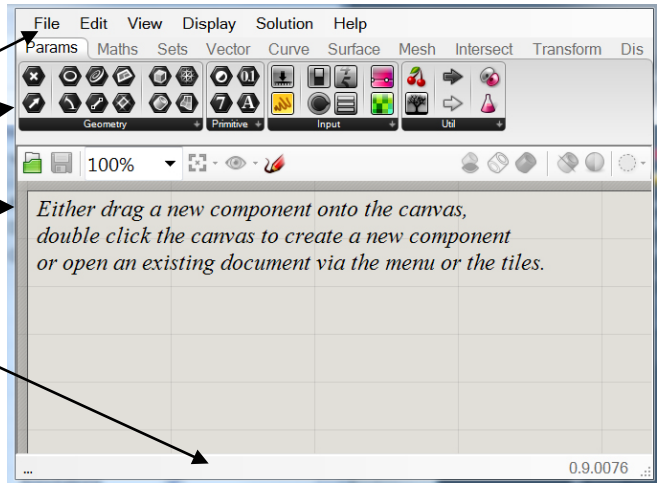




Grasshopper interface:

The GH window has four major sections of the menu bar (the title bar) area,
component tabs area,
the canvas area,
and the status bar area at the bottom.

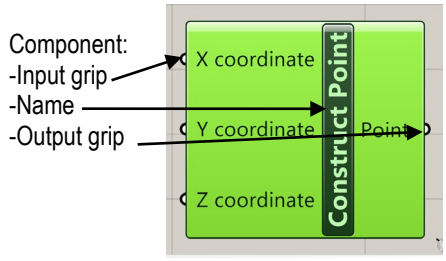
Each component has its own contextual menu by moving mouse on top of it to find it. Right click the right mouse button will get help information.



Basic concepts of GH:

The objects that make up a GH file are called definitions, which have two main classes of "parameters" and "components". **Parameter**s store data, whereas **component**s process data. Definitions are set up by double click the canvas and type in the definition name.

1. **Components** are generally divided into three main parts. The left part is the input grips with semi circles, the middle one is the name part with either icon or name abbreviation, and the right side is the output grip. The input grip takes in data to use in whatever operations the component handles. After the data is processed, new data will be generated and passed on through output grip to other components.



2. **Parameters** store data that is manually entered by the user, or data that is inherited from other components.

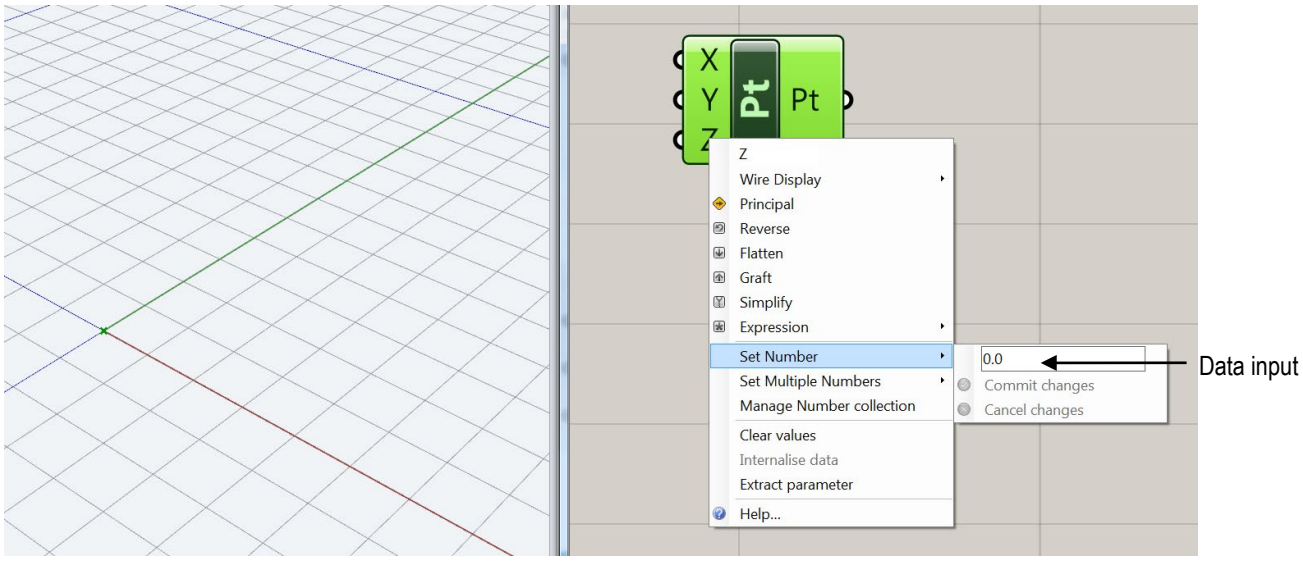
Note: The geometries created in Rhino can be used by GH and internalized as a part of GH geometries. The **GH** created geometries in Rhino viewports can't be selected nor modified in Rhino scene, unless they are baked from GH to Rhino.

Setting input data:

In GH, the fundamental and critical techniques are the methods used to get data. Without having data, definition components could not do anything. Input data to a component (definition) usually is done by three methods:

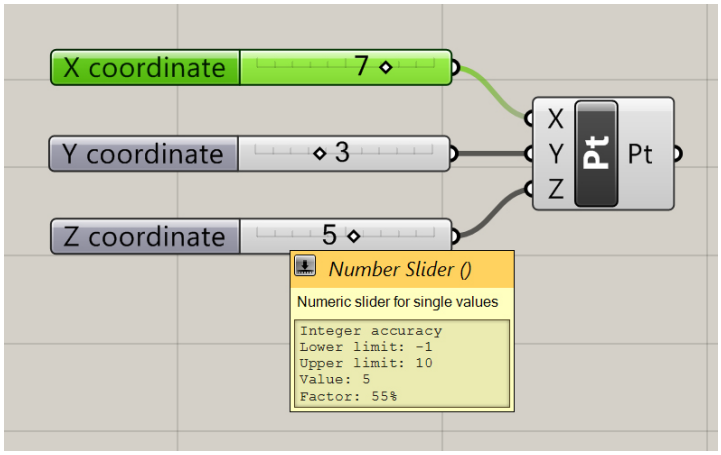
1. **Local definition**, values are manually typed in the contextual menu, which reside in the GH environment. Since the data is persistent inside the component, it needs manual input for modifications.

   For example: Double click the GH canvas, on the "**Enter a Search Key**" window, type "**construct point**" to create a point component, which will generate a physical point at the base of 0,0,0 on Rhino. Then click the right mouse button on X, Y, and Z > select Set Number, enter the value representing the coordinate value in the box respectively, and click "**Commit Changes**". The point in Rhino will be updated as well. (Note, **construct point** function could be found in the **Vector** menu > **Point** folder > the **first tool** is the construct point function.)



2. **Direct definition**: other components or parameters feed data into the component.

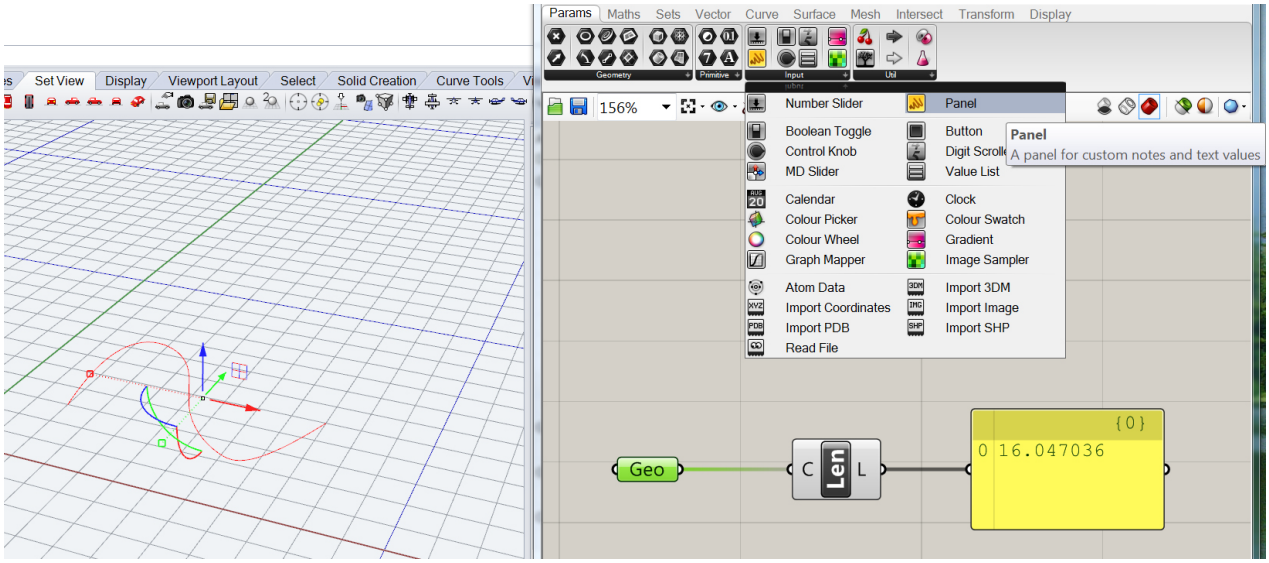
   Define three "**Number Slider**" components and link them to the X, Y, and Z input part of the Point component. The output data of the slider will provide input data to the point. In sequence, the location of the point in Rhino will be changed accordingly in real time. Such a link or connection between two components is done by "**wires**". The entire structure of these wired components as a group on executing sometime is called a definition or an algorithm.
   If the wire is drawn in **single line**, then it has one data item passing through the wire. If the fancy wires draw in **double line**, then it is a list for passing multiple values. A **dashed double line** shows multiple lists of data items are passing on, which is a list of lists.



3. **Direct import** from geometry modeled in Rhino.

   1. Draw a **curve** in Rhino.
   2. Define a **Geometry** in GH, Geometry contains a collection of generic geometry. It could be a circle, a rectangle, or a curve. If it is a curve, then we could use the output data to measure its length. If it is a circle, the output data provides the value of centroid coordination and its area value.

3. For the Geometry component
   Right click > **Set one geometry** > Select the curve in Rhino.
4. Define a **Length** component to get the output data of length available.
5. Define a **Panel** component to display the data.
6. Link the *geometry* to *length* to *panel* to see the length shown on panel.



Note 1: The length definition will take a curve and measure its length shown on the output grip, which provides the measurement through panel window. In this example, the output data of Geometry component is the other component's input data. Such a data comes from the Rhino object of "curve". Whatever the data is updated, the GH will receive an update as well. To disconnect the link, right click the component's output grip on the right side **> Disconnect**, select the component from the list to cut the link. It is the same as selecting the input side of grip to disconnect data input from other components or parameters.

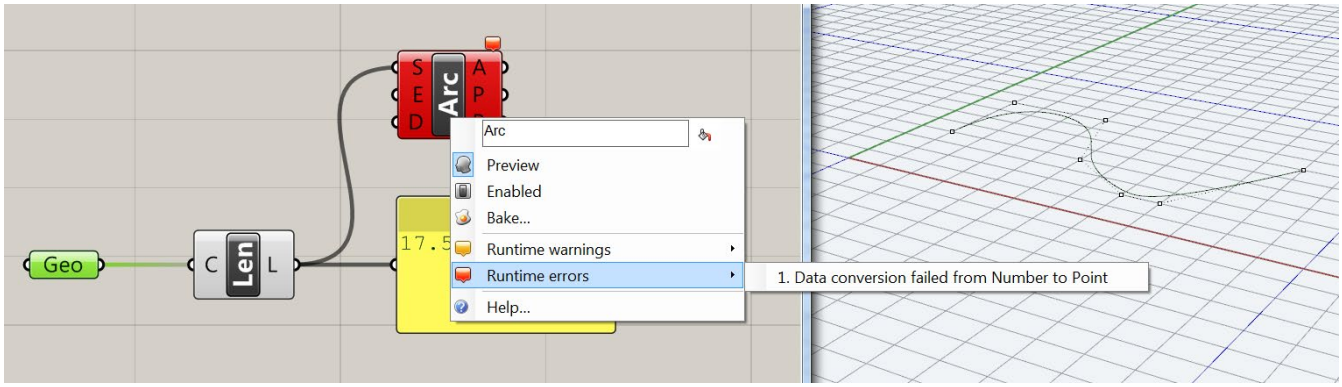
Note 2: To embed the Rhino geometry into the GH file, select the Geometry component, right click the mouse button and select **Internalize data** to save the Rhino geometry to GH functions. After the GH file is saved, the geometry is also saved in the GH file. Thus, we don't have to worry about managing or finding geometry data after the GH files are loaded into Rhino. But that data created from GH, from that point on will persist. On the other hand, objects in GH are visible in Rhino but not tangible. To edit a GH object in Rhino, it must be baked into Rhino by right clicking the GH component, selecting **Bake** to export the object to Rhino. Baked objects are Rhino objects that cannot be edited and affected by any manipulation in GH but could be edited in Rhino.

Representation of GH in Rhino

Once we started to create a GH component with number sliders to control the dimension of the geometry, we made GH objects and visualized them in Rhino's viewports. This is a live connection. If we adjust the grip on the slider, the GH program will re-compute a solution and display the update. The geometry preview we see in Rhino is a lightweight representation of the solution and it automatically updates.

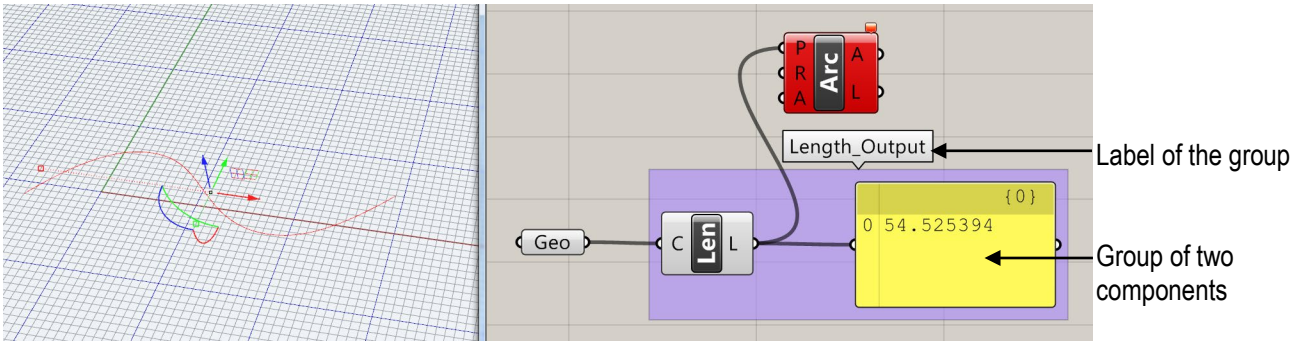
Error message:

Rhino has three different color schemes to signify error messages. If the object color is gray, then the component is correct. If it is orange, it is a warning. Algorithms with warnings may still function, but you must be careful to check and understand why the warning comes up. Most of time, it is the input data problem.
Particularly, red color of the component means an error in its inputs. Methods to find error is right clicking the rectangle shape on top and select "**Run Time Errors**" to understand the possible error source or reasons. It usually is caused by empty input or wrong data type to match with the required type for operations.



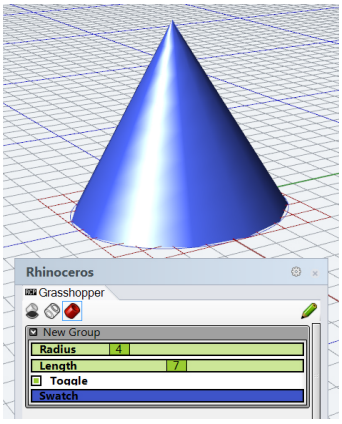
Organize components and parameters:

When components are getting complicated, we could use group to group components together and tag them with a label. For instance, select the length and panel definitions, select the **edit** menu > **group (**or simply type Crl + G**)** to group them together and name it on the top of the window. In this example, similar components or connected components could be organized by groups to differentiate the functions for making the entire organization recognizable. The other function is **cluster**, which also group a group of components together.



Exercise 1: 3D objects -- control the color of an object, which is a cone, with a logical method.

Create a cone in GH, which will change colors, dimensions, and visibility.



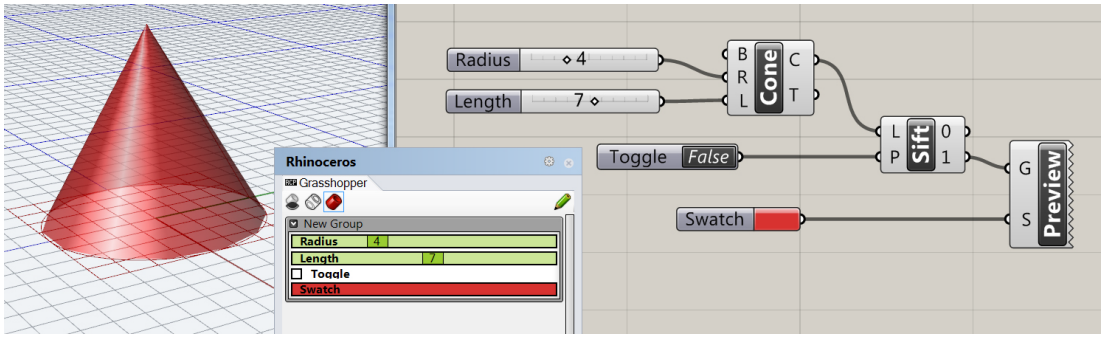
-- Define a **Cone**, and put two number sliders for R & L. A cone has three inputs, the base point of x and y, a **R**adius, and a **H**eight.

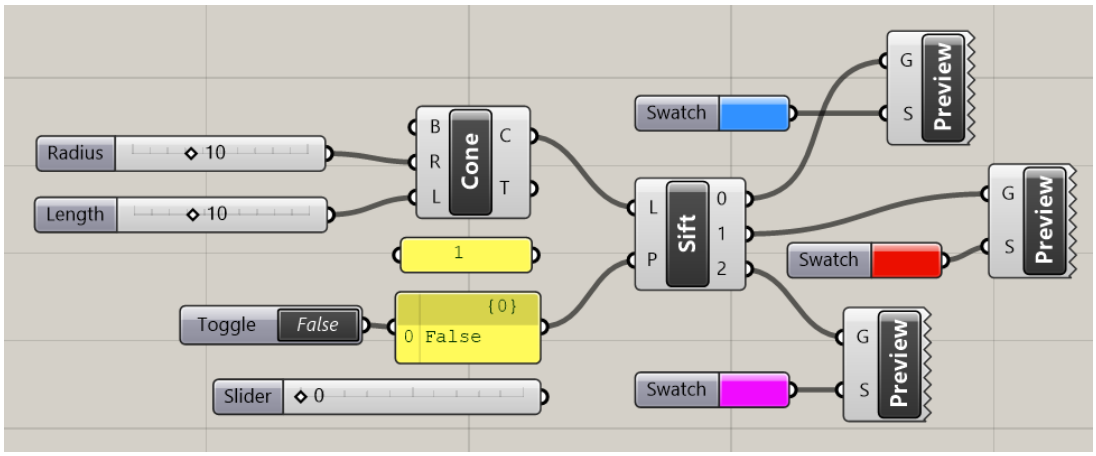
-- The results of the cone will be "**Sift**" (to filter out) to either 0 or 1 output (in this case). The function of **Sift** locates at **Sets menu > List > Sift Pattern**. Sift (in short) contains list order. Instead of creating two output lists that contain only the items selected, **sift** will actually include nulls for those items not selected. Combination of **Sift** and a **Boolean Toggle** is therefore ideal to simulate conditional statements. In this case, Sift is sorting the item by 0, 1. Zoom in on Sift to add more outputs.

-- A **Boolean Toggle** will provide true (1) or false (0) value for the sift component. True is 1, and false is 0.

-- If it is true, then in the output of 1, create a **Custom Preview** (or in Display > Preview > Custom Preview) to display a particular color defined by "**Swatch**" component, otherwise keep the original color..

The color swatch component is a special function that could quickly set up individual color for the geometry. Custom Preview component allows users to preview the **Shader** of the **Geometry**. In this example, Geometry is the value coming from the Sift function that has 0 or 1 to control the list of the geometry variables. The Shader goes with the color value given from Swatch.





This example controls the color of a cone by numbers. True is 1 that shows blue, and false is 0 that will go with cyan color. These control parameters of number sliders, Boolean value of true or false (or number sliders), and color change swatch could be exported to a **View menu** > **remote control panel**. Right click the Radius slider > **Publish to Remote Panel**. The remote control will be easy to handle inside the Rhino without showing the GH window. Color could be controlled by slider (see the image above).

Save files in GH:

The components (or definitions) built up in the GH could be saved as ".gh" file, which is a default binary format file with small file size. Because of its small file size, it could be read quickly by GH. But it is only readable by GH. The other file format is ".ghx" that could be read by other systems but the file size is bigger. The geometry created in GH will be reviewed in Rhino. They are not editable by Rhino. Especially, after the GH file is closed, the geometry in Rhino disappears. These GH previews could be converted to Rhino and editable in Rhino through the "**bake**" command. Thus, select the geometry of a component (for instance, the corn component) and click the right mouse button and select "Bake". It brings up the bake attributes dialogue, which allows the control of how to convert the GH geometry into Rhino geometry. The newly converted geometry will be Rhino editable geometry.

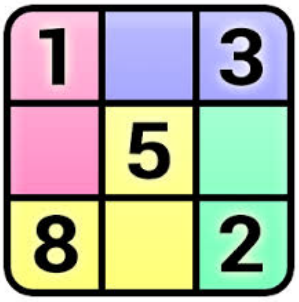
Other Interface of Math and Logic Functions

1. Mathematic equation: Users could build up their own math formula.
2. Logic expression (the Boolean operation of true of false)

There are some challenges on setting up if-then-else conditions in GH, which will be explained later.

Exercise 2: 2D shapes -- Sudoku

This is an exercise of GH 2D example in Rhino.



The basic concept of this example is to randomly take four cells away from the nine cells created in the 3 by 3 grids, which is the one-dimensional Rubik cube.

1. Use 2D "**Square Grid**" (or type **sqgrid**) to generate 3x3 grid with the size of 10 unit. (Square Grid is 2D grid with square cells. Among the input variables, plane represents the base plan for the grid. Size is the size of each grid cell. Extent X is the number of grid cells in x direction of the base plane and Extent Y is the number of cells in the base plane y direction. For the output grips, **Cells** will provide a list of these nine cells; and the **Point**s output will provide coordination of four sets of corner points.)

2. Results of the cells are provided on a list.

3. Use "**Random Reduce**" to reduce the number on the list.

4. **Flatten** the list given by "SqGrid" to generate a clean list. It is because that data inside this parameter must be flattened to make a completed, lineal, list. The method of doing it is to move the mouse on top of the list input grip and right click > select **Flatten**. Results are that the list item of the data becoming a string which could be understood as continuous curves, instead of four segments of lines.

5. Control the reduction number to 4.

6. Random seed number (S) could be random number (97 and 40 for a cross shape).

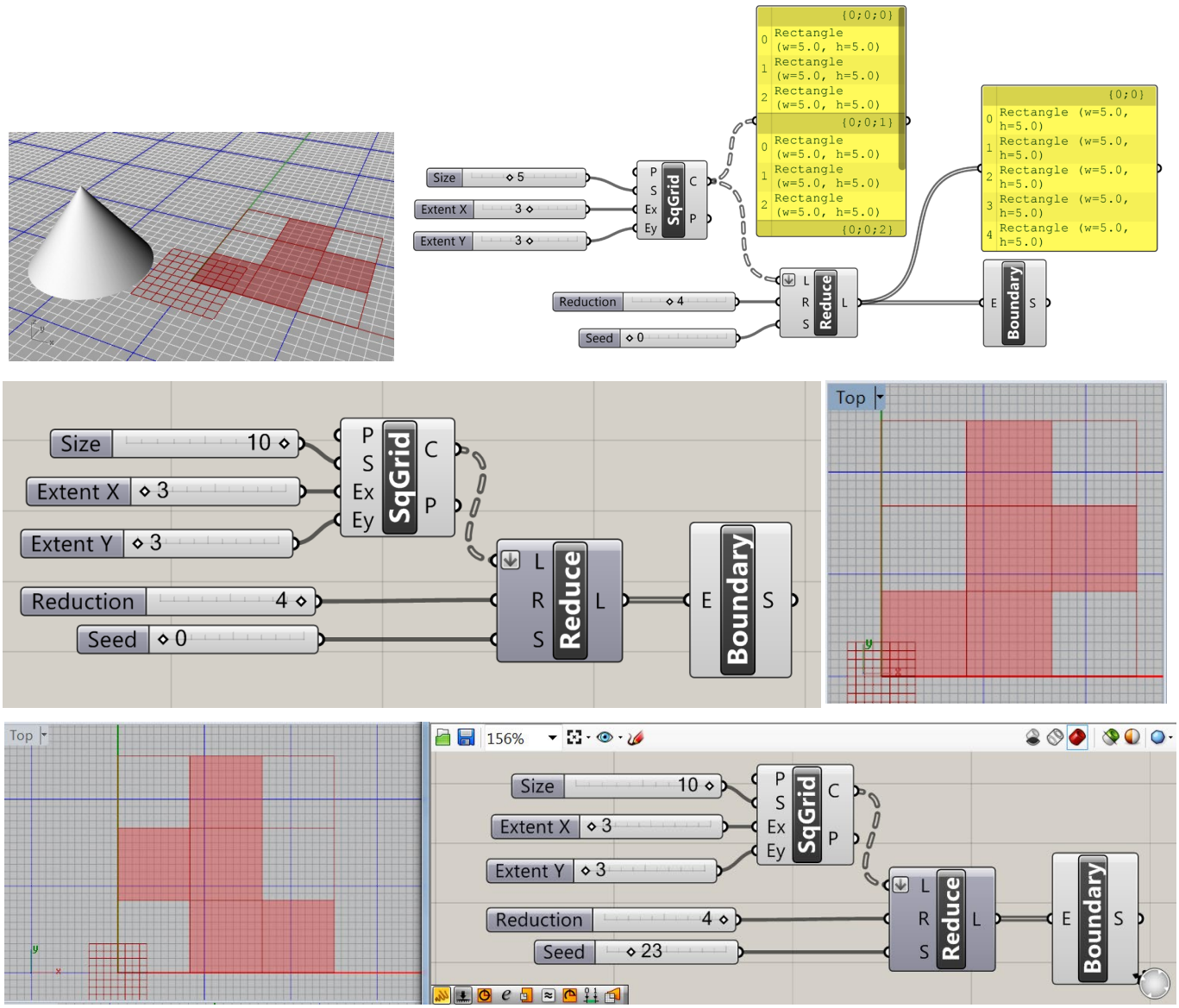
   (Note: Concepts of generating a random number started in the ancient time around 13 Century. But John von Neumann redeveloped it in the 1960s. Here is the fundamental method. For instance, for generating a sequence of 4-digit pseudorandom numbers, a 4-digit starting value is created and squared, producing an 8-digit number (if the result is less than 8 digits, leading zeros are added to compensate).

   The middle 4 digits of the result would be the next number in the sequence and returned as the result. This process is then repeated to generate more numbers. In this example the 4-digit number is the seed number. Of course, this algorithm is not a good method, and has some problems – what happened if the middle four numbers are 0000?)

7. Apply the reduced list to create grids by "**Boundary Surface**" (Boundary Surface component will create surfaces for an enclosed curve, which shall be a collection of boundary edge curves as input data on the left. The result on the right is the planner surfaces.)

8. Use File menu > **Export Quick Image** (or Clt + Shift + Q) to save the GH component into a png file. In this exercise, save the file as gh format with "stdID#_Fname_Lname_Subject" filename. The following images are the jpg image (**ViewCapture ToFile)** from Rhino and png (**Export Quick Image**) image, on the right, saved from GH canvas.

9. The random number will create the filled pattern, for instance, 40, 56, 97 and 136 are the cross shape, 82 is the T shape, 165 is the U shape, 253 is the T in 90 degrees, whereas 623 is the zig-zag shape.



Note:

1. Regarding the data structure and processing, GH uses ***data-trees*** and **lists** to structure data. GH doesn't use names for its objects; it will use a list of numbered objects. Sometimes it is hard to see what object a number is in a list that represents. These lists, however, have a major advantage in that their objects are defined in the lists. Such a data structure could be seen through the list shown on the panel, see the top image on this page.

3. Concept of data processing on **data tree** could be found on: http://wiki.bk.tudelft.nl/toi-pedia/Grasshopper_Data_Tree_Editing.

4. Details of the GH basics could be found on: http://wiki.bk.tudelft.nl/toi-pedia/Getting_Started_with_Grasshopper.