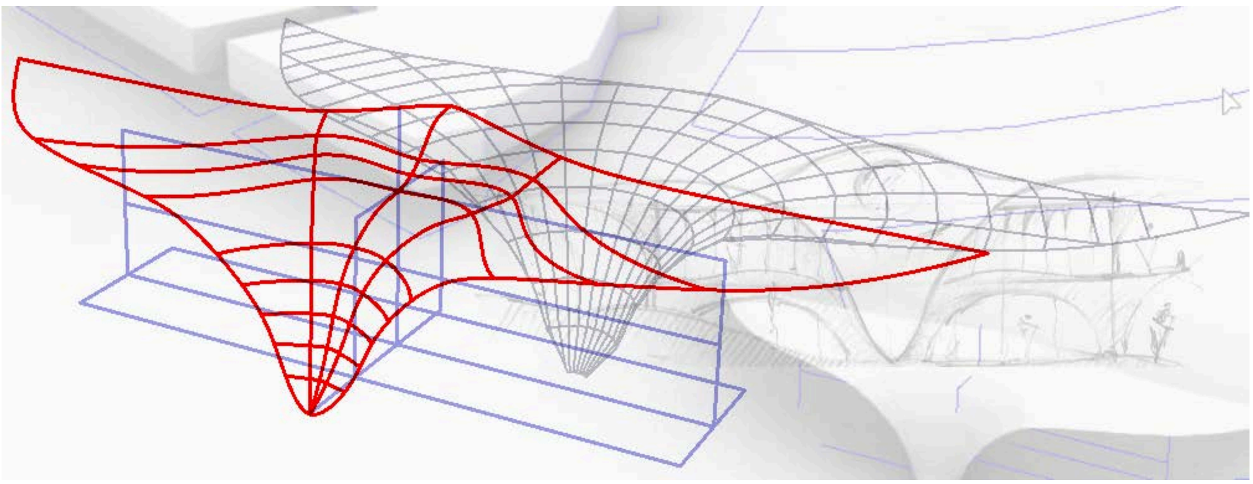
Rhinoceros

Modeling Workflows in Architecture



Preface

View video tutorial...

Digital modeling has taken leaps and bounds in the past few decades. From basic tools used to optimize drafting, to an intelligent medium that is changing how we design and build. Digital means are infiltrating every aspect of design and construction today. This fast pace development of technology, and the inclusion of end users in the making of digital tools, is forcing a volatile and fast-changing platform with numerous workflows. In commercial software available today, we can distinguish four dominant approaches to digital modeling: direct, algorithmic, object-based and parametric. These approaches are described in part three of this guide. Rhinoceros and its plugins support all four modeling methods, which makes the task of presenting Rhino to new users very challenging. The versatility of Rhino and the ease in which they can be extended through plugins means that it is very hard to be specific about how Rhino is used, by who and when.

While contemplating best ways to present the Rhino modeling environment to architects, I thought it might be best to base the presentation on modeling methods and workflows, rather than a description of a finite set of tools. My hope is that this approach will promote critical understanding of the tools, and encourage users to be creative in how they use Rhino. This is perhaps the best way to understand Rhino's best potential, but also help challenge and push it in a positive direction.

The content is presented in three parts. The first part presents common modeling vocabulary in 3D modeling, and how it manifests in Rhino. The second part, which is the bulk of this document, is about modeling workflows in architecture using Rhino core modeling tools. It covers four critical stages of modeling: setting up the modeling environment, concept modeling workflows, detailed modeling workflows, and prototyping. The third involves a brief discussion of digital modeling methods in architecture, and how Rhinoceros, Grasshopper and other McNeel plugins fit within these methods.

While this is by no means a comprehensive manual, I hope that future editions will expand to include other modeling methods, and additional design and production workflows.

Rajaa Issa
Robert McNeel & Associates
August, 2020

Part I: Modeling vocabulary

[*View video tutorial...*]

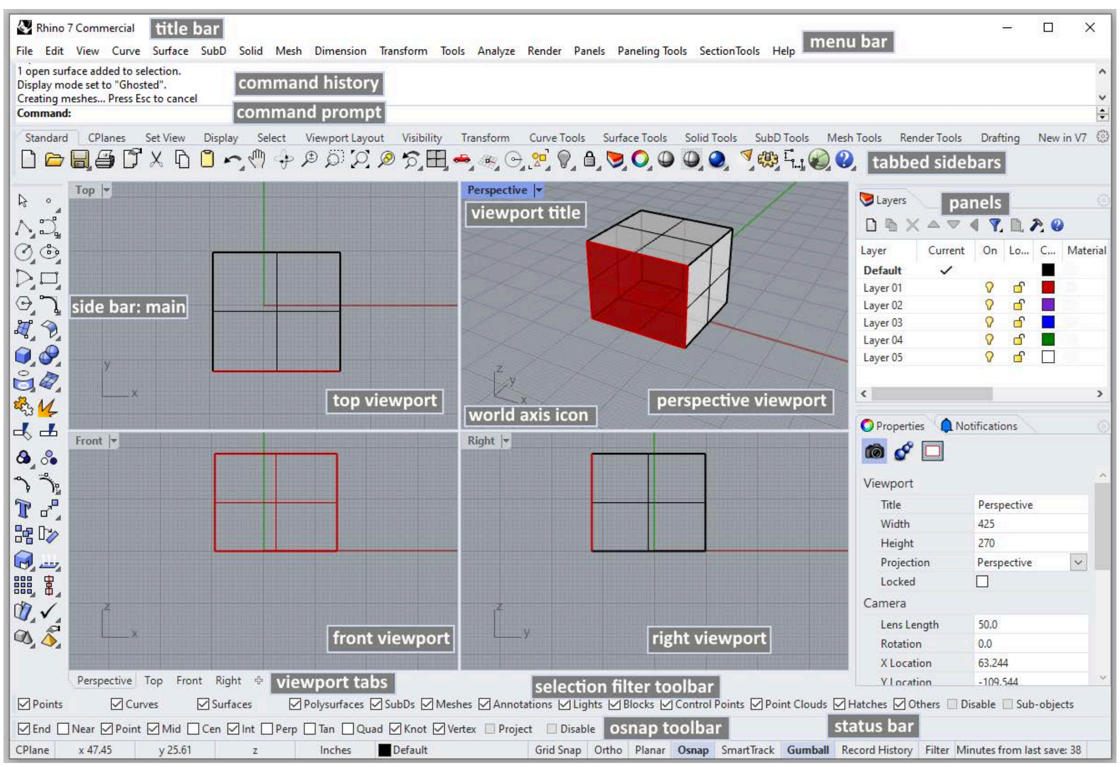
3D modeling is the process of using computers to create, visualize, edit, and share three dimensional forms. Computers use geometry as the primary conduit to represent design form and use data to attach information to geometry. Most 3D modeling applications share common vocabulary and the very first step to learn how to model within any environment is to understand its vocabulary. That includes workspace organization, access to different tools, modeling controls, what geometry is supported and how it aggregates and how data is applied and managed. Visualization, analysis and presentation of 3D forms is also an integral part of modeling applications. Models are stored and exchanged using files and file formats are either specific to the application or more widely used across multiple ones. This chapter discusses the modeling vocabulary in Rhino.

1.1 Workspace

1.1.1 The user interface

[*View video tutorial…*]

Once you open the Rhino application, you will see a screen with a graphic area in the center (viewports) surrounded by menus, toolbars, controls and panels. Most of the workspace elements can be rearranged, removed or expanded.



The Rhino interface in Windows

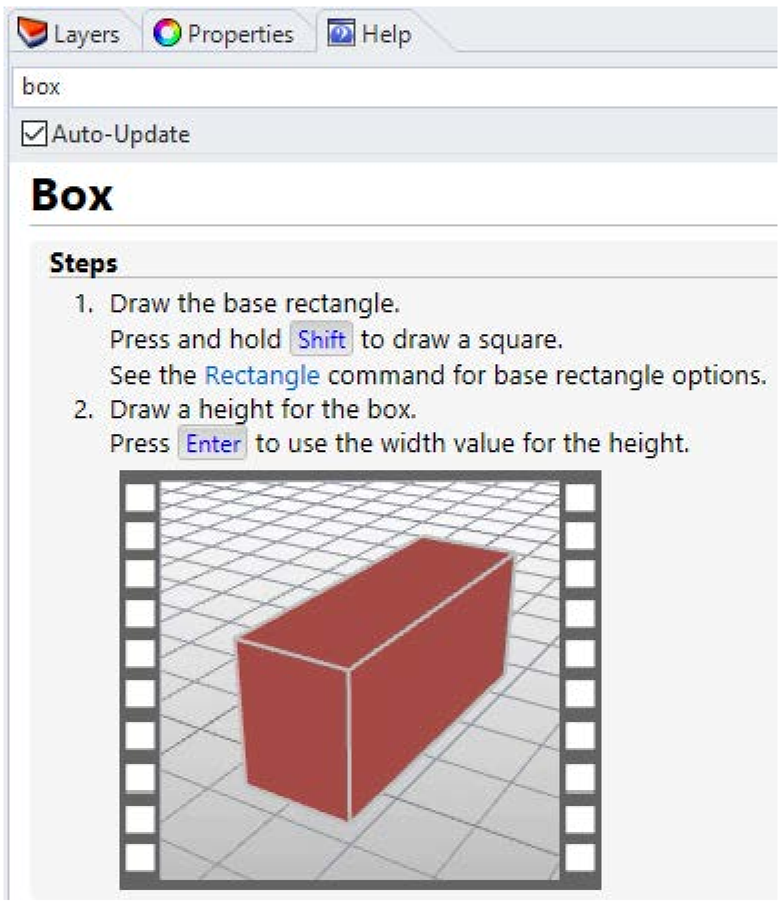
Title Bar	Displays the filename and the Rhino version used.
Menu Bar	Access commands, options, and help through text-based drop-down menus.
Command Prompt	Area to type and run commands, and see options and other information displayed by the command while running or after it exits.
Command History	Displays the most recently used commands and all their prompts.
Tabbed Toolbars	Groups are containers with one or more toolbar, with a tab at the top for each toolbar.
Sidebar	Toolbar to access common commands and options related to the selected tabbed toolbar.
Tabbed Panels	Rhino controls such as layers, properties, materials, lights, display mode, help and more are displayed in panels with tabs. Tabbed panels can be arranged side by side, or vertically.
Layer and Property Panels	One of the important panels that can be tabbed. The other important panel is the properties panel. Most users have at least these two panels visible.
Status Bar	Displays the coordinates of the pointer, the units and the current layer of the model, toggles and other options.
Osnap Toolbars	Used to set object snap settings.
Selection Filter Toolbar	Used to narrow down what geometry can be selected.
Viewport Tabs	Click to make active a viewport. Also, use the "+" to add more viewports and Layouts (paper space).
Standard Viewports	Displays different views of the model within the graphics area. Viewports can show a grid, grid axes, and world axes icon. Any number and combination of viewports can be arranged within the graphic area. The default viewport layout displays four viewports (Top, Front, Right, and Perspective).
Viewport Title	Display the viewport projection. Click the arrow to access viewport settings through the drop-down menu.
Gridlines and Axes	Used to aid modeling. Their visibility, colors and other settings can be customized.
World Axes Icon	Used to aid modeling. It changes when rotating the model.

Description of Rhinoceros interface elements

Resources: User Interface
Video tutorials: User interface for Windows User interface for Mac **Rhino level 1 training:** Use interface, snapping, navigation and display modes

1.1.2 Help

The help files include the full functionality of the modeling application. [***Rhino help***] is accessible online and also from within the application using the ***CommandHelp***. Topics are displayed in a dockable panel. The help offers detailed description of the commands, options and short videos that show the workflow. If you check the ***Auto-Update*** option in the command-line, the help topic updates for the current command. To access the entire Rhino help document in a browser window, go to ***Help*** > ***Help Topics*** menu, or press ***F1***.



Help can be set to display help for the current command

1.2 Viewports and Navigation

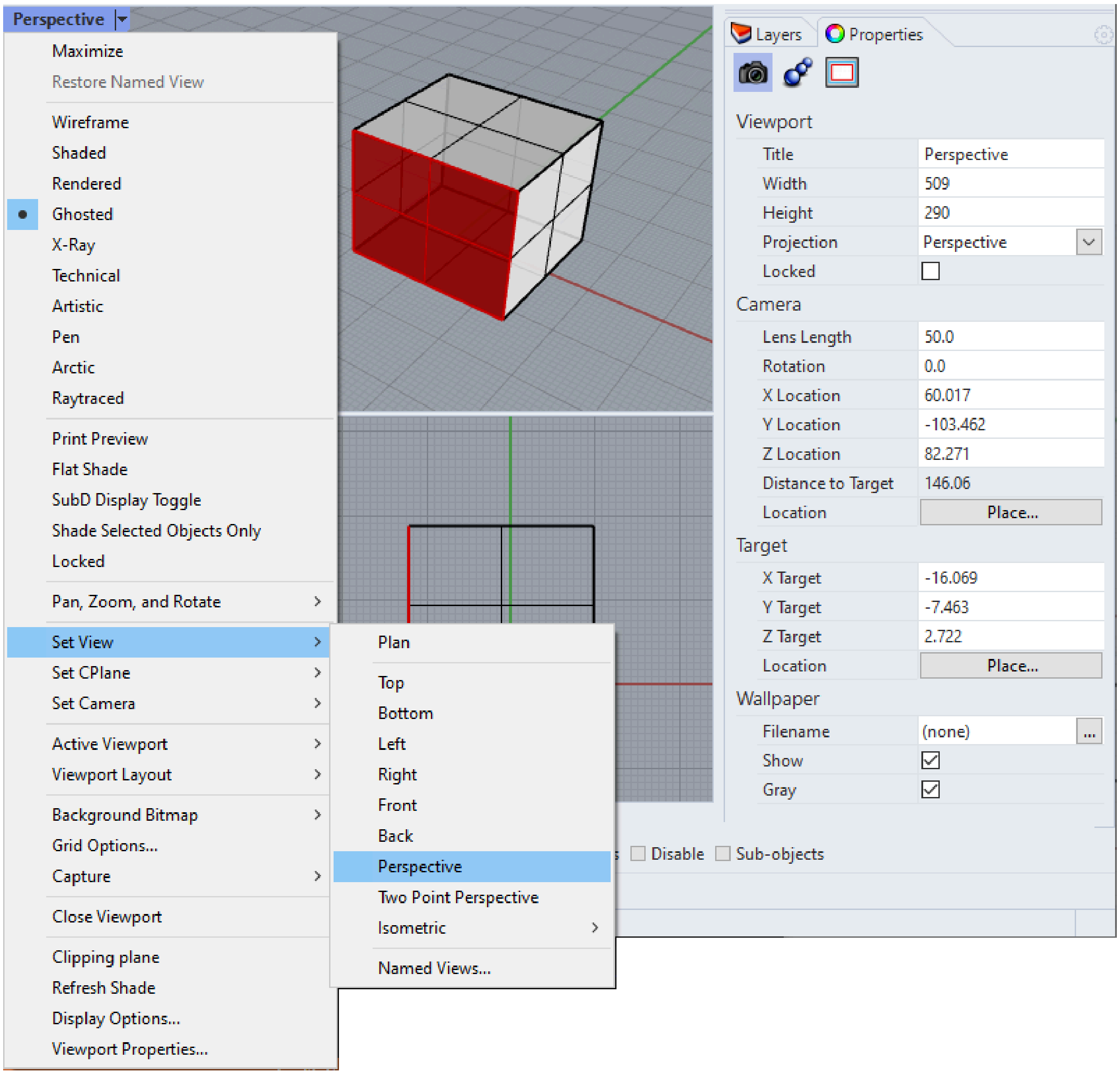
1.2.1 Viewports

The 3D modeling space is contained inside special windows called viewports. You can think of viewports as cameras looking at the same model from different angles. Furthermore, viewports can be set to parallel, isometric or perspective projection. To assist modeling, each viewport includes an origin point, coordinate axes and a grid drawn on a default plane called *construction* *plane*. You can think of a [construction plane]() as the default plane where geometry is drawn, unless coordinates are typed in (relative to world coordinate), or if snap to some other geometry.

Each view in a viewport is seen through a camera lens. The invisible target of the camera is located in the middle of the viewport. You can assign different projection, zoom and camera angles for each of the viewports as needed. To change your view:

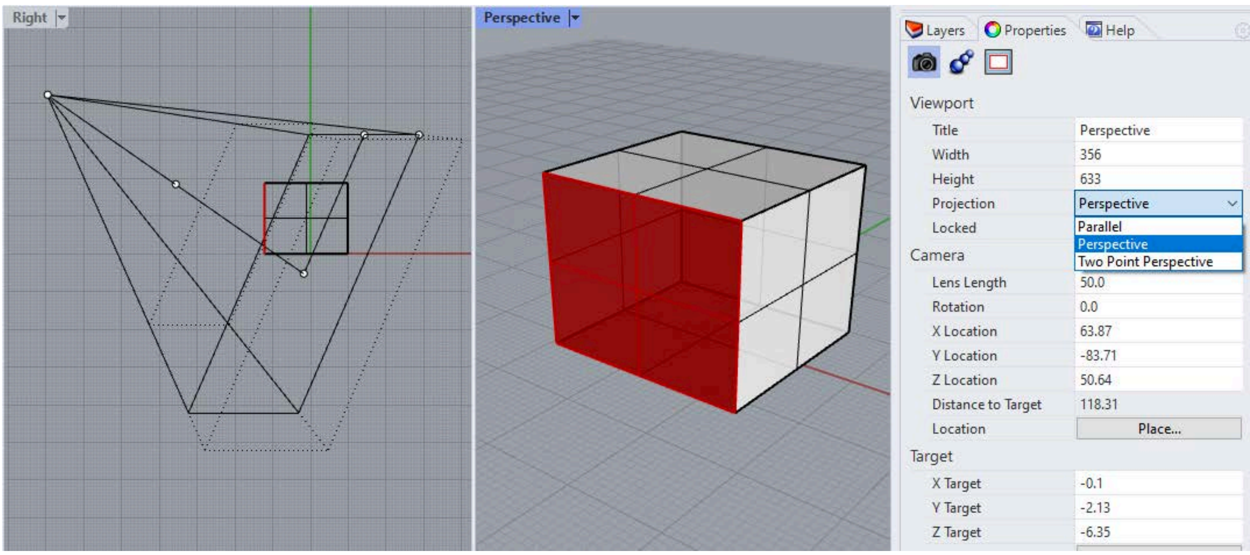
1. Click on the arrow next to the viewport title.
2. Select Set View submenu.
3. Choose your preferred view.

You can also access these steps from the command line using the ***SetView*** command.



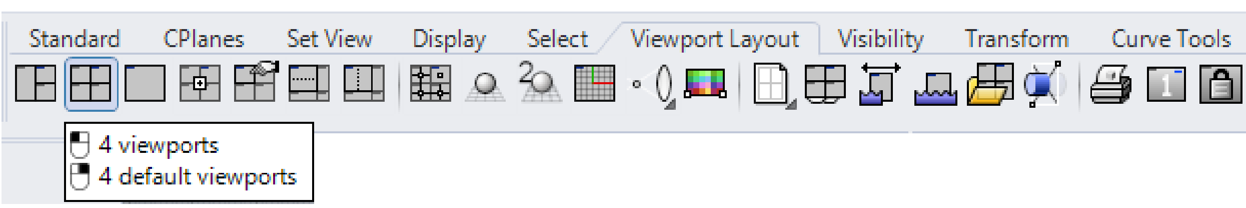
The property panel shows the active viewports camera information and projection. The viewport menu shows the current display mode and give access to viewport commands and options

The property panel shows the viewport settings and camera location. You can adjust the projection of a view directly from the property panel. You can also adjust the camera location and target. You can visualize the camera and edit interactively, you can run the Camera command and toggle the view.



Change view projection. Run the ***Camera*** command and toggle to ***Show*** to show the camera.

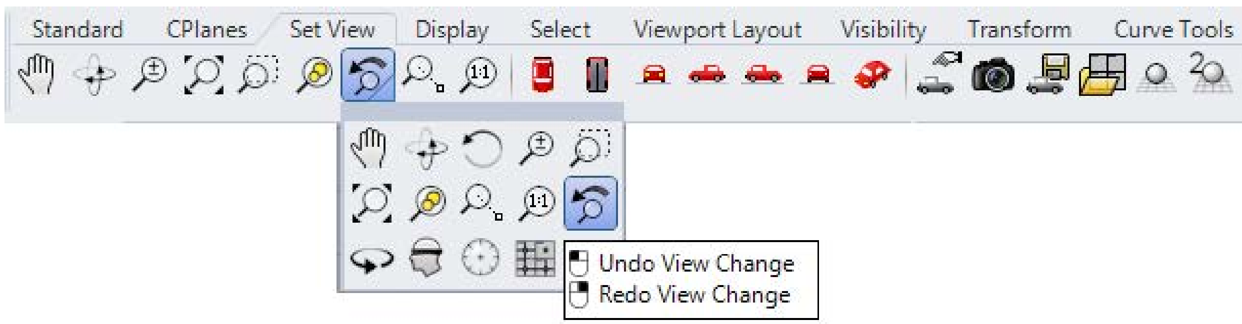
You can customize the viewports and their position to suit your preferences. The position of viewports is adjustable. To move and resize viewports, drag the viewport title or borders. You can create new viewports, rename them, and use predefined viewport configurations. To toggle between a small viewport and one that fills the graphics area, double-click the viewport title. You can display the viewport titles in tabs if you prefer. The highlighted tab designates the active viewport. Tabs make it easy to switch between viewports when using maximized or floating viewports. One unique feature about Rhino is that each of the standard four viewports has a different construction plane (except perspective, which uses the *Top CPlane* by default).



Reset viewport layout to 4 views

With Rhino, you can open an unlimited number of viewports. Each viewport has its own projection, view, construction plane, and grid. If a command is active, a viewport becomes active when you move the mouse over it. If a command is not active, you must click in the viewport to activate it. You can divide your viewport to have multiple viewports with different projections from Viewport Layouts, then split it either horizontally or vertically. You can go back to the standard four views.

There are designated commands to undo and redo view changes. Click in a viewport, press your ***Home*** or ***End*** keys on your keyboard to undo and redo view changes, or click on undo-redo view changes under the *Set View* tab in the toolbars.



View undo and redo

1.2.2 Navigation

Navigating modeling space refers to the ability to reach certain parts of your model and be able to view them up close or from far at any angle and projection. Terms such as “zoom”, “pan”, “parallel projection” are standard in all modeling software. The computer mouse is usually used to navigate models along with specialized commands or tools to help quickly get around. Screen gestures and virtual reality tools allow navigating touch screens and VR.

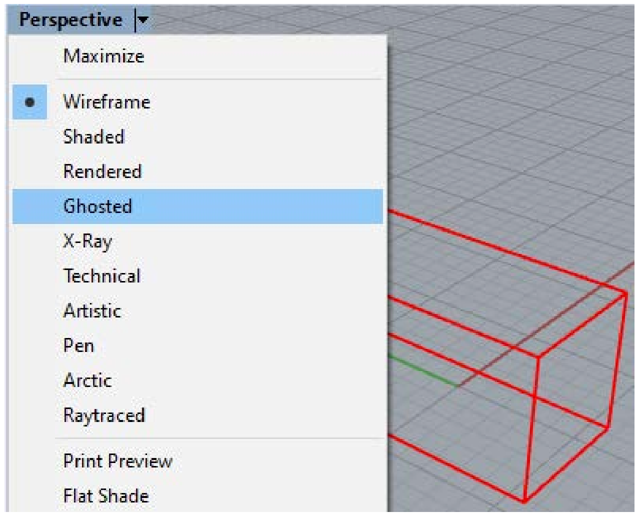
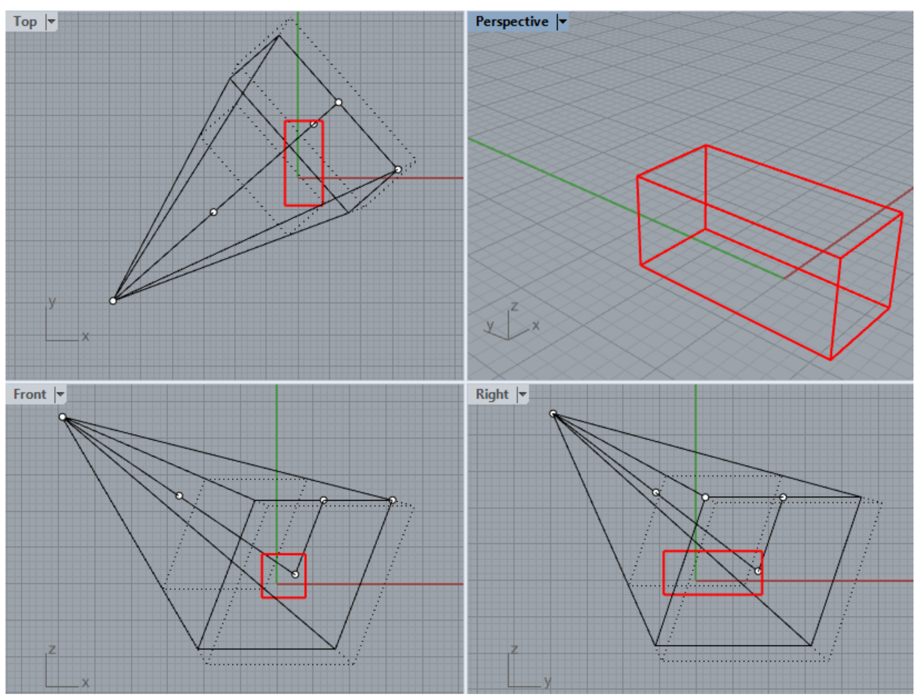
View navigation includes panning, zooming and orbiting. Panning means shifting the view without changing the camera angle. Rhino Pan command supports panning at any projection. The simplest way to pan is to drag the mouse with the right mouse button held down (also need to hold down the Shift if you are in perspective). To zoom in and out, use the mouse wheel, or hold down the Ctrl key and drag your mouse up and down with the right mouse button held down. Orbiting is only active in perspective views. With the right mouse button down, moving the mouse rotates the view around the center of the view (or the target of the camera). You can pan, zoom or orbit your view in the middle of any command to see precisely where you want to select or snap.

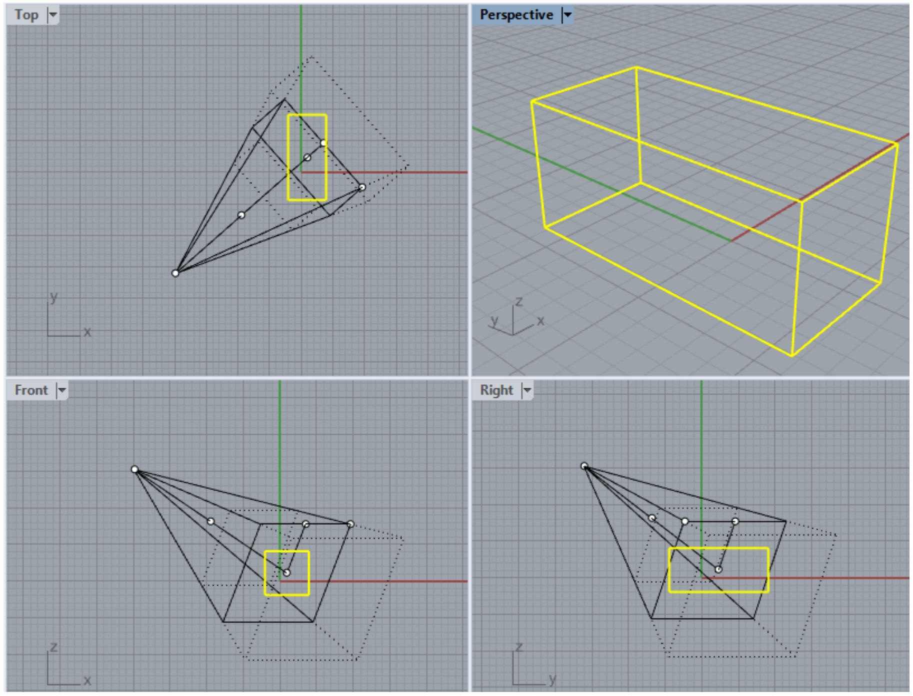
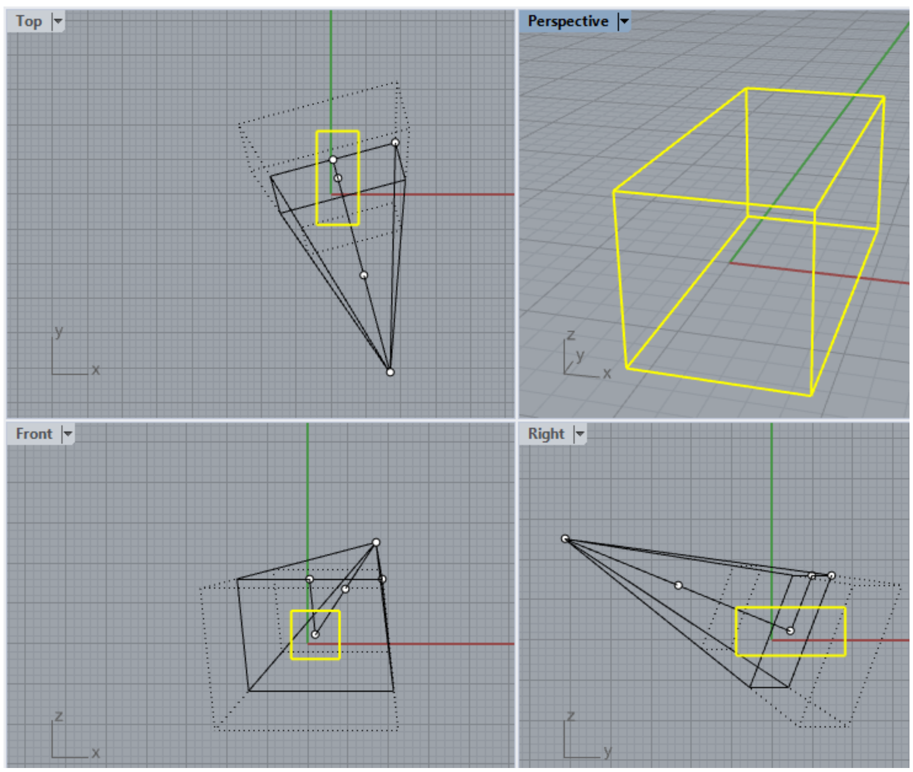
Function	**Mouse action**
Repeat last command or end a current command	Right mouse button
Selects objects	Left mouse button
Customizable Pop-up menu	Press wheel
Orbit the perspective viewport and pan parallel viewports	Drag with right mouse button down
Pan perspective viewport	Shift + Drag Right mouse button
Zoom in and out	Roll the wheel

Zoom in and out	Ctrl + Right mouse button

Quick reference to Rhino mouse functions

1.2.3 Viewport navigation tutorial

Change the wireframe view to Ghosted mode, then zoom so that the target of the camera is located at the center of the selected object, then orbit around the selected object.	
Change to Ghost view using the viewport drop down menu	
Use the ***Camera*** command to show the perspective view camera location and target.	

Select the box, then use the ***Zoom*** command with ***Selected*** option to center the camera target to the box center *(**ZS** as keyboard shortcut)*. Observe how the camera location changes	
Use right-mouse-down then move the mouse to orbit and observe how the camera changes location without changing the target. You can also click on the camera points to change location, target and tilt of the perspective view. Run the ***Camera*** command again to hide.	

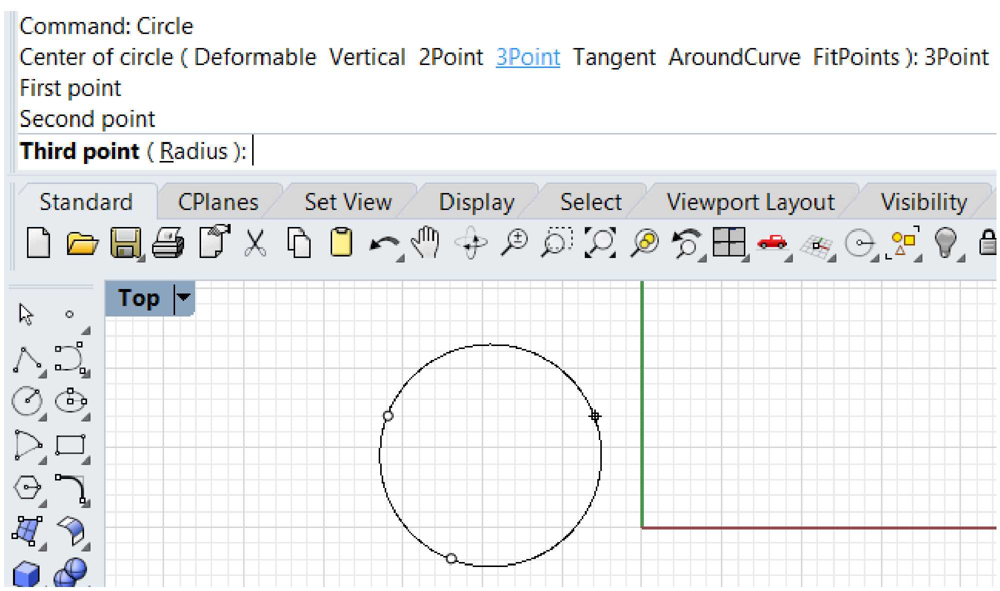
1.3 Access to commands

[View video tutorial...](#)

Commands are actions or functions that perform specific operations. For example, ***Circle*** is a command that helps create and add a circle to the 3D model. Rhino allows invoking the same commands using many different ways such as command-line, menus, toolbars, or interactively through widgets, mouse and keyboard events. For more advanced access, Rhino supports creating custom commands through [macros and scripting]. The different ways to access Rhino commands mean that users have the flexibility to choose the method they are most comfortable with to help increase their productivity. Each command has a name and most commands allow setting various options.

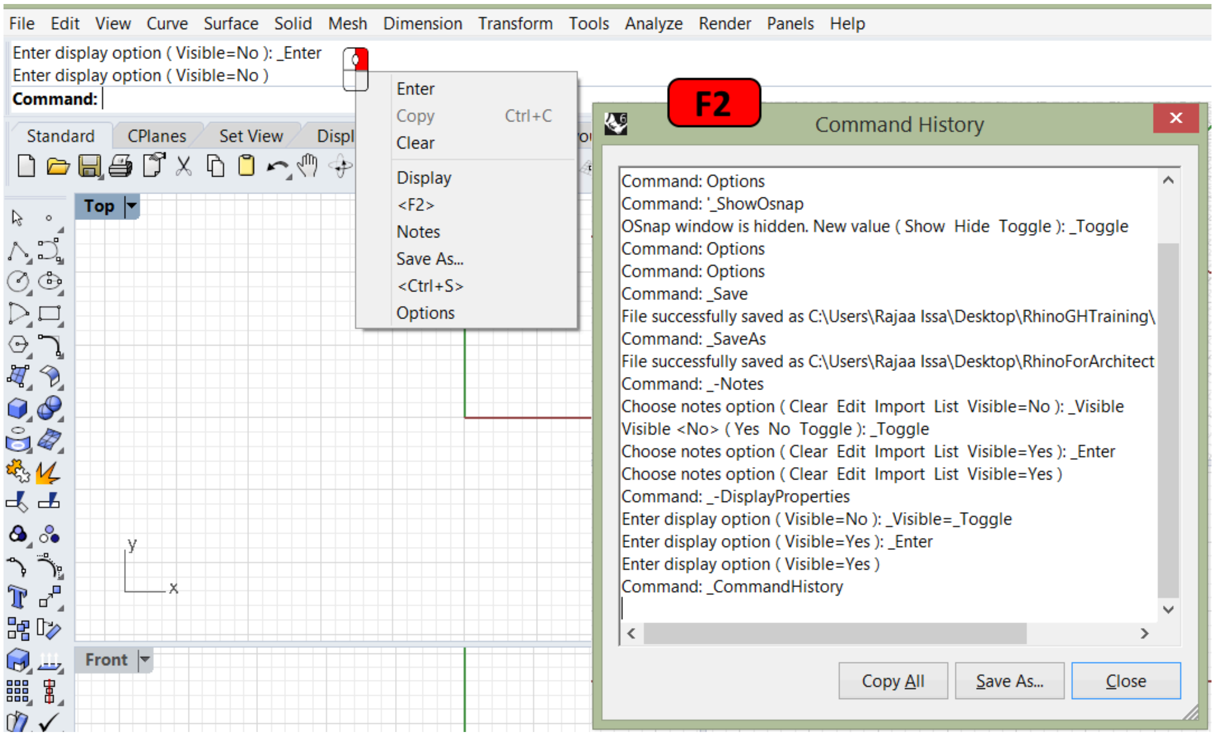
1.3.1 Command-line

This is a widely used method to access commands in Rhino especially among users familiar with the commands they commonly use. Commands are run by typing their names (or the first few letters of it) in the *Command Prompt*. Typing in the command prompt shows a list of all commands starting with the same letters. Frequently used commands show on top of the auto-complete list. Right-mouse-click on the command line to view recently used commands. You can also press *Enter* to run the last command (or right-mouse-click or space key), and press *Esc* to cancel a running command. When inside a command, the command prompt area shows the command instructions and options. Many Rhino commands do not support dialogs and hence command options can only be set through the command prompt area.



The command prompt shows options and record steps in the command history area

The [command history] area shows recent commands and options used. You can press ***F2*** to view the command history and all the options used. Right-mouse-click on the Command prompt area shows a menu with all recent commands used that you can select directly from to repeat.



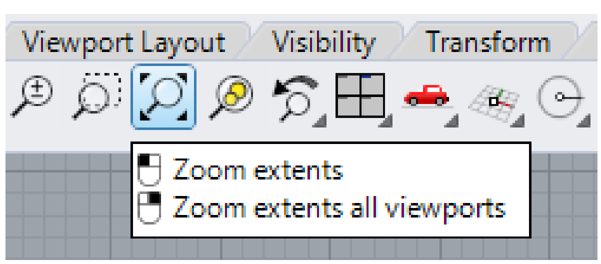
Right-click on the command area to access recent commands. Click **F2** to navigate the last 500.

1.3.2 Toolbars

Rhino toolbars contain buttons that provide shortcuts to commands with various options. You can float a toolbar anywhere on the screen, or dock it at the edge of the graphics area. Rhino starts up with the *Standard* toolbar group docked above the graphics area and the Main toolbar as the sidebar on the left.

Tooltips

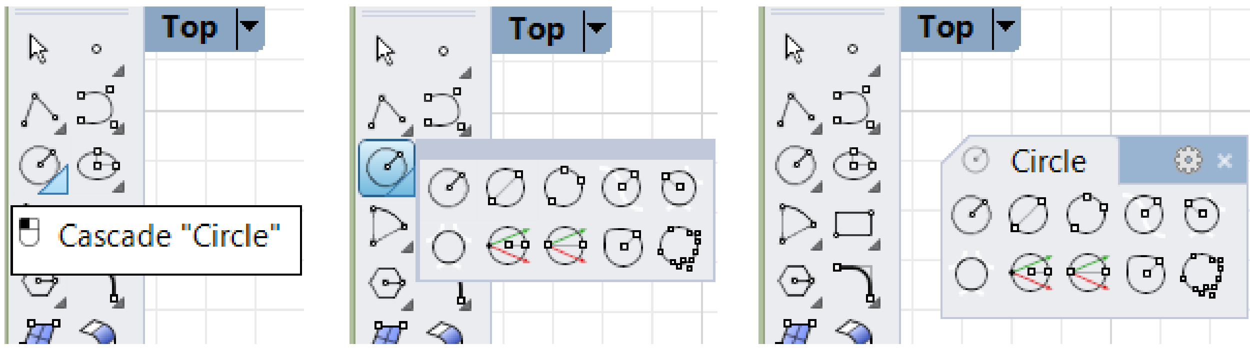
Tooltips tell what each button does. To access the tooltip, move your pointer over a button without clicking it and a small tag with the name of the command appears. In Rhino, buttons can be set to execute multiple commands and set specific options. The tooltips also indicate which buttons have dual functions as in the following example.



Tooltips give hints about what the button command does

Cascading toolbars

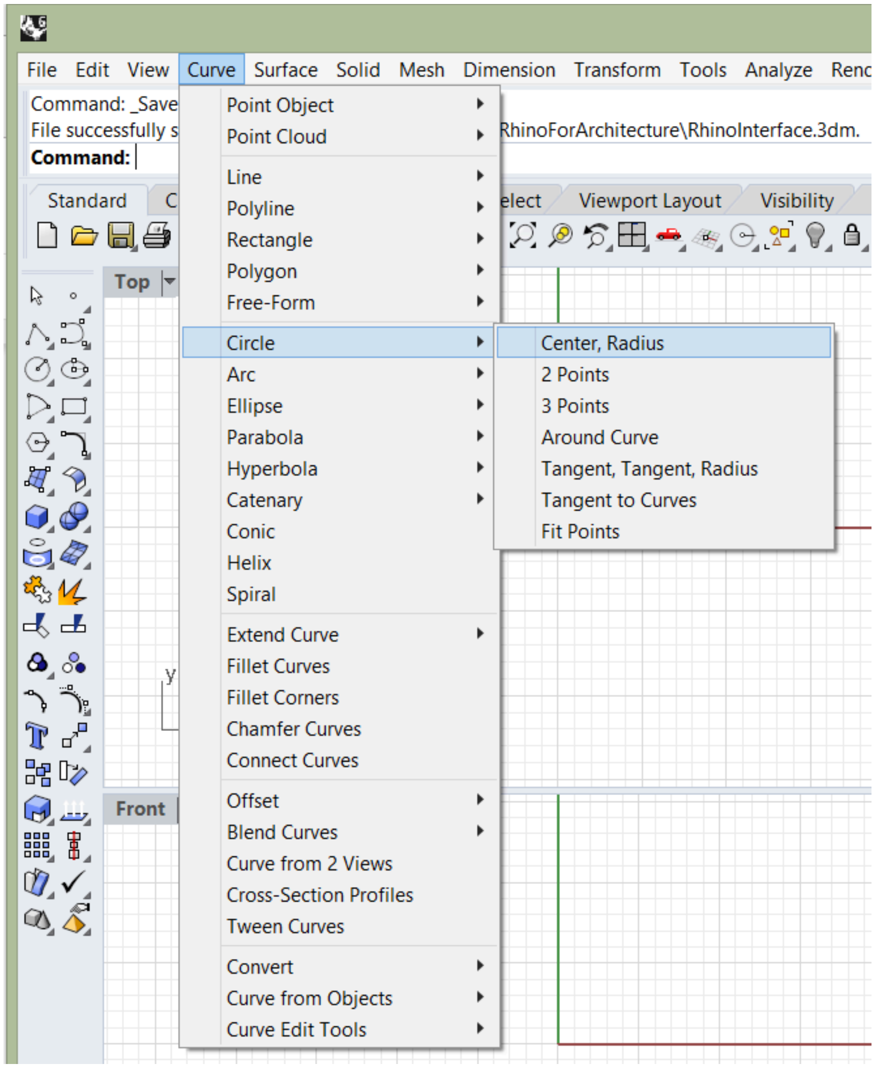
A button on a toolbar may include other buttons in a cascading toolbar. Usually, the cascading toolbar contains variations of the base command. After you select a button on the cascading toolbar, the toolbar disappears. Buttons with cascading toolbars are marked with a small black triangle in the lower right corner. To open the cascading toolbar, hover over the black triangle and left-mouse-click. You can select a button from the cascade toolbar, or peel out by clicking on the top margin of the toolbar.



Copy of the cascade toolbar can be peeled off

1.3.3 Menus

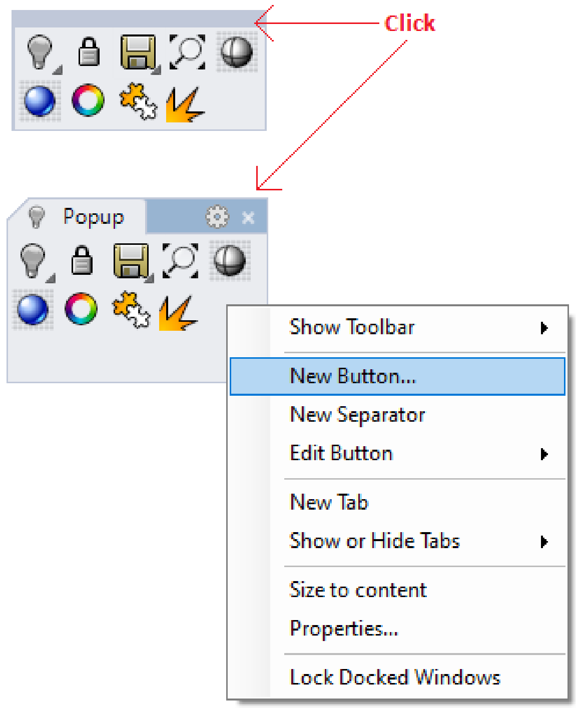
You can find most of the Rhino commands in the menu bar. Each menu groups similar commands together. There are many submenus within each drop-down menu to further group similar functionality. For example the *Curve* menu includes curve creation and editing commands and under Circle to see all the different ways to create a circle. Third party plug-ins sometimes add their own menus to the menu bar.



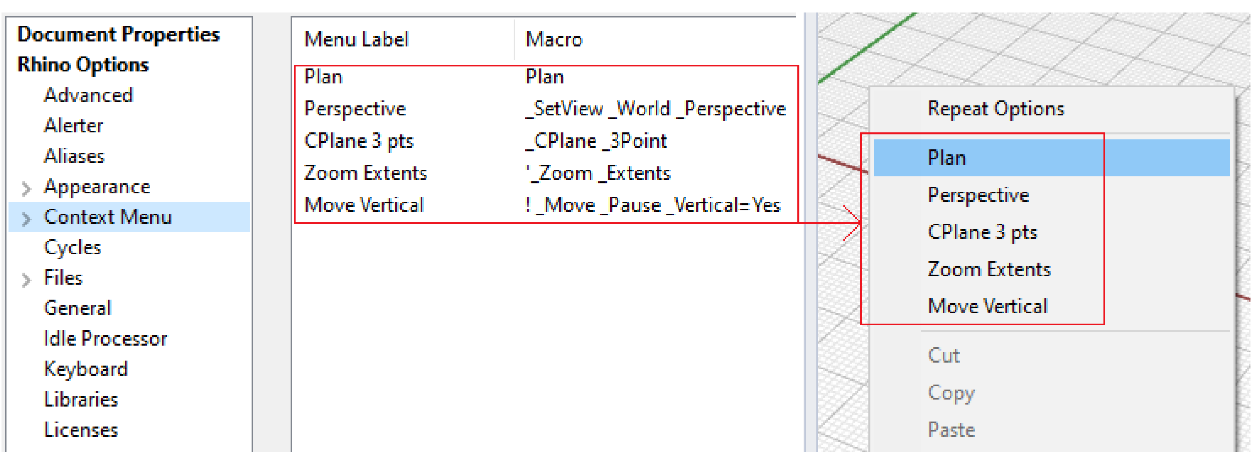
Curve menu and submenus include all Curve commands

1.3.4 Custom access

There are a few ways to customize your access to Rhino commands. The main objective is to increase productivity for repeated commands and sequence of commands.
The pop-up menu is accessible when you press the wheel (or middle mouse button). You can think of it as an on demand toolbar that you can add or remove buttons from or create your own custom buttons with custom macros. For details about how to customize toolbars check the toolbars help topics.

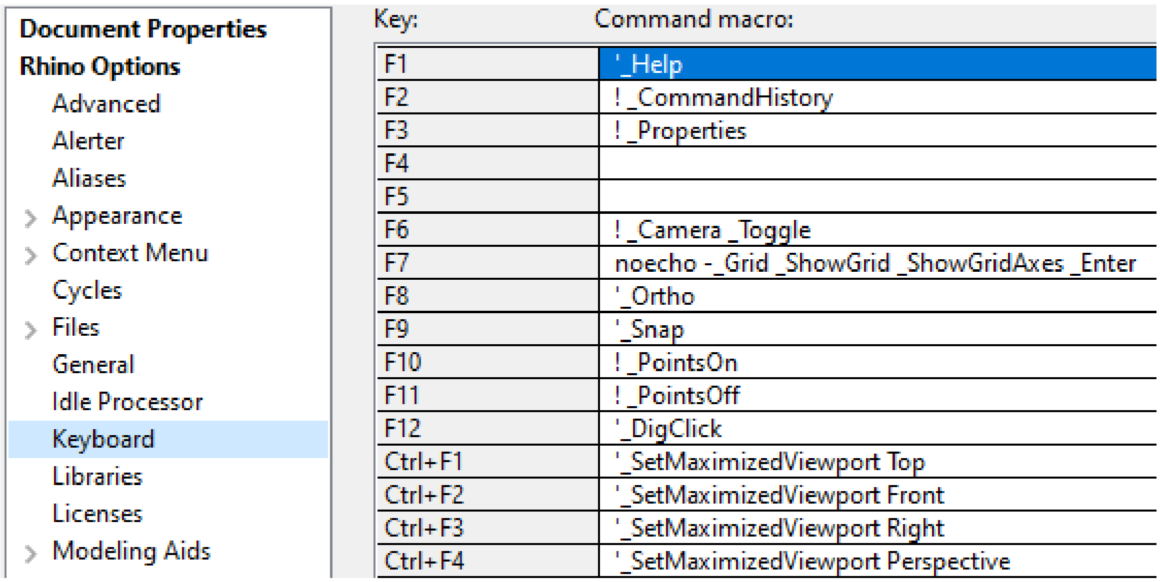


The context menu shows when you press the right mouse button for a second then release. You can add to it additional commands that would be readily accessible with mouse control.

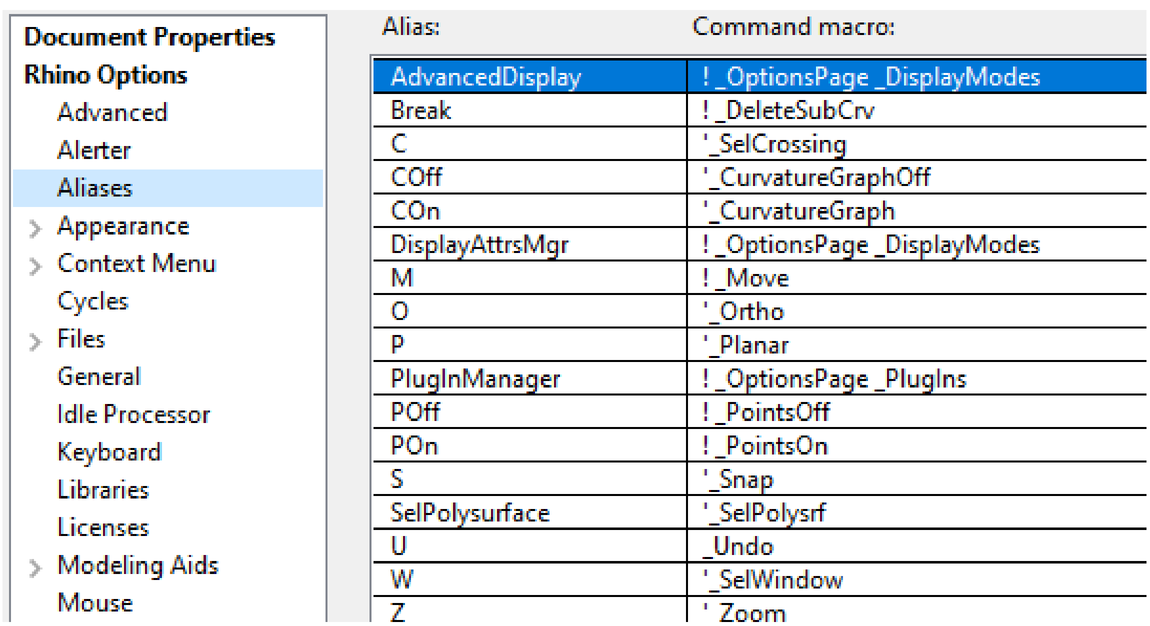


Right mouse down for a second, then release to show the context menu in Rhino

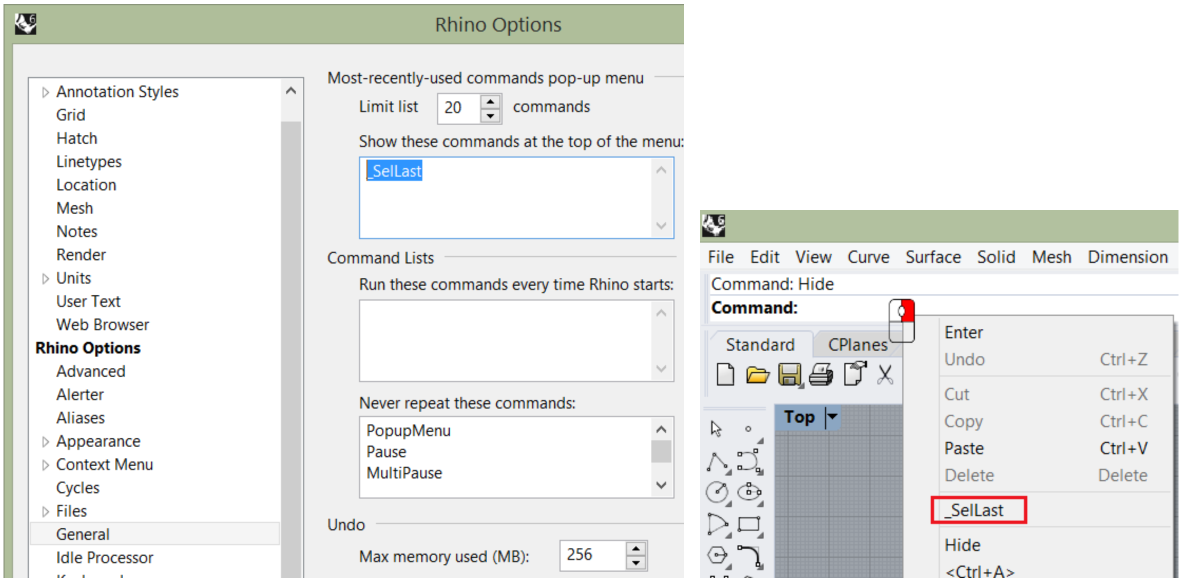
You can set up a custom keyboard access. Rhino comes with a few preset keys that can be edited by the user. For example ***F1*** opens the ***Help***, ***Ctrl+S*** to ***Save*** and Ctrl+Z to ***Undo***. You can find the full list under the ***Rhino Options*** under the ***Keyboard*** section (open by running the ***Options*** command).



Aliases allows setting custom command names to run existing commands and macros. Macros allow setting one or more commands with specific options to run all at once. Rhino has few preset aliases such as ***ZE*** for ***Zoom*** with ***Extents*** option and ***M*** to ***Move***. You can change and add to the list of aliases, save them and share with other users.

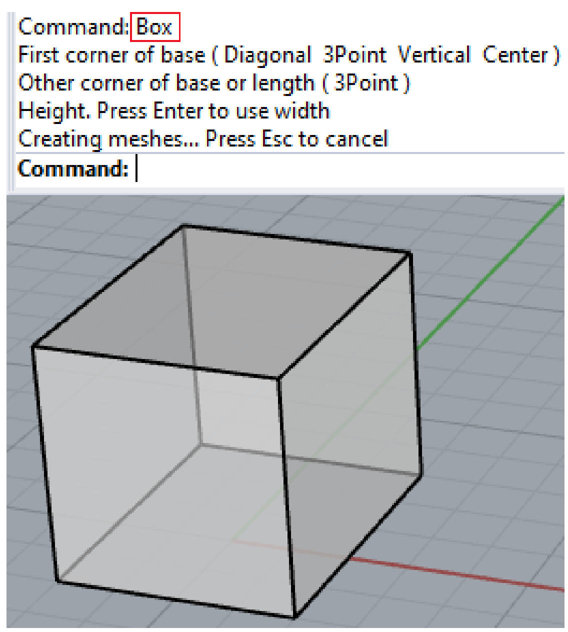


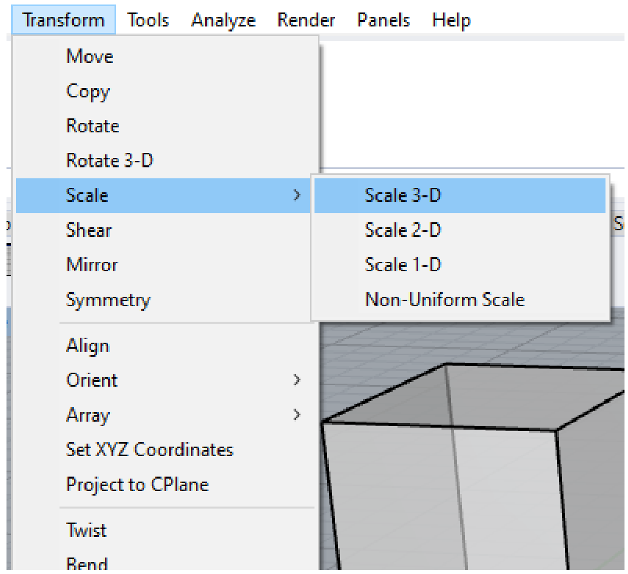
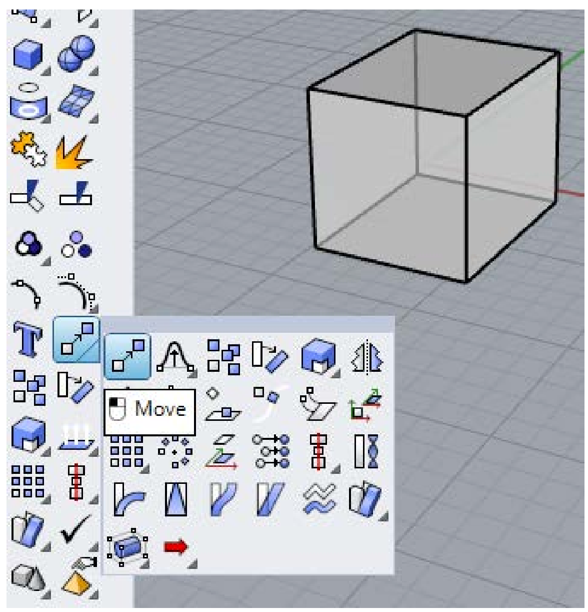
Most recently used commands are accessible when you right mouse click on the command area. You can add your choice of command on top of the list by setting the ***Rhino options > General***, then enter your command macro in the top field as in the following image.



Set startup command menu

1.3.5 Commands access tutorial

Exercise: Create a box then scale and move using 3 different ways to access commands in Rhino	
1- Enter ***Box*** in the Command-line and follow prompts	

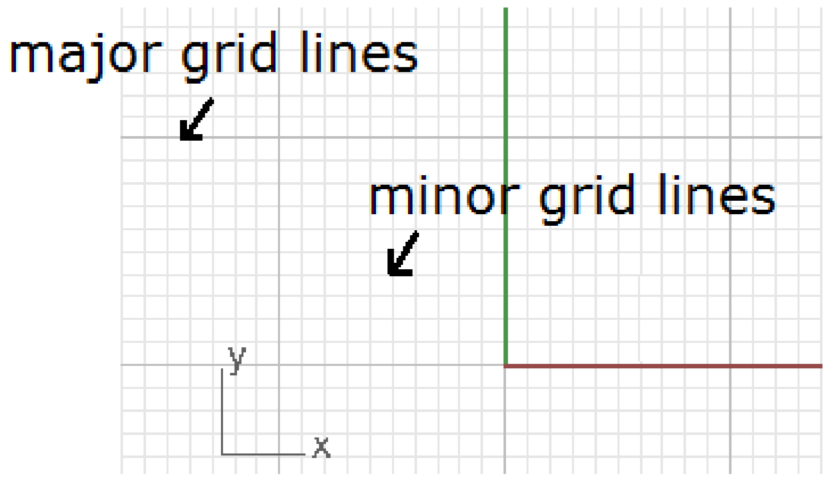
2- Select the ***Scale*** command from the ***Transform*** drop-down menu to run the Scale command. Follow prompts for complete.	
3- Click the ***Move*** button in the *Standard* side toolbar	

1.4 Modeling aids

Interactive digital 3D modeling can be challenging without guides, constraints and ways to organize and manage geometry and data. Modeling aids help locate and manage 3D geometry. Modeling guides include grids, coordinate axes and smart tracking mechanisms. Constraints include ortho, geometry constraints, filters, selection constraints, gumball, and construction planes. Layers help organize geometry and assign some attributes. Geometry in Rhino also has attributes that can be managed using the properties panel. Document and application preferences can be set in the Rhino document properties and the rhino options. We will cover most used parts in the following sections and the rest of the document, and you can reference the full details in the Rhino help file.

1.4.1 Grid and Grid Snap

Grids are a handy visual reference of the orientation and the scale of the modeling space. You show, hide, change the number of grid lines, spacing and intervals in which you can snap to.

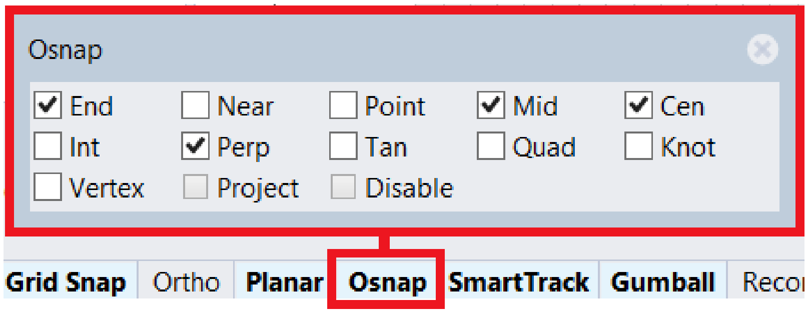


Grids align with construction plane and has major and minor lines

Grid Snap helps snap on grid intersections. You can also toggle Grid Snap on and off by pressing F9 or typing the letter S and pressing *Enter*. Pressing F7 hides or shows a reference grid in the current viewport of the graphics screen at the construction plane.

1.4.2 Osnap

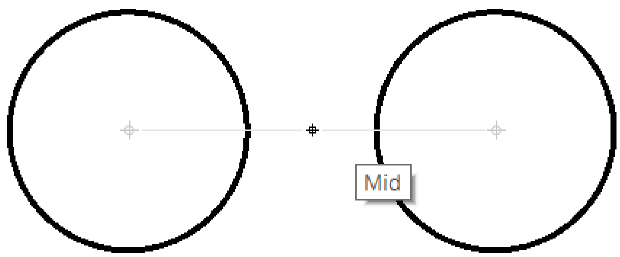
Object snaps constrain the marker to an exact location on an object such as the end of a line or the center of a circle. These are very important tools to help model accurately and quickly. You can customize Osnap to follow one or more constraints. To quickly use only one constraint, right mouse click on it to disable the others. Another right mouse click on it will restore the previous Osnap state. One special Osnap is Project. If checked, your geometry will be projected to the view construction plane even if you snap outside of it.



Object snaps

1.4.3 Smart Track

Smart Tracking uses temporary reference lines and points that are drawn in the Rhino viewport using implicit relationships among various 3D points, other geometry in space, and the coordinate axes' directions. Smart Track uses OSnap settings to create the reference lines.

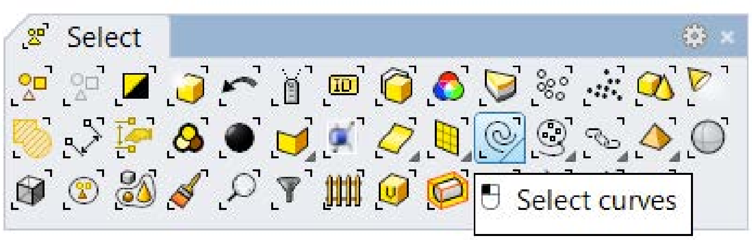


Smart Track to create temporary guides and snap on these guides

It is also possible to set up persistent modeling guides that appear when selecting points. Use AddGuide and RemoveGuide to add and remove these guides.

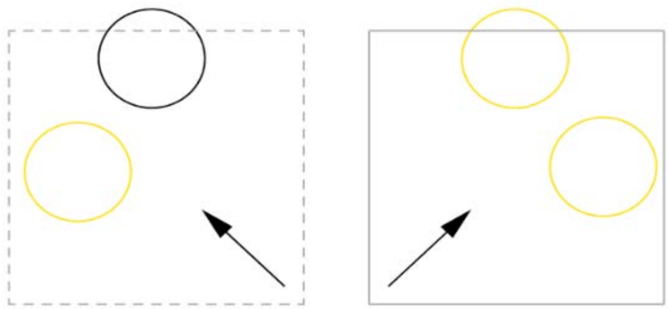
1.4.4 Selection

Rhino supports different ways to select geometry. Selection commands allow selecting all objects in the file that are of a certain type or share certain attributes.



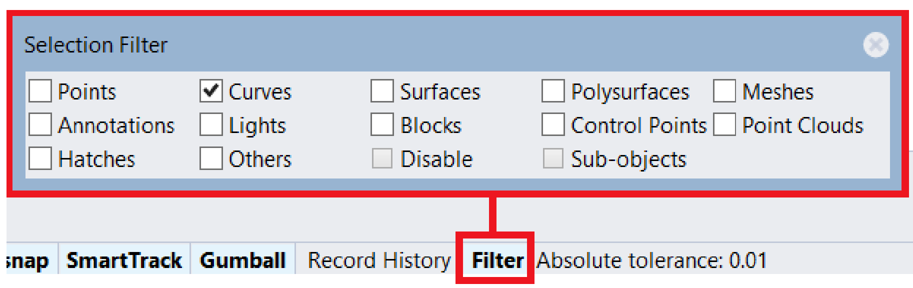
Selection tools

A cross or box window selection is used to select geometry by dragging the mouse (with left button down) around a specific area inside a viewport.



Left: cross window selection from right to left selects objects completely within the window. Right: box window selection from left to right selects all objects wholly or partially within the window

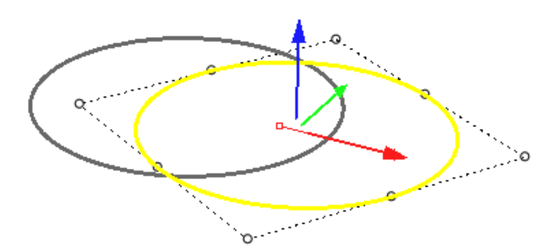
The selection filter allows isolating certain object types. The window selection would then select only the types permitted by the selection filter. For example, if only “Curves” is checked, then no other object type is selected even if they fall within the window.

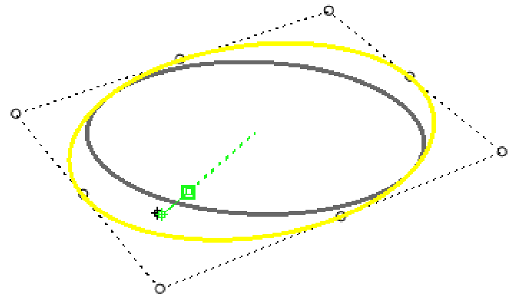
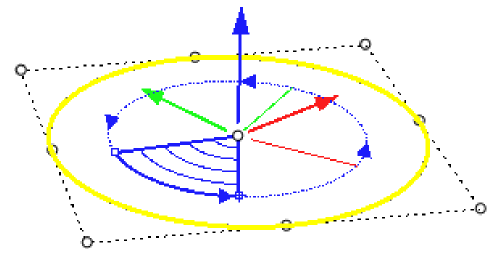
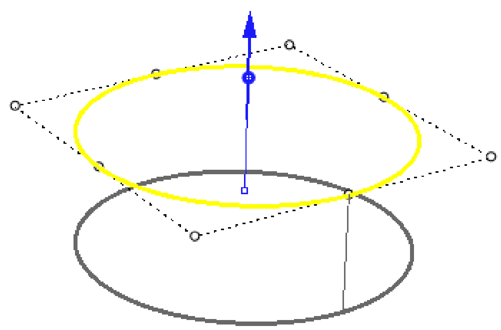
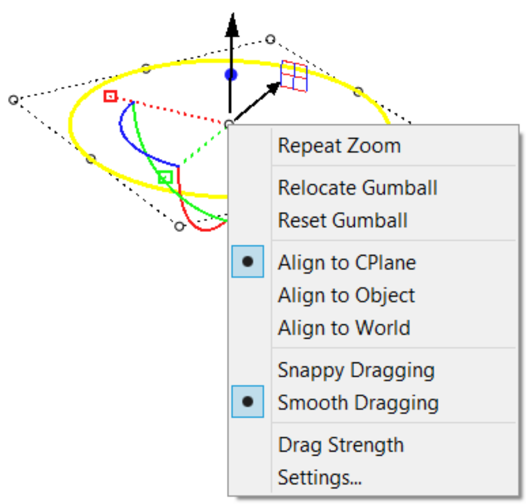


Selection filter to isolate types for selection

1.4.5 Gumball

Gumball displays a widget on a selected object which is used to facilitate direct editing. The Gumball provides move, scale, and rotate transformations around the Gumball origin.

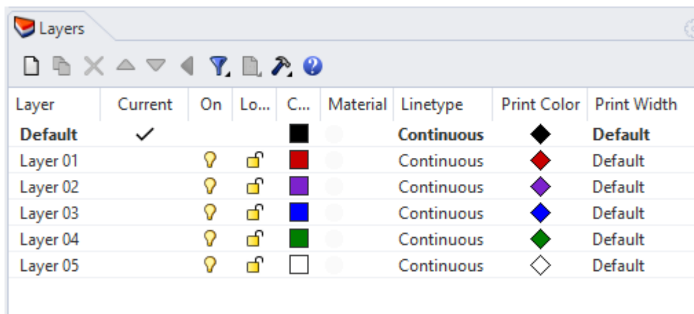
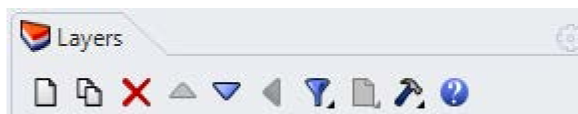
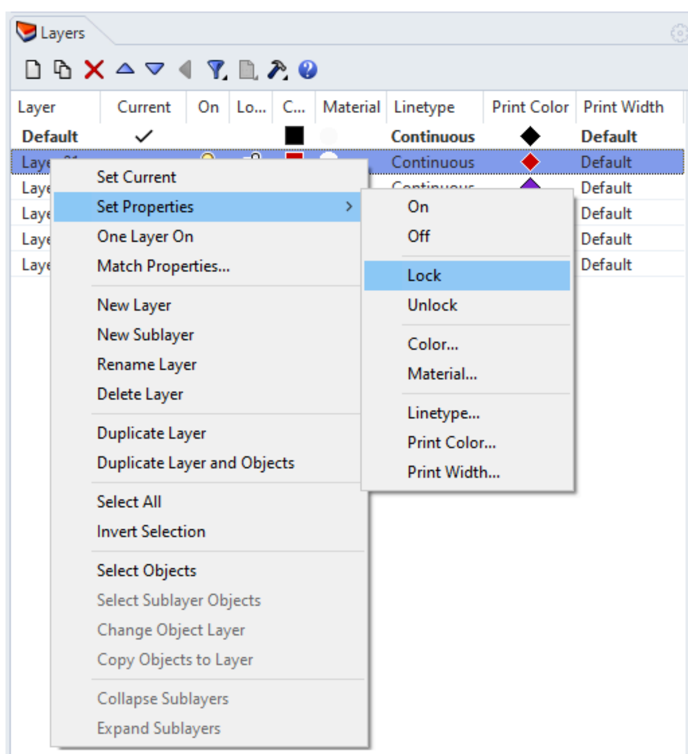
Dragging Gumball arrows move the object in the arrow direction. Note that pressing the Alt key makes a copy of the object.	

<table>
<tr><td>Moving the Gumball handles scales the object in the handle direction. If dragged with the Shift key down, the scale becomes uniform in both directions</td><td></td></tr>
<tr><td>Dragging Gumball arcs rotate the object.</td><td></td></tr>
<tr><td>Dragging the arrow handle extrudes the object. With Shift, it extrudes in both directions.</td><td></td></tr>
<tr><td>Other Gumball functions can be accessed with a right mouse click at the center handle.</td><td>
</td></tr>
</table>

Gumball functions

1.4.6 Layers

Layers are important to organize and access objects in the document, and are used by most modeling software. You can assign meaningful names to layers and set common attributes such as colors and materials. You can quickly hide, lock or select the objects in any layer, and move objects between layers. For details about the different parts of Rhino layers and related commands, please refer to the [layers section in the help](#).

Default layers with new document using one of the Rhino templates: - Multiple layers are created with different colors. - *Default* is designated as the current layer (new objects are placed in the current layer). - No material is assigned.	
Layer table functions are used to manage layers (add new layer or sublayer, delete, move within the table, filter and other functions.	
Right mouse click on any layer shows all commands related to the selected layer. For example you can set as the current layer, change any property, create a sublayer or select all the objects within that later.	

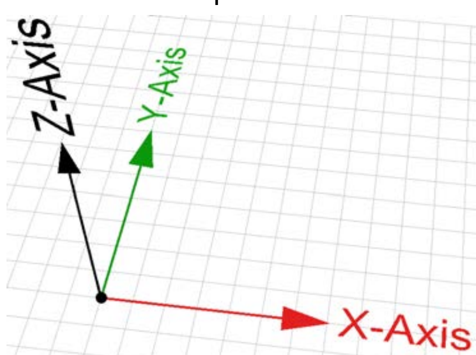
Layers table in Rhino

Resources: Modeling aids
Rhino level 1 training: [Modeling helpers](#) **Video tutorial:** Gumball: https://vimeo.com/260472052

1.5 Coordinate system

1.5.1 Overview

The coordinate system is used to describe the location of points in space. The most common one for architectural applications is the Cartesian coordinates. Rhino uses two Cartesian coordinate systems: construction plane coordinates and world coordinates. World coordinates are fixed in space while construction plane coordinates are defined for each viewport. Rhino also follows the right-hand-rule to define the relative directions of X, Y and Z axes directions. That means when the right hand thumb points in the positive x-direction and the forefinger points in the positive y-direction, then the middle finger points in the positive z-direction. Also if the thumb is pointing in the Z direction, rotating towards fingertips points to the positive angle direction. There are three World planes (XY, XZ, and YZ) that meet at the origin point. The location of any point in space is described with three ordered numbers (tuple). The World origin is located at (0,0,0). Points in Rhino can be defined in either absolute or relative coordinates. You can also use polar notation to describe a point.



Rhino uses left-hand Cartesian 3D coordinate system

The following table includes examples of specifying points in Rhino relative to the construction plane and world coordinates.

Construction plane coordinates	Enter x,y,z values to place points relative to the current construction plane. You may omit z and y values (they are set to 0 in this case). Examples: 0 = (0, 0, 0) in CPlane coordinates 1,3.5 = (1, 3.5, 0) in CPlane coordinates 1,3.5,6 = (1,3.5,6) in CPlane coordinates

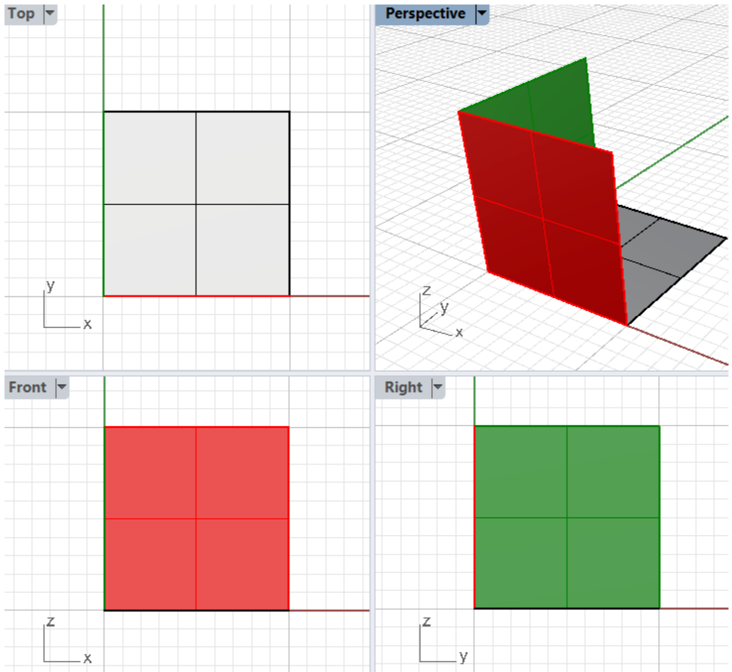
World coordinates	When the construction plane is different from the World planes, you need to specify that your point is relative to World coordinates by preceding the x,y,z by “w”. Examples: w0 = (0, 0, 0) in World coordinates w1,3.5 = (1, 3.5, 0) in World coordinates w1,3.5,6 = (1,3.5,6) in World coordinates
Polar coordinates	Use distance, angle and z value (from CPlane origin). Examples: 17<45 (radius<rotation angle) 17<45,8 (radius<rotation angle,z)
Relative coordinates	This helps locate a point relative to the last point used. Suppose last point used was (1,1,0): **@3,4** = 4,5,0 (move 3 units in the x direction and 4 units in the y direction from the previous point) Alternatively, use “R” or “r” instead of “@” to express relative coordinates: **R3,4** (or **r3,4**) You can also use relative polar coordinates: **r4<45** or **@4<45**

Coordinate notation in Rhino

1.5.2 Construction planes

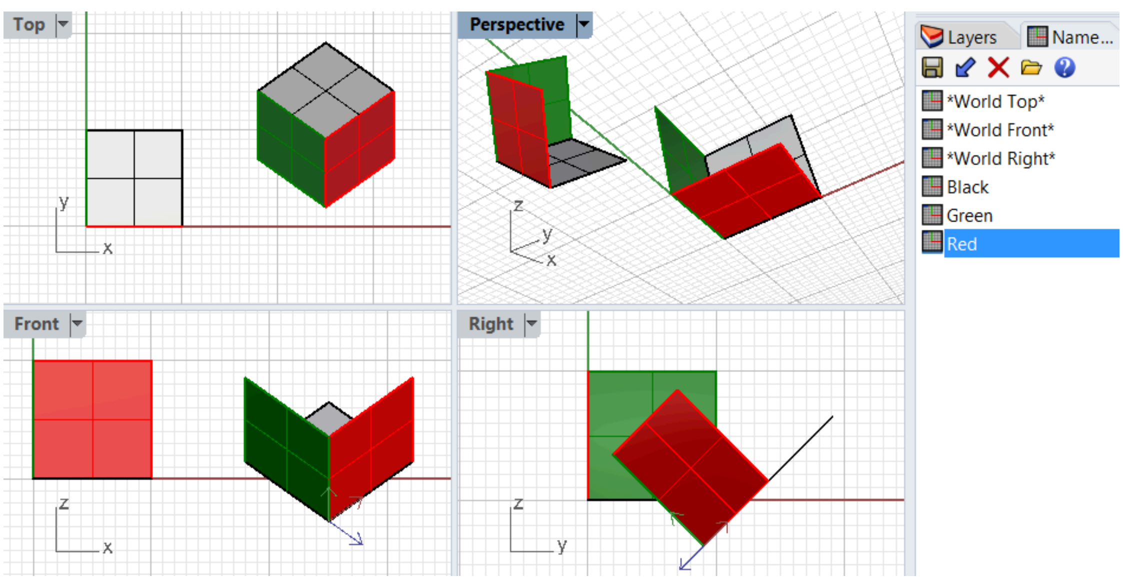
Construction planes (CPlane) are used in modeling software to orient and guide modeling. They align to the World origin and direction by default but can change to be relevant to the model. Creating moving and viewing geometry in Rhino is highly influenced by the orientation of the construction planes. For example, points you create always land on the construction plane unless you use coordinate input (key in as text), use elevator mode, or object snaps.

Rhino default parallel views have their own construction plane that is parallel to the view itself but they share the World origin. Perspective view uses the *World Top* construction plane.



Four Rhino view construction planes

Rhino supplies a rich set of commands to realign the construction plane, and synchronize other viewports to follow if needed. If your model has specific directions that you need to use often, then you should save in the named construction plane table. This way you can reference quickly and not have to recreate every time you need them.



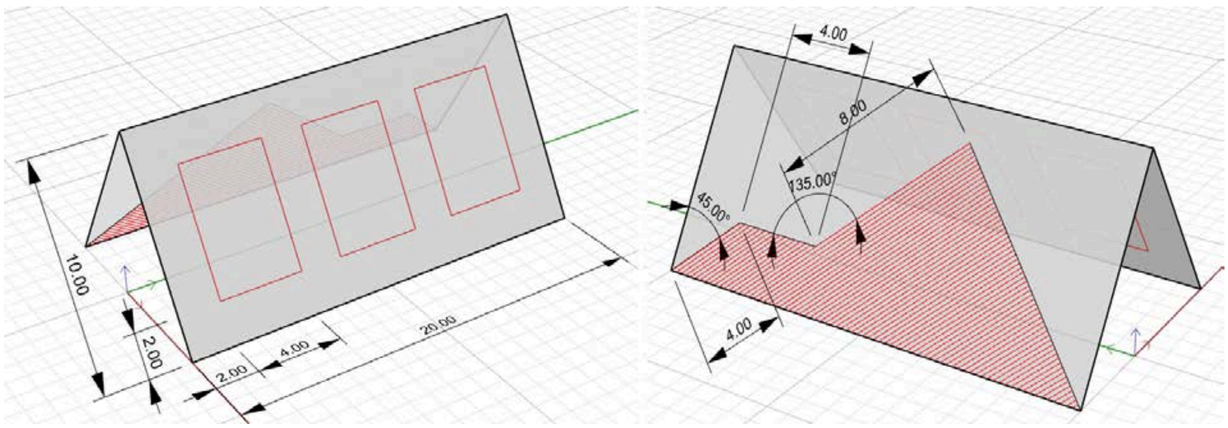
Save construction planes in Perspective view only

Resources: Accurate modeling

Rhino level 1 training:
[Precision modeling]
[Drawing Lines and Polylines in Rhino 6 using absolute or relative coordinates]

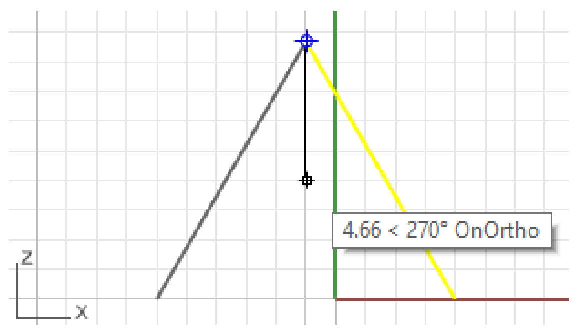
Rhino Help:
[Accurate modeling]

Exercise: Accurate modeling and construction planes



Solution Steps:
Create the geometry:

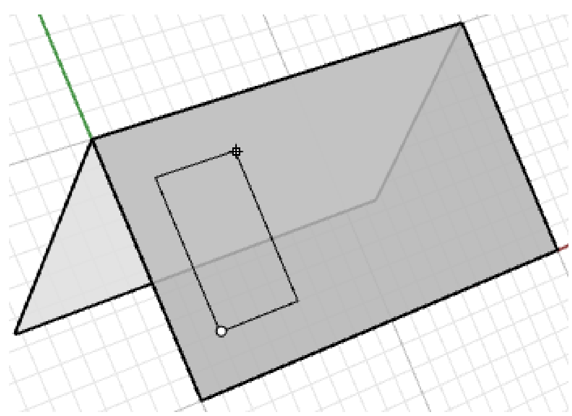
- In ***Front*** viewport (xz plane) draw ***Line*** from any point on the x-axis to create 10 unit length line using relative coordinate at angle 60: r10<60
- Use ***Mirror*** to create the other line
- ***Join*** the 2 lines
- ***ExtrudeCrv*** by 20 units along the world y-axis



Draw the three rectangles on one side

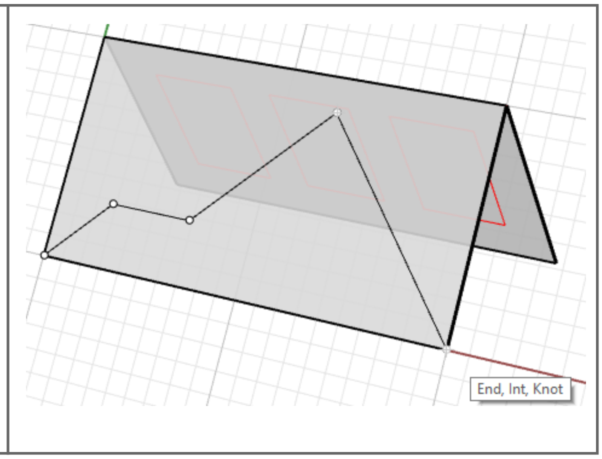
- Align the CPlane to the side: ***CPlane 3Point***
- ***Rectangle***. From: 2,2, To: r4,6
- ***Copy*** to create the other 2 rectangles.
  From: 0 (since the new CPlane origin is located at the lower corner of the side)
  To: r6,0,
  To: r12,0.
- ***Select*** the rectangles and change their display color in the ***Properties*** panel to be red.

Note:
Dimensions are created within one plane, so they should be added while in the CPlane that aligns with the side.



Draw hatch border on the other side

- ***CPlane 3Point*** to align the CPlane
- Polyline. Specify the points using accurate coordinates as in the following:
  From: 0 (since the new CPlane origin is located at the lower corner of the side)
  To: r4<45 (or @4<45 using “@’ instead of “r”)
  To: r4<0
  To: r8<45
  To: snap to other lower end of the side
  To close, select the ***Close*** option
- ***Hatch*** to fill the boundary
- Select the hatch and change ***Display Color*** in ***Properties***
- Reset CPlane: ***Cplane World Top*** options



1.6 Geometry

Modelers support various geometry types. The most common types are NURBS and polygon meshes. NURBS is the primary geometry type in Rhino, and it enables high precision free-form modeling in very small tolerances.

1.6.1 NURBS geometry

Rhino uses NURBS curves and surfaces as its primary method to represent geometry. NURBS is an accurate mathematical representation of curves that is highly intuitive to create free forms and edit them[1].

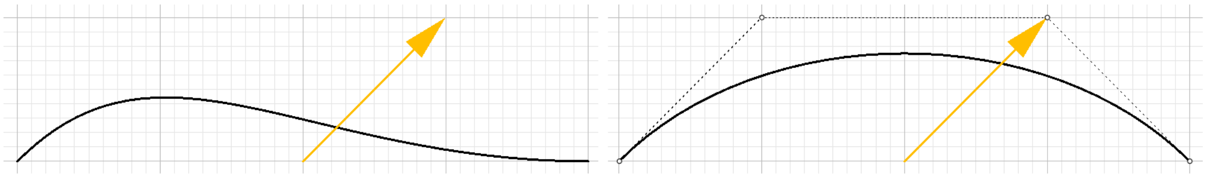
NURBS Curves

When representing curves using NURBS, the control structure makes it easy and predictable to edit. The control structure of a curve consists of a list of points that are used to construct the curve and also to edit it. Rhino *Curve* command draws a NURBS curve by default. For example, here are the steps to create a curve.

NURBS curves

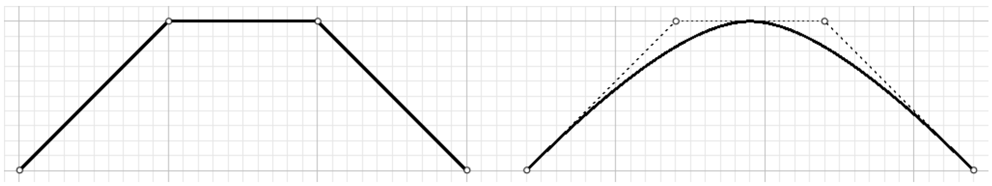
At any point after you create the curve, control points can be turned on (use *PointsOn* command) and dragged to adjust the curve interactively (use *PointsOff* command, or *Esc* to turn off control points when done), as in the following.

[1] For details about NURBS geometry, refer to chapter 3 in the *Essential Mathematics for Computational Design* (free download)



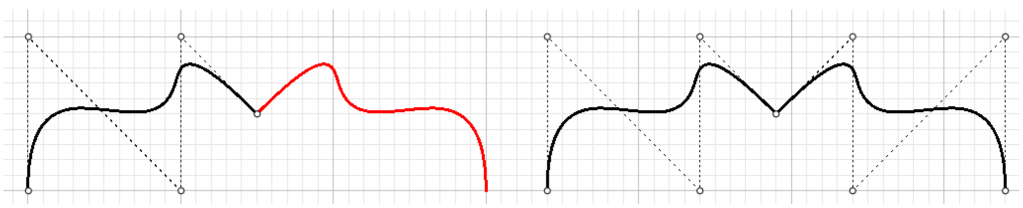
Curve editing by moving control points

Other than control points, you need to pay attention to the curve degree. The degree determines how smooth a curve is. In simple terms, degree 1 creates polylines, degree 2 creates arcs, and degree 3 creates smooth curves. Higher degrees simply increase the smoothness and are not very commonly used. The degree of the curve in the example above is set to 3 (the default when you run *Curve* command), but there is a *degree* option that you can change. Here is how our curve looks when set to degrees 1 and 2 using the same control structure.



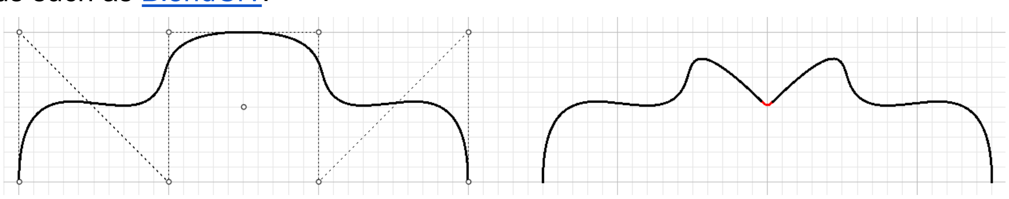
Curves degree affects how smooth they are

It is possible to join two curves together if their end points coincide. The new curve is called polycurve (or polyline if it consists of lines).



Joining curves create kinks

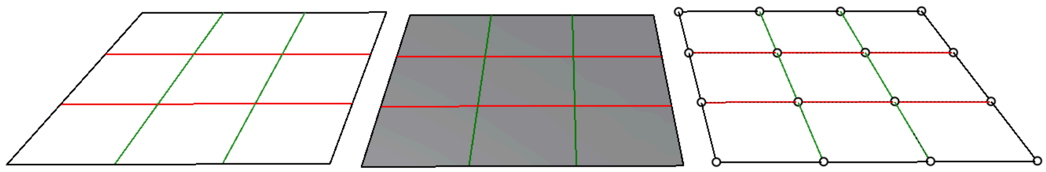
Notice that joining smooth NURBS curves together creates a “kink”. Deleting the control point directly on the kink fixes the curve continuity, but you can also join curves smoothly using commands such as BlendCrv.



Delete control points or join curves with a blend helps create smooth curves from two curves input

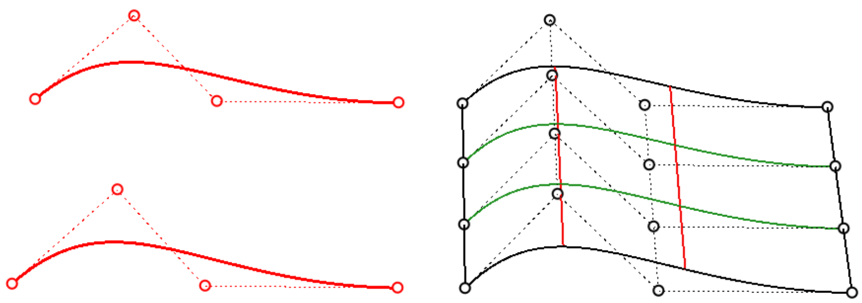
NURBS surfaces

Surfaces in Rhino are created using NURBS. You can think of them as a network of NURBS curves in two directions. They are infinitely thin, infinitely flexible, mathematically defined digital membranes. Surfaces are represented on screen by either some outline curves plus some interior curves called isocurves, or by a shaded picture which makes a surface appear to have some substance. How surfaces are painted on the screen is dependent on the display mode in the viewport, and does not affect the surface NURBS structure in any way. The important thing to remember about surfaces is that they are defined with great precision at every point. They are not approximations.



NURBS structure as a grid of control points in u and v directions

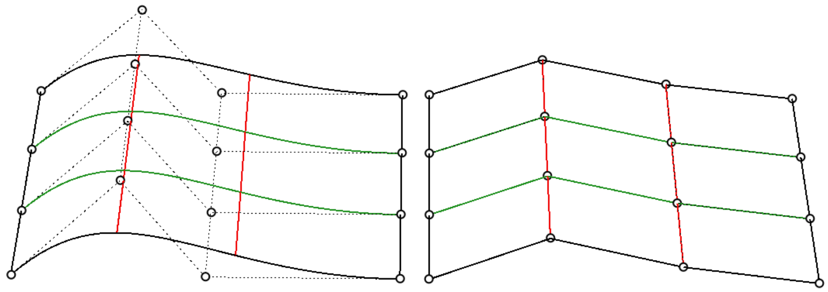
Two NURBS curves can be turned into a surface using Loft command. Notice that the control structure (the collection of control points) is very similar to the curves used to make the lofted surface. In the same manner, a NURBS curve can be edited by dragging control points, NURBS surfaces are modified when dragging control points.



Loft two curves to create a surface preserves input NURBS curve structure when possible

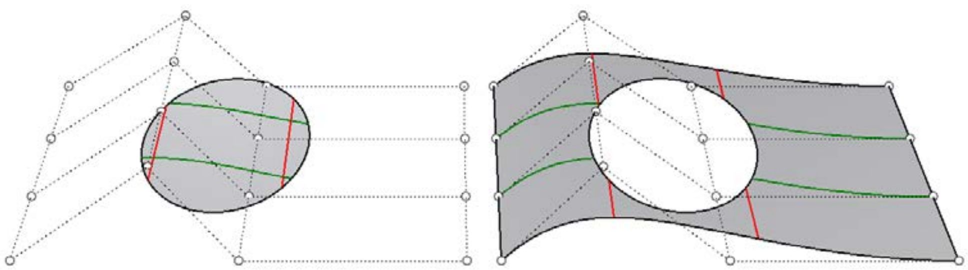
There are a few concepts associated with NURBS surfaces that are useful to remember. NURBS surfaces are rectangular with two main directions referred to as the “u” and “v” directions. The two directions do not have to be linear, so surfaces may bend in space. It is always useful to change the color of isocurves going in u-direction (e.g. set to red) from those going in v-direction (e.g. set to green). It keeps you aware of the general structure of your surface. To do that, go to Rhino Options > View > Display Modes > [your mode] > Objects > Surfaces > Isocurve color usage

There is the idea of a “degree” in NURBS surfaces, and it defines the level of smoothness of the surface. With degree 1 surfaces, you end up with a faceted surface that has creases surrounding each unit area surrounded by adjacent control points. The higher the degree, the smoother the surface, but you typically need more control points to achieve it. Commonly used degrees are 1, 2, 3 and 5.



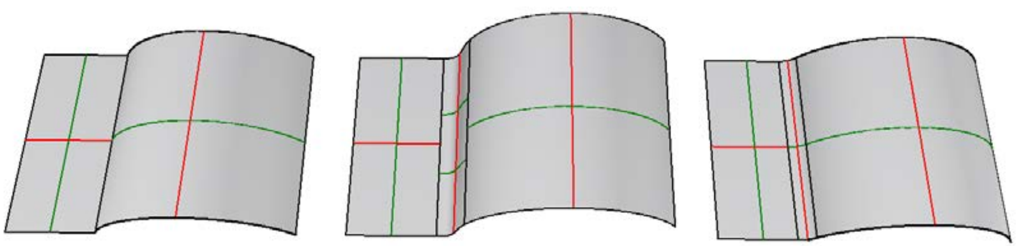
Surface degree affects smoothness

NURBS surfaces typically have a rectangular boundary. If the surface is non-rectangular, then it is likely to be “trimmed”. Trimming involves defining a boundary (using curves). The underlying surface remains rectangular, and you can always Untrim the boundary to go back to the original surface. Adding holes to a surface is the same as adding an irregular boundary. It involves trimming an inner boundary. Again, the underlying structure of the NURBS surface remains intact in all cases.



Trimming surfaces does not change the underlying NURBS structure

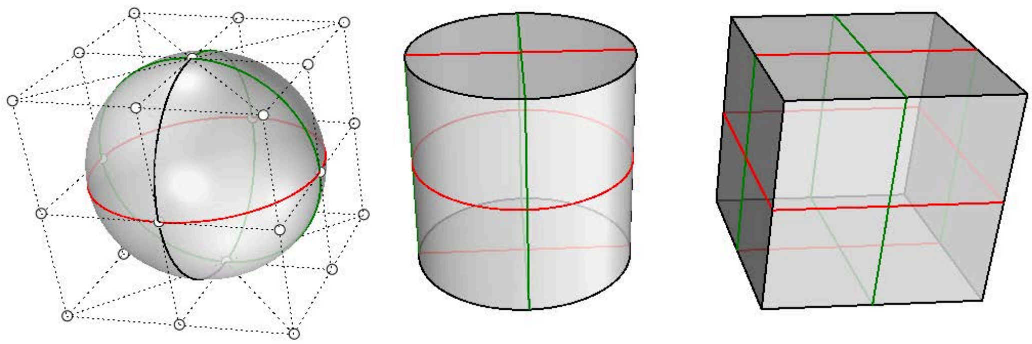
It is possible to join surfaces together to make a bigger object. That happens if the two surfaces can be joined along at least one edge with the model tolerance. If you don’t plan carefully, the two surfaces might not connect smoothly. There are tools in Rhino to blend surfaces with the desired smoothness (or what is technically called continuity). One example is to use the BlendSrf command. It is also possible to blend or round the edges connecting two surfaces stitched together using a command such as FilletEdge and BlendEdge.



Joining two surfaces with or without blending

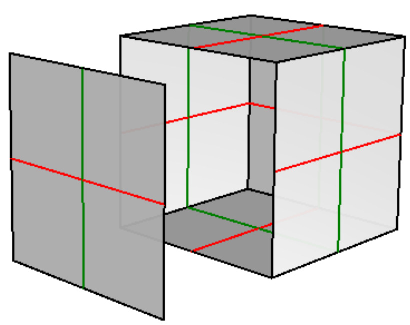
1.6.2 Solid geometry

Rhino NURBS surfaces are infinitely thin; they have zero thickness. Enclosing a volume can be achieved with one or more surfaces stitched together as one object. The *Solid* menu includes all primitive forms such as Box, Sphere, Cylinder, etc.



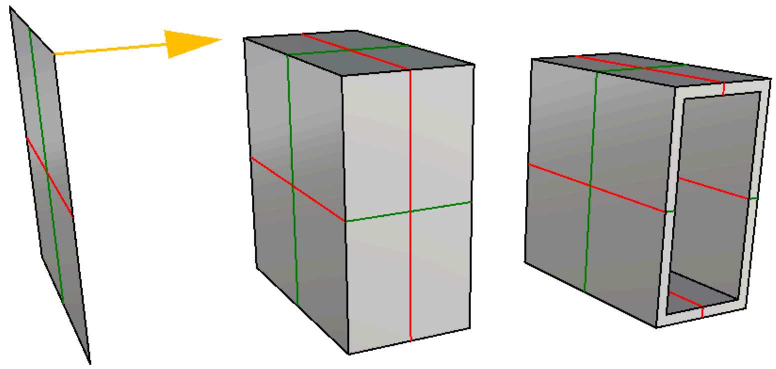
Solid primitives

The more common way to create a closed polysurface or solid in Rhino is to join enough single surfaces together to enclose a space producing what is called a boundary representation, or brep for short. A box is an example of this type of object. We call these objects solids, but it is important to remember that there is nothing inside them. They are volumes in space enclosed by the infinitely thin surfaces. If you remove one side of a box and look inside you will see the backsides of the five surfaces.



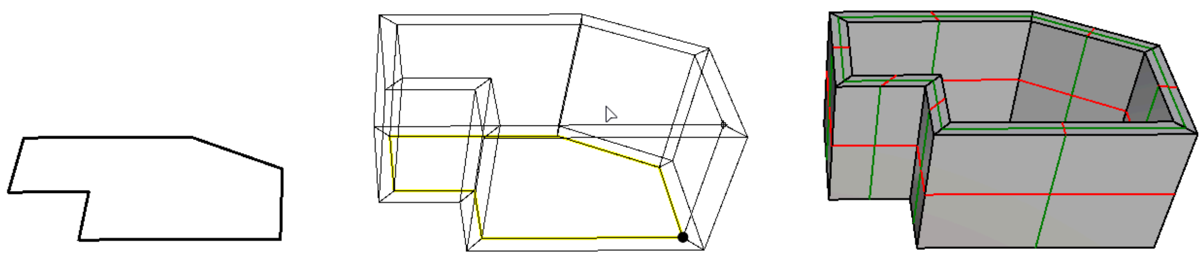
Solids are made out of surfaces joined together to enclose a closed volume

You can also create breps by extruding or offsetting surfaces. Shell command is another example of turning an open surface or polysurface into a solid with thickness.



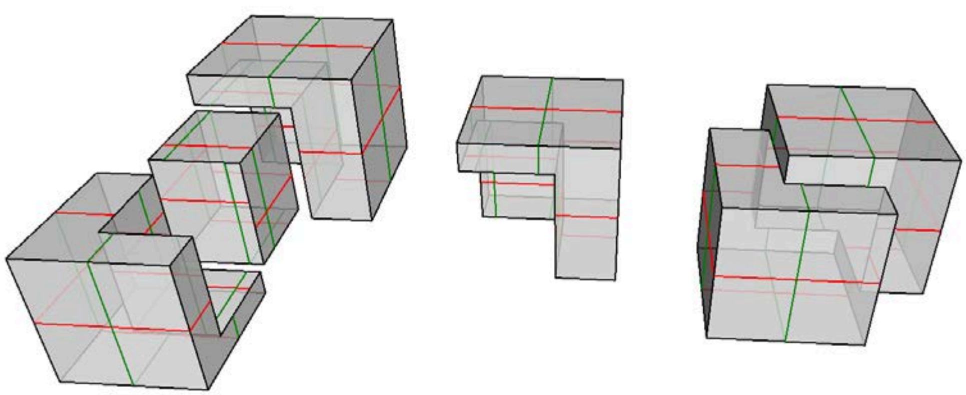
Solids from surfaces and polysurfaces. From left to right use: ExtrudeSrf then Shell commands.

You can create solids from curves as input using commands such as Slab.



Slab command starts with a curve, offset, then extrude normal to the curve plane

There are modeling operations that work only with solids; most importantly the Boolean Operations. Commands such as BooleanUnion, BooleanDifference, and BooleanSplit ensure that the results remain solid.

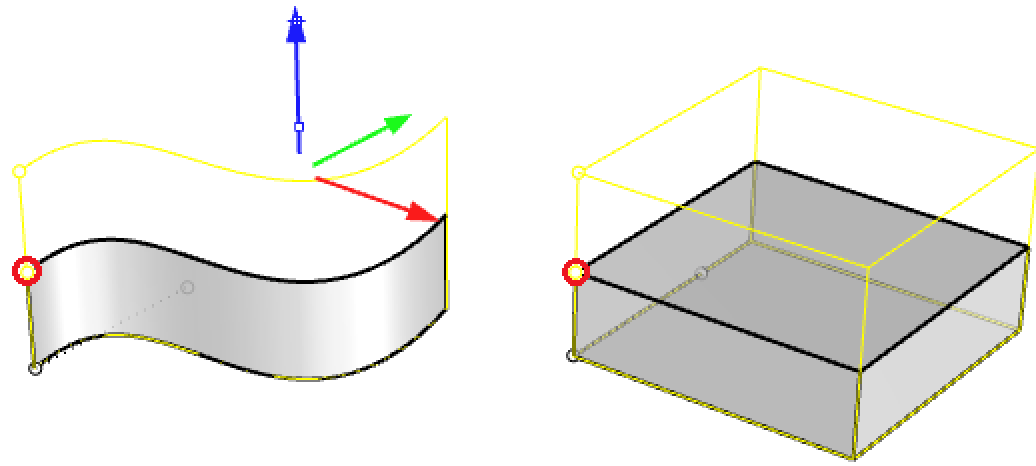


Solid Boolean operations. BooleanSplit (left), BooleanDifference (center), BooleanUnion (right)

1.6.3 Extrusion geometry

Another object type that is related to a polysurface and a solid is the lightweight extrusion object. Lightweight extrusion objects use less memory, mesh faster, and save smaller than the traditional polysurfaces.

In models containing large numbers of extrusions represented by traditional polysurfaces, performance can be sluggish due to the relatively high demand on resources. If the same objects use a lightweight extrusion type, the model is more responsive and leaves plenty of memory available.



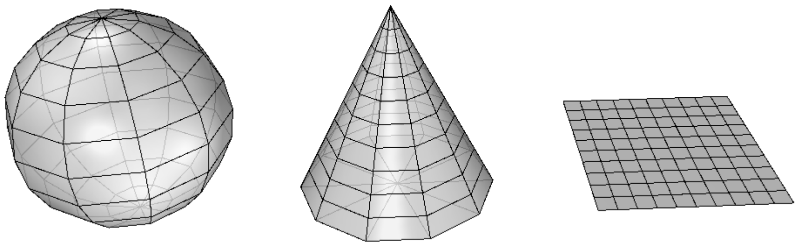
Editing extrusion objects with PointsOn command

In Rhino commands like ***Box***, ***Cylinder***, ***Pipe***, and ***ExtrudeCrv*** create lightweight extrusion objects by default. They take up much less file size and can be very useful for dense architectural models with numerous rectilinear components such as structural elements. Note that editing these objects might result in converting them to Breps (typically when the operation result has no reasonable extrusion replacement that can be found).

1.6.4 Mesh geometry

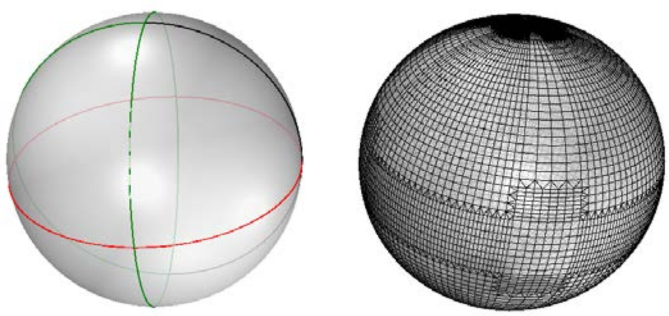
Rhino creates, edits, and otherwise uses polygon meshes. Polygon meshes are sometimes used to depict the same type of objects as surfaces, but there are important differences. Polygon meshes consist of a number, sometimes a very large number, of points in space connected by straight lines. These straight lines form closed loops of three or four sides, that is, polygons. The mesh geometry is described by 3D corner points (mesh vertices) creating faceted faces and straight edges. The mesh also describes the relationships between points to create edges and faces. Rhino supports mesh operations to edit meshes, boolean and repair.

You can create mesh models in Rhino using triangular or quad mesh faces. Meshes can be open (boundary edges are called naked) or closed. Mesh edges can also be non-manifold (connect to more than 2 faces).



Mesh primitives

Meshes are used to create shading and renderings on the screen. If you display a NURBS surface in shaded mode, you actually are looking at the polygon mesh derived from that surface. All polysurfaces in Rhino, have a render mesh attached to them to help create shading and rendering. The render mesh can be made more or less accurate based on speed and resolution requirements. Note that it is relatively easy to go from smooth breps to any resolution render mesh, but it is hard to turn a faceted mesh into a smooth brep. Meshes are also used for 3D printing. Deriving accurate meshes from surface models is important. Rhino has several tools to help accomplish this.

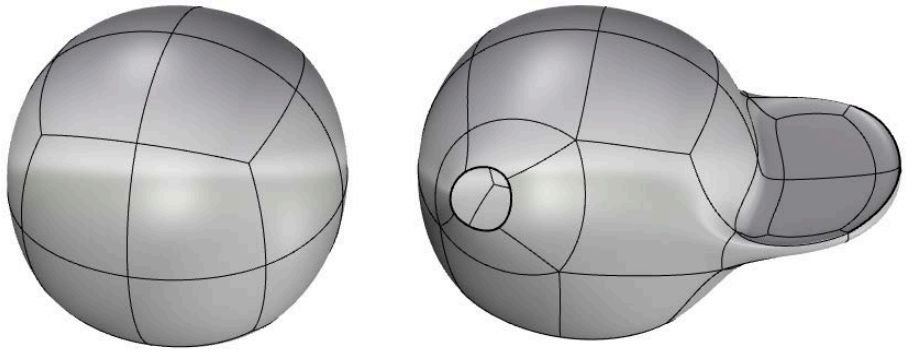


On the right is the extracted render mesh from the NURBS sphere on the left

1.6.5 SubD geometry

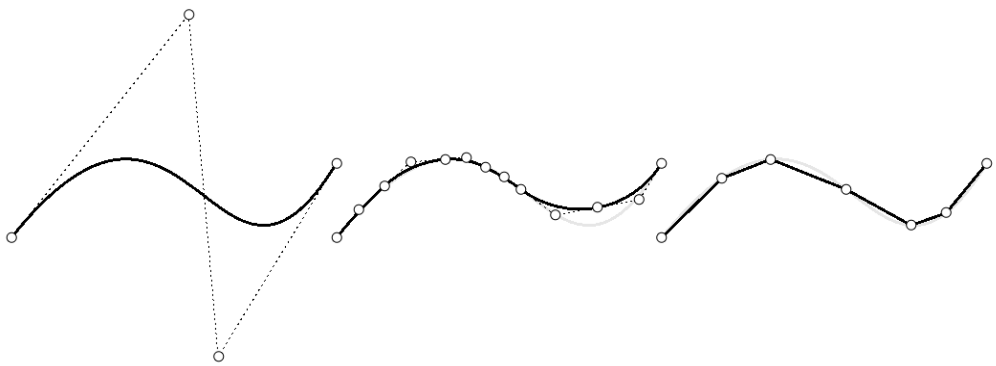
Rhino SubD objects are high-precision Catmull Clark subdivision surfaces. They can have creases, sharp or smooth corners, and holes. The Rhino SubD object is designed to quickly model and edit complex organic shapes. Unlike traditional mesh-based SubD implementations, Rhino SubD objects are NOT a subdivided mesh object.

Rhino SubD surfaces are predictable, measurable, and manufacturable. They can be converted to either high-quality NURBS or mesh (quads or triangles) objects when needed. Rhino supports many commands to create SubD objects as primitive geometry or from curves using extrusions, lofting or sweep. It also supports a rich set of editing tools.



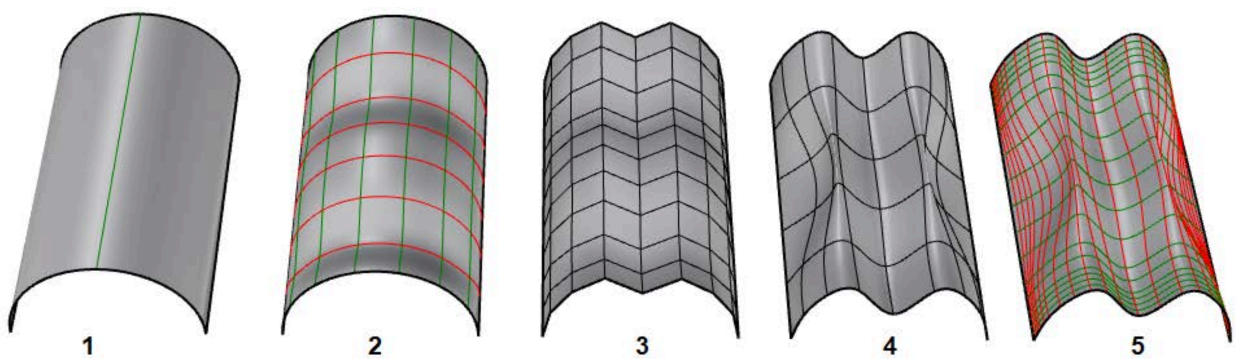
1.6.6 Transition between geometry types

Rhino supports converting between geometry types. For curves, a smooth NURBS curve can be turned into arcs or lines using Convert command.



Convert from NURBS curves to Arcs and Lines

For surfaces, there are four main geometry types: NURBS, extrusions, SubD and meshes. There are a number of commands in Rhino to convert between surface types.



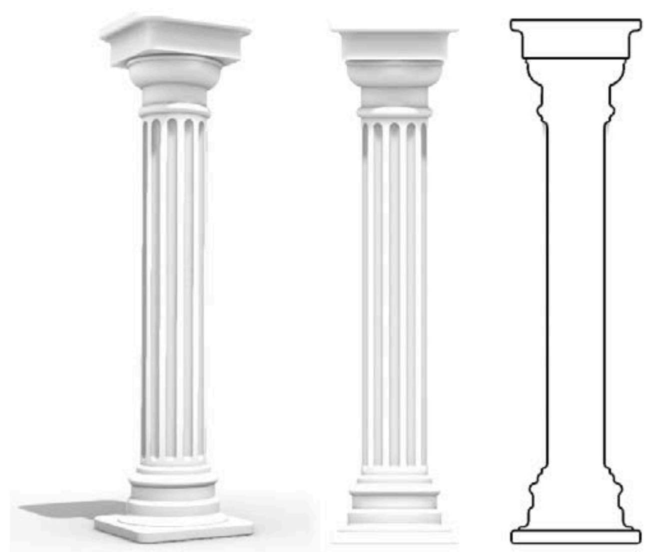
Transition between geometry types: (1) Extrusion (2) To NURBS (3) To Mesh (4) To SubD (5) To NURBS

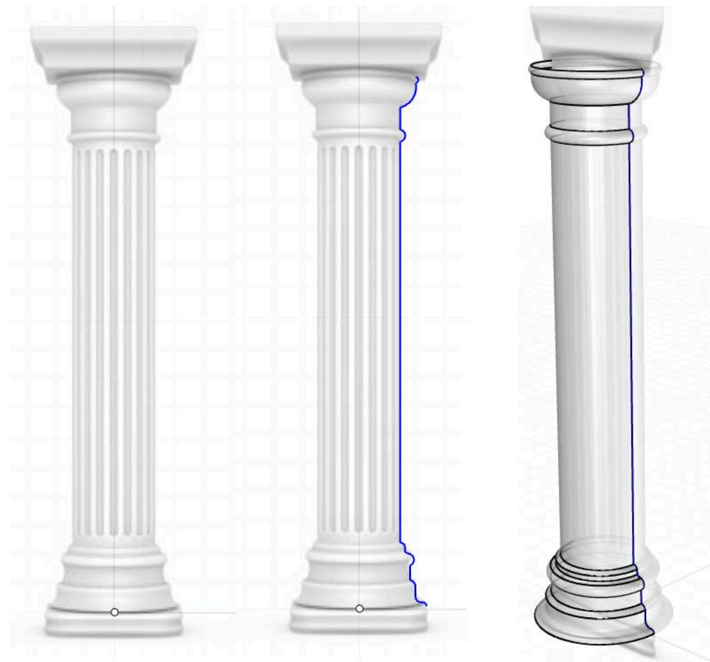
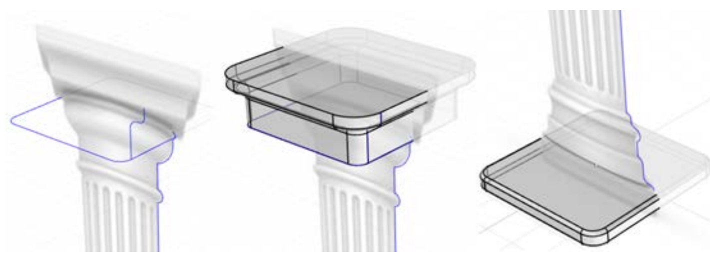
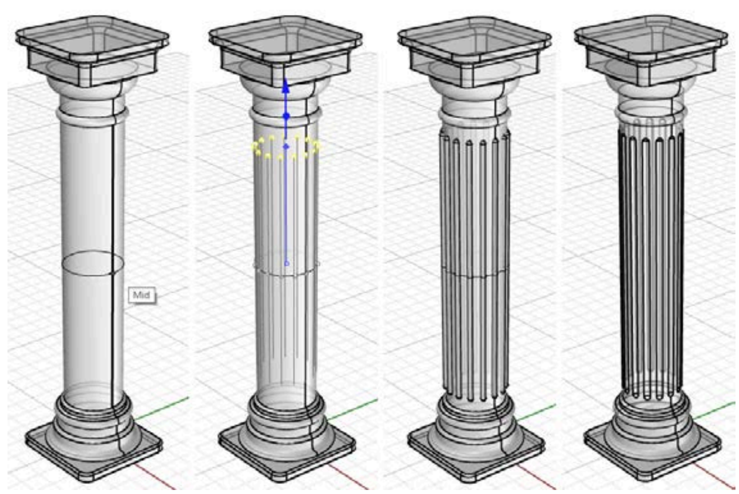
1.6.7 Geometry transformations

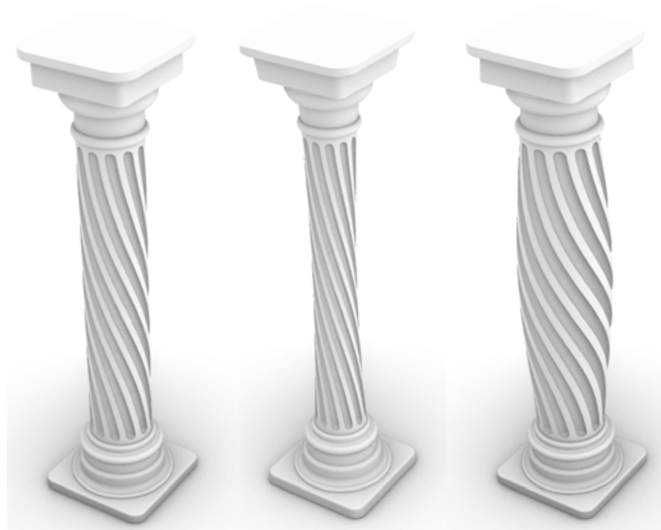
The common transformation commands include ***Move***, ***Scale***, ***Rotate***, ***Project*** and ***Mirror***. Those can be done in Rhino through running commands with these names. Some of them can be directly accessed using the ***Gumball*** control. There are other more complex transformations such as ***Twist***, ***Stretch***, ***Bend*** and ***Flow***. All transformation commands are accessible through the ***Transform*** menu

Resources: 3D and 2D Geometry
Rhino level 1 training: Editing Geometry Point editing Creating Deformable Shapes Modeling with Solids Creating surfaces Transforming Solids

1.6.8 Geometry tutorial

Exercise: Model a Doric column, then use advanced transformation commands to modify.	
Analysis; The column has three parts: the base, body and top. The base and top are rectangular while the body is cylindrical. **Modeling strategy:** Base: Create a rounded rectangle, extrude then fillet the edges to smooth. Body: Create a profile curve from arcs and lines, join then revolve around the centerline. Array cylinders without spherical ends around the column then use to subtract from the main body Top: Create profile curve, then sweep along a rectangular rail	

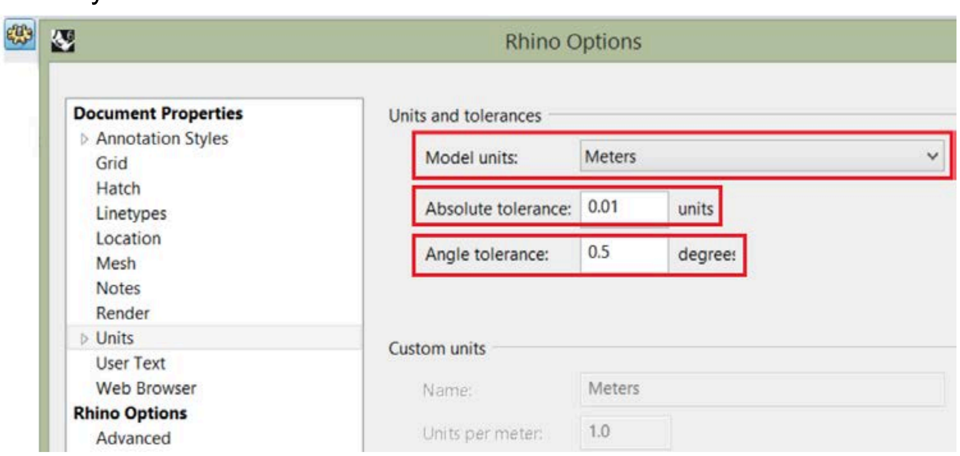
Import the column image using ***Picture*** command, Locate the center of the bottom at the World origin (0,0,0), and scale the image to the true height of your column. Draw the profile of the column body using ***Line*** and ***Arc*** with ***StartPoint*** option (tip: for productivity create Aliases for "***_Line***" and "***_Arc _StartPoin***t" commands for speed) then ***Join*** body profile curves. Use ***Revolve*** command to create the body of the column revolving the profile around the world z-axis ***Cap*** the revolved surface to create a solid (necessary for a Boolean operation later)	
Draw top rail using ***Rectangle*** with ***Rounded*** option. Use ***Line*** and ***Arcs*** for profile then ***Join*** Use ***Sweep1*** command to sweep the profile along the rail then ***Cap*** Draw rounded rectangle for the base, ***ExtrudeCrv***, ***Cap*** then use ***FilletEdge*** to round the top and bottom edges	
ExtractIsocurve to get the circle at the midpoint of the body ***Divide*** the circle to 15 segments (16 points) then extrude all points using the ***Gumball*** sphere handle and Shift key down (to extrude both directions). Alternatively, use ***Line*** with ***BothSide*** and ***Vertical*** options at the start of the circle, then ***ArrayPolar*** around the circle ***Pipe*** with ***Multiple*** and ***Cap=Round*** to create all pipes around the body Use ***BooleanDifference*** to cut out the pipes from the body and complete the body.	

Exercise: Advanced transformation of polysurfaces	
Use ***Twist*** transformation to create a twisted body of the column Experiment with ***Stretch***, ***Bend*** and ***CageEdit*** commands to sculpt the column.	

1.7 Units and tolerances

1.7.1 Overview

Rhino is always modeled in real full scale. So if your building is measured in meters, you should use “Meters” as your units in Rhino. You can set your model units under Document Properties>Units. It is important that you check your units before starting any modeling. Units affect the scale of your model.



Units and tolerances settings

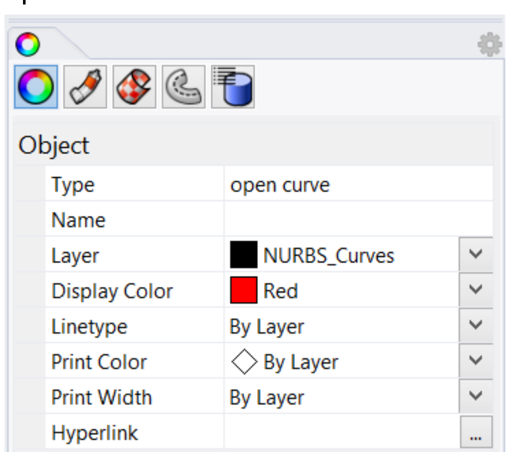
Rhino modeling involves two types of operations. One is exact, and the other is approximate. For example, when you create a circle, you specify the exact location in space for the center, and exact radius. Now if you project this circle on a NURBS surface, this cannot be an exact operation in NURBS modeling. The amount of approximation is made within some tolerance. You can set that tolerance in the “Absolute tolerance” field. The smaller the tolerance, the tighter the model, but it may involve more complex structure, or take longer to calculate. You should consider what the model will be used for downstream when deciding what tolerance value to use. If the model is only used for visual representation, or renderings, then tolerances can be relaxed. If you will pass the model to be printed or analyzed in some engineering application, you need to check acceptable tolerances in the target application. Angle tolerance is used to evaluate if two curves or two surfaces are tangent within that tolerance.

Resources: Set up units and tolerances
Video tutorial: Tolerances tutorial in Rhino: https://vimeo.com/85108857

1.7.2 Attributes and data

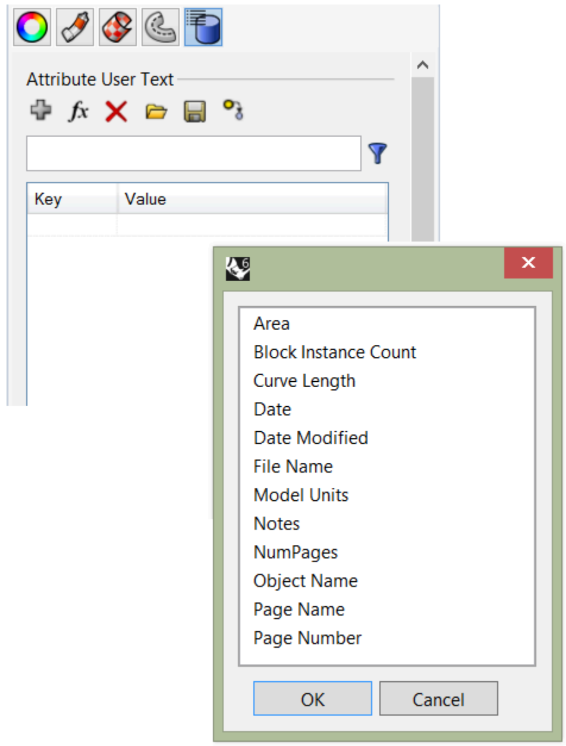
Modeling objects consist of two parts: geometry and data. You can think of data as information that describes the geometry attributes such as name, color, material, linewidth, group affiliation, etc. Data also includes custom information specific to the modeler or the workflow.

Rhino objects have attributes that can be accessed and set directly using the properties panel. Each geometry type has a different set of data and attributes associated with it, and they all appear as icons under the properties main tab.



Curve properties

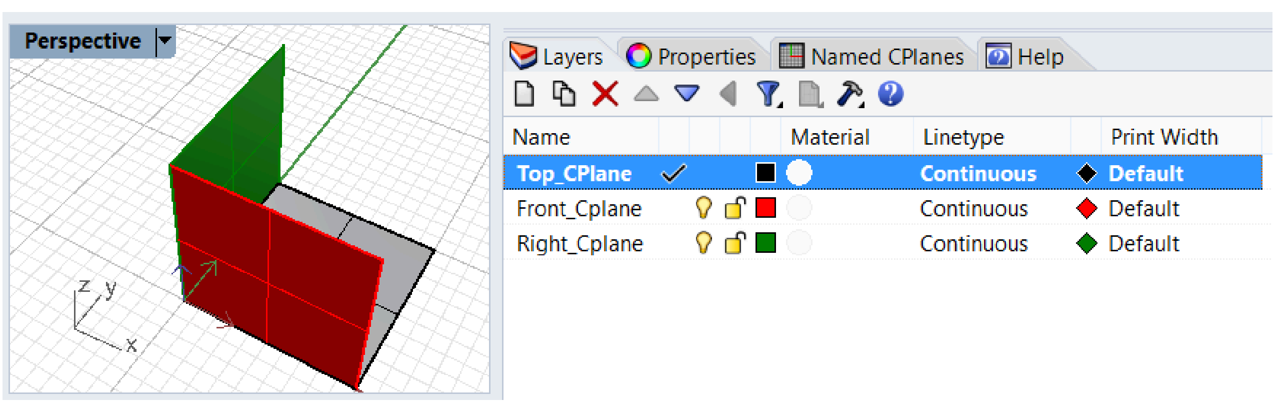
Users can attach custom data to objects using a user string. You can add functional attributes such as length or area, but you can also create your own key and value that is specific to your modeling workflows. For example, you may tag all your roof panels with the “Area” Key and the area of each panel is calculated and attached. It also changes dynamically when the panels are scaled.



User text is part of the object properties panel

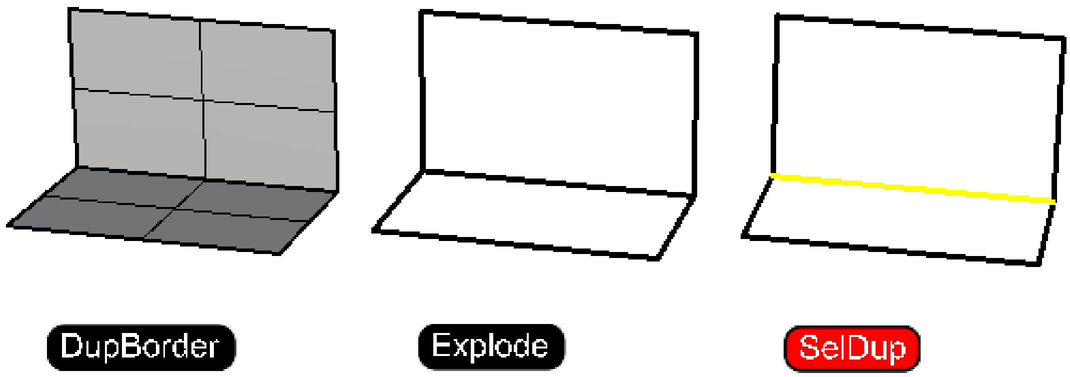
Rhino plugins can access objects' user data and attach any information that can inform how certain objects display or behave.

Designers typically organize their model geometry and data in [layers](). This helps isolate and group data in one place. For example, you might put all your lights in one layer, site geometry in another and so on. Other than grouping data, layers allow you to hide, select all objects, or assign attributes. For example, you can assign color, lineweight or material to the layer instead of objects. There is extensive information about layers in the Rhino help file.



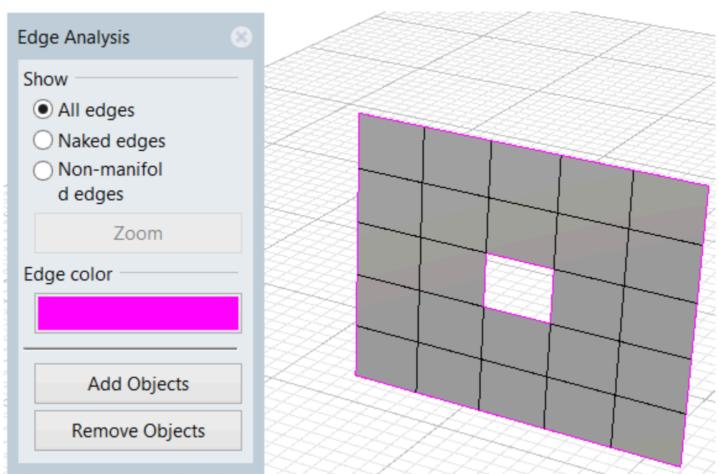
Layers to group data and assign general attributes

Selection tools by object type or attributes are other important tools to help navigate your model. For example, you might need to hide all points in the model. You can run SelPt, then Hide commands. You might also want to delete all duplicate geometry in a model. The command SelDup selects geometry that perfectly overlap (within your document tolerance), then runs the Delete command. It is also possible to select objects with a certain name or color. Knowing how to use selection commands can improve your productivity.



Selecting duplicate geometry using SelDup

There are also specialized tools to identify specific information about your model. For example, you might need to identify and zoom to a naked edge in a dense mesh using ShowEdges command. This is important with mesh modeling to create clean closed mesh ready for 3D printing.

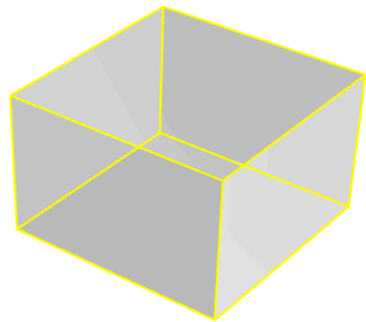
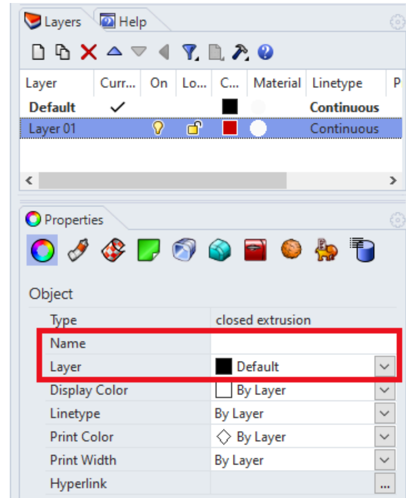


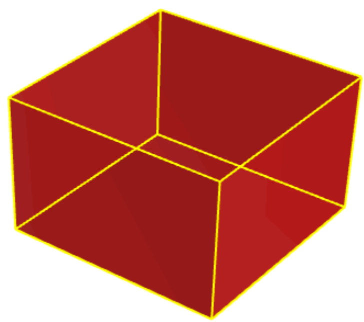
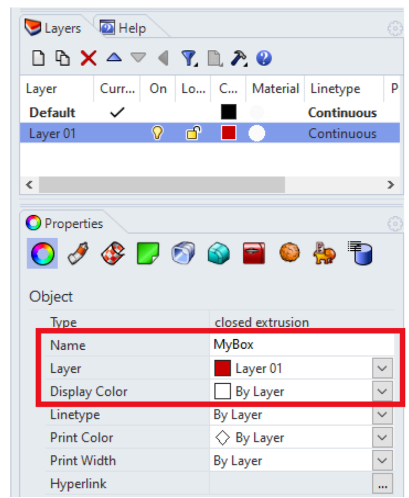
ShowEdge is a specialized geometry navigation tool

1.7.3 Attributes tutorial

Modify layer and object attributes and add user text such as area and volume to an object

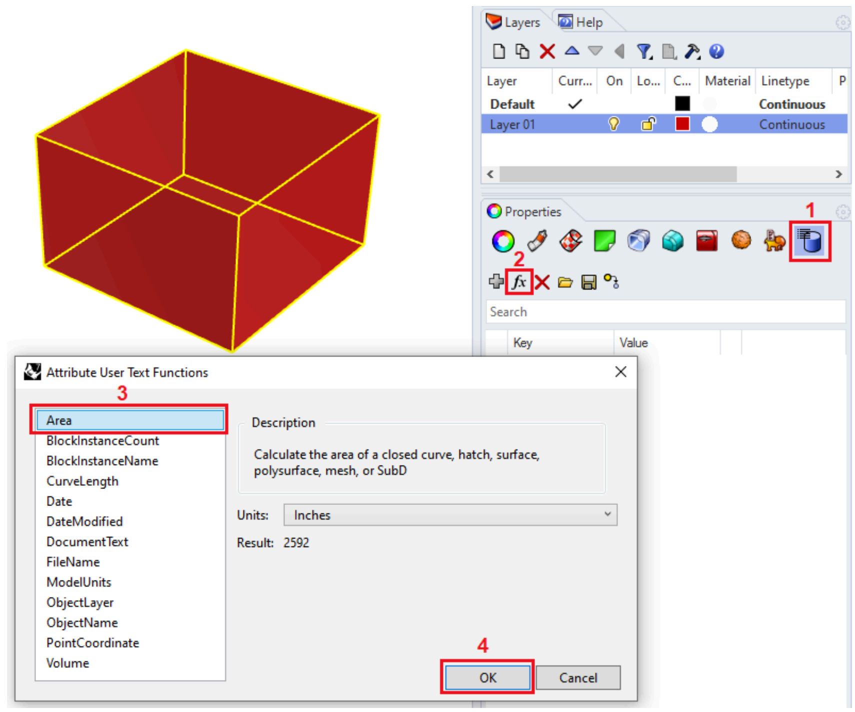
<table>
<tr>
<td>The box is located in the Default layer

Make sure the box is selected to see its properties in the Properties panel. Notice that the object has no name.</td>
<td>

</td>
</tr>
<tr>
<td>Move the object to Layer 01 and notice that the box color changes too because its Display Color property is “By Layer”.

Make sure the box is selected to see its properties and you can enter a name for the object</td>
<td>

</td>
</tr>
</table>

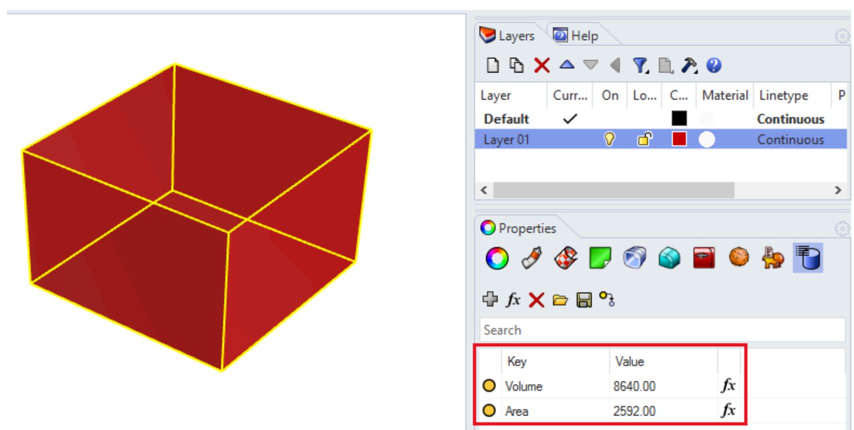
Add user text to calculate the surface area of the box and attach to the object using the User text tab in the ***Properties*** panel, then:

1- Clock on the user data icon in the Properties panel.
2- Click the function icon to open available functions.
3- Select ***Area*** (or ***Volume***)
4- Click OK

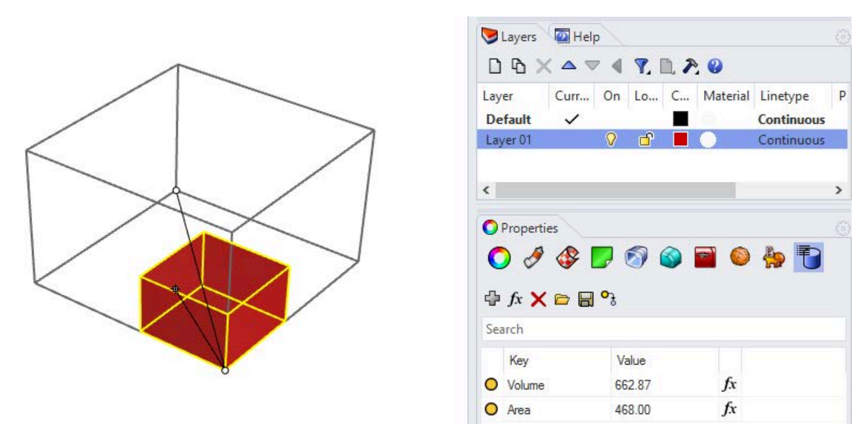


The area and volume are calculated and displayed in the Key/Value box of the user text properties.

Note that the user text is attached to the object as data that persists even after you transform the object



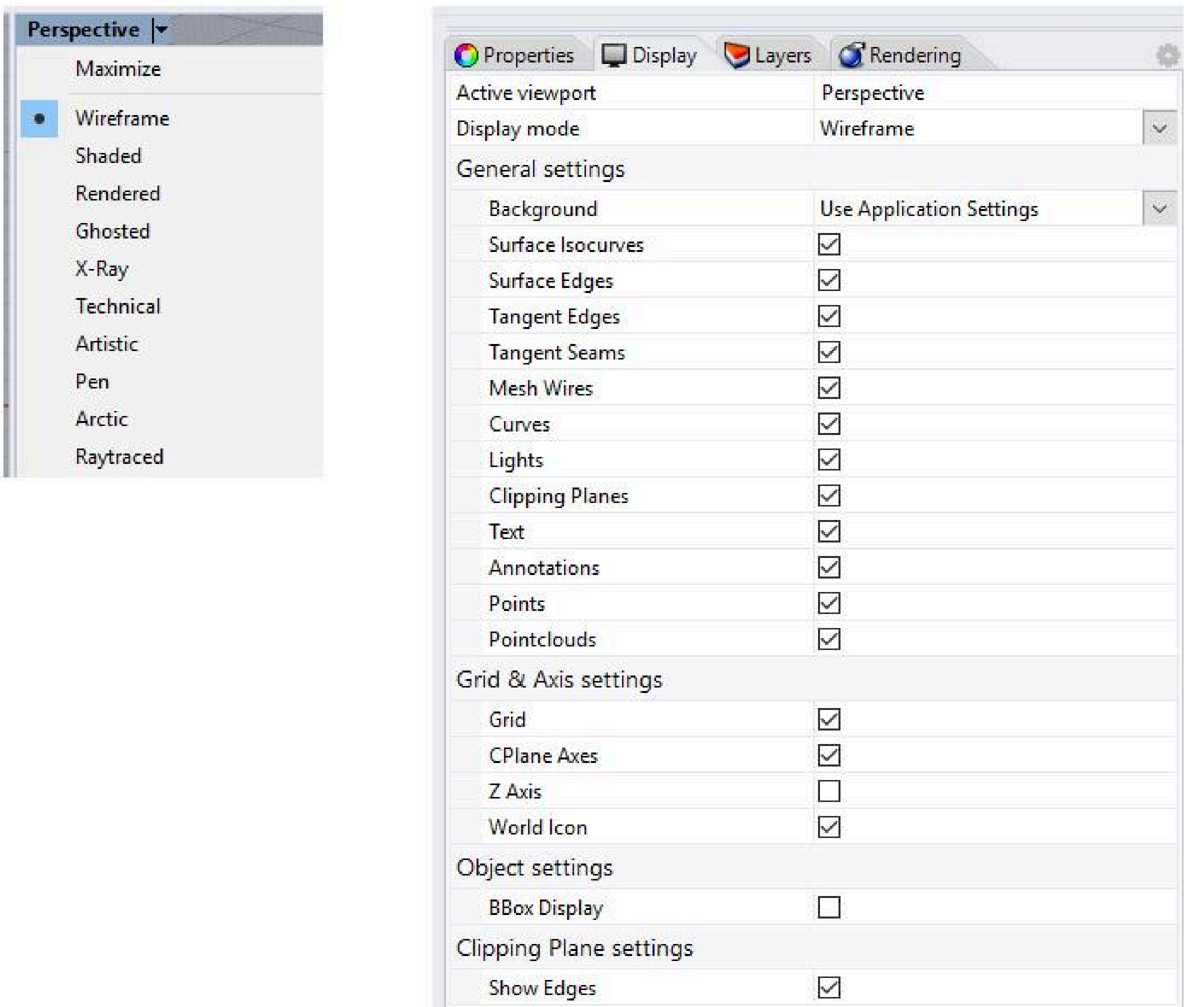
When scale the box, the area and volume recalculates dynamically and display in the user text area



1.8 Visualization

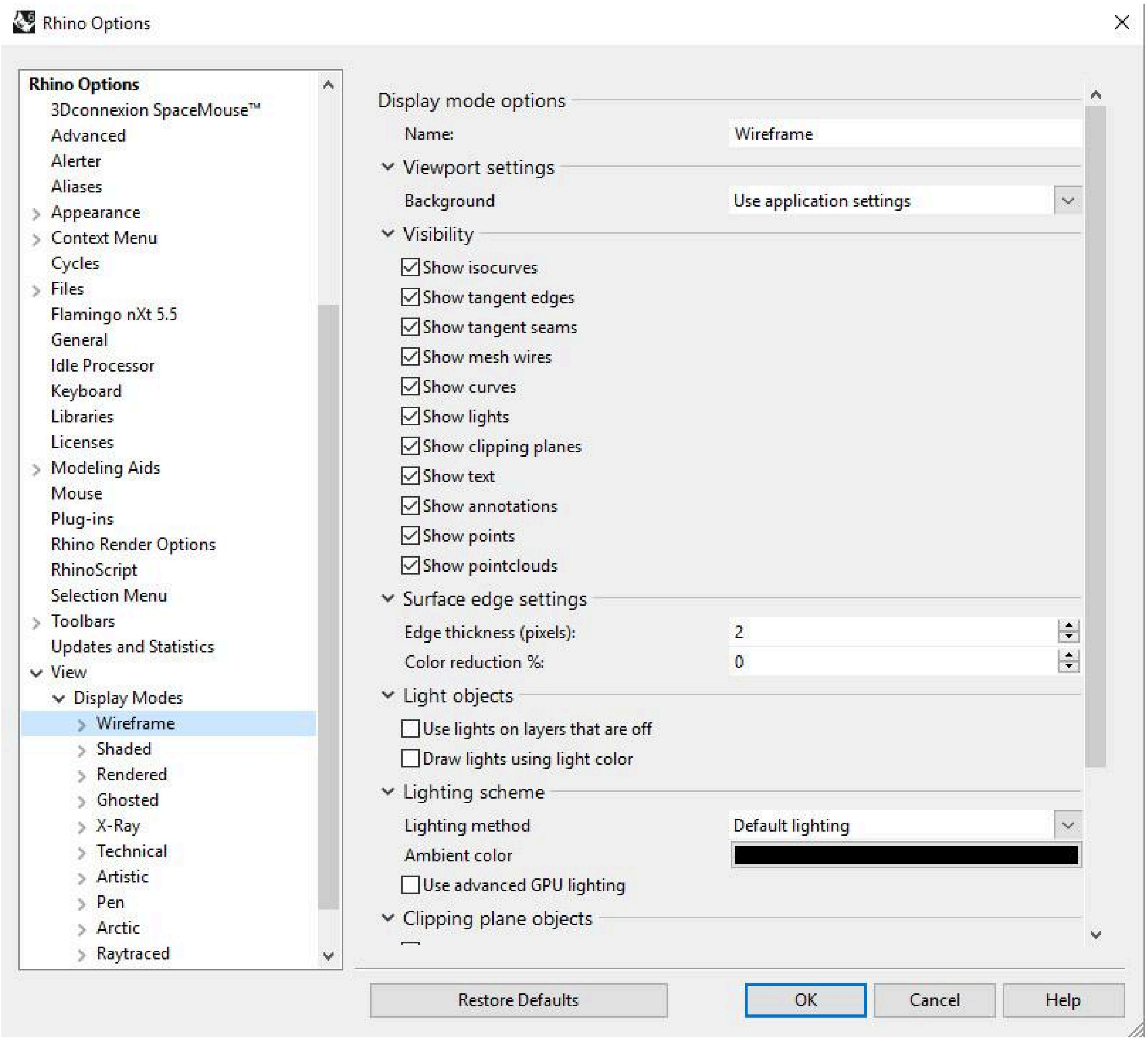
Communicating and presenting design work to others is an integral part of any creative workflow. Rhino includes a variety of options for visualizing and rendering. Display modes, interactive rendering and a myriad of third-party plugins for rendering ensure you can capture and communicate your design intent as you create.

Display modes in the Rhino viewport can be selected via the viewport drop-down menus or by way of the Display panel at the right of the Rhino UI.



Display models in Rhino

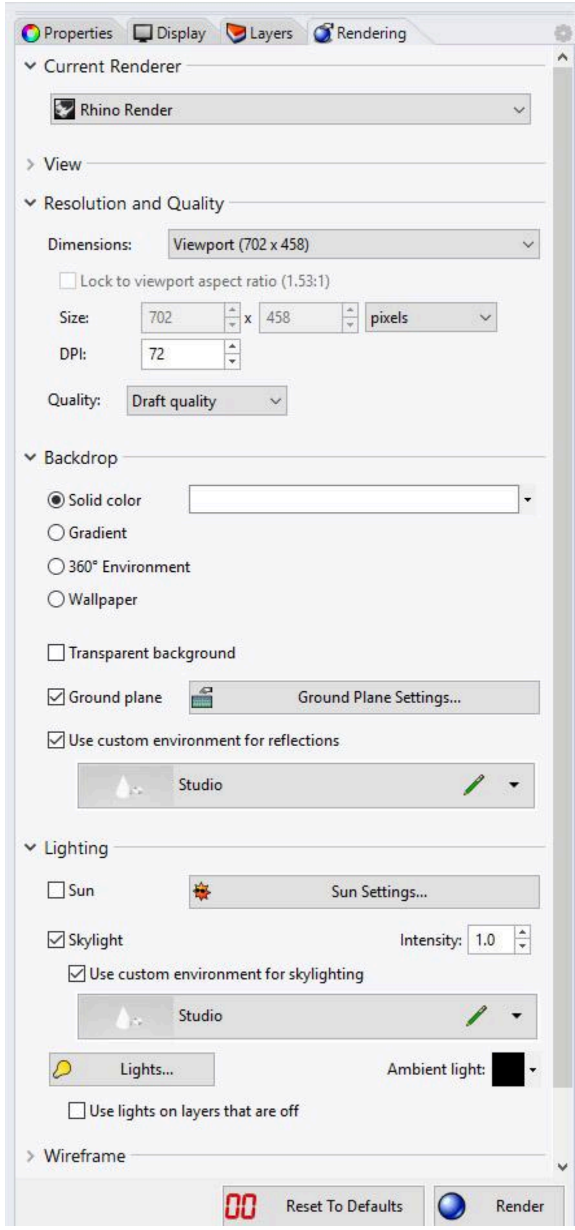
Display modes can also be created, customized and saved through Rhino Options > View > Display Modes.



Display models customization through Rhino Options

Resources: Display modes
Video tutorials: Display modes in Rhino: https://vimeo.com/84982383 Create custom display modes: https://vimeo.com/260992627

Rendering controls for the default Rhino *Render* as well as the interactive rendering display mode called Raytraced can be accessed in the *Rendering* panel. If you are using a separately installed rendering plugin in Rhino, refer to that plugin's documentation for UI locations. The Render drop down menu, then *Current Renderer* flyout can be used to switch between rendering plugins.



Rendering panel

Resources: Texture mapping and Rendering tutorials
Rhino level 1 training: [Rendering and lighting] **Video tutorials:** Texture mapping in Rhino: https://vimeo.com/268449998 Rendering materials: https://vimeo.com/274523131 Introduction to environments: https://vimeo.com/288162458 Realtime rendering: https://vimeo.com/259726371 Decals: https://vimeo.com/259779341 Advanced decal: https://vimeo.com/259885405

1.9 2D drawing and annotation

Extracting 2D drawings out of 3D models is common to most 3D modeling applications. In Rhino there are few commands to help section through models, orient planar curves to any plane and extract outlines of scenes. ***Section*** and ***Contour*** commands help extract one or series of sections through a model. Sections remain in the true location of the 3D modeling space, but can be moved or oriented to any plane using ***Orient*** command. In essence, a section is the result of intersecting a plane with the model. Rhino has a robust ***Intersect*** command that can also be used to create sections.

Perspective or parallel views of a model can also be extracted as planar line drawing along with hidden line information and boundaries using the ***Make2D*** command in Rhino.

Section, ***Contour*** and ***Make2D*** create vector drawings consisting of curves and points that are actual geometry objects in Rhino. Those can be snapped to and annotated. If you simply need an image of your section, or would like to look inside your model in the viewport, then you can use another set of commands. ***ClippingPlanes*** and ***ViewCapture*** are two very commonly used commands that are used for that purpose.

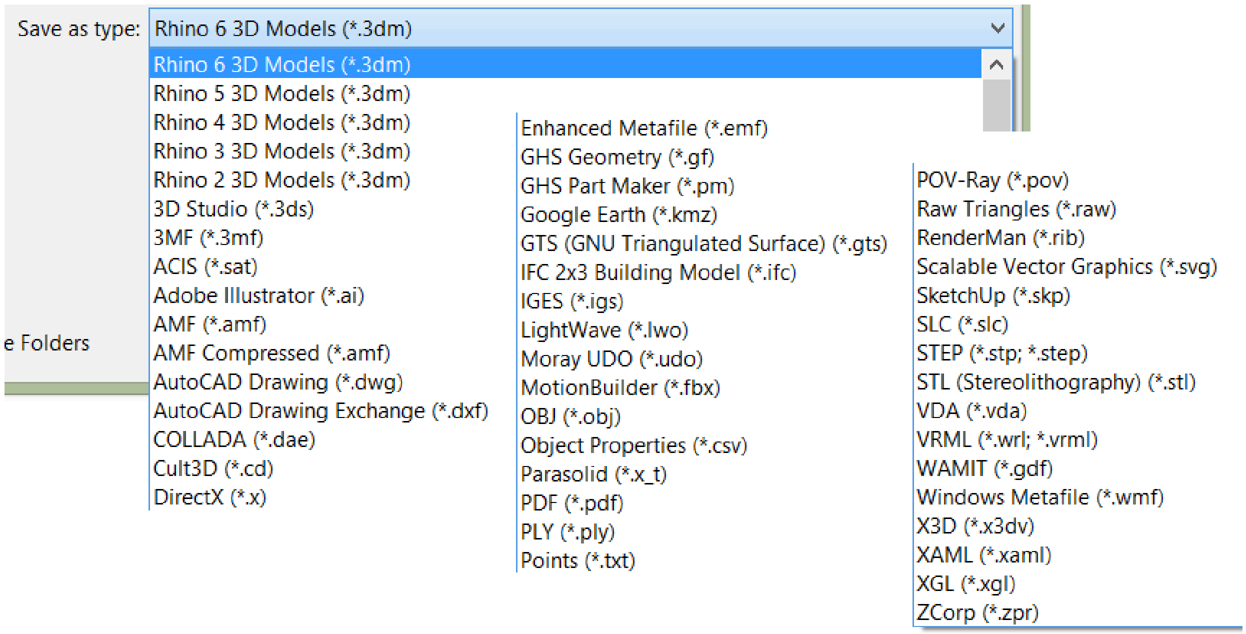
Rhino also supports a special paper space viewport called ***Layout***. It supports placing views of the model to scale, add paper titles and ***Print*** to PDF and other formats.

Resources: Drawing and annotation
Rhino level 1 training: Annotating Printing and Layouts

1.10 Files

3D modeling applications save geometry and data to a file using a specific format. When a file format is publicly available (open source), other applications can easily add support to open files using that format. This helps promote interoperability among different applications and flow of information. Rhino publishes its *3DM* file format in *OpenNURBS* allowing other applications to fully support it.

Some applications only support a limited number of geometry types, or specialized object types. This makes interoperability more challenging, and some data is bound to be lost in the conversion. Rhino pays great attention to file I/O and supports many different import and export formats, making it possible to model in Rhino and then export your model to downstream processes, or import models from other software applications into Rhino. For a complete list of import and export file types refer to the Rhino Help > Contents > File I/O > File Formats.

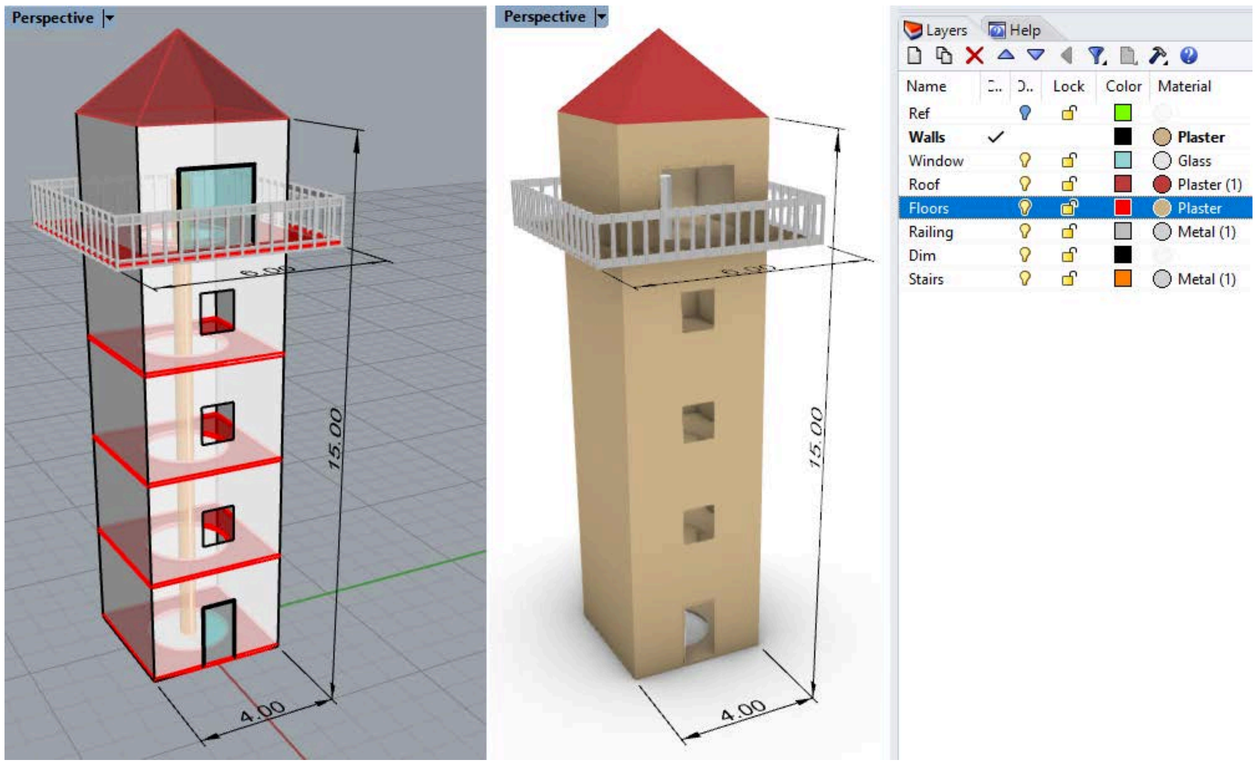


The Rhino export file formats

Tutorial: Files and templates
<u>Rhino level 1 training:</u> Importing and exporting files Printing and Layouts **<u>Video tutorial:</u>** Setting templates in Rhino: https://vimeo.com/86730224?templates

1.11 Lighthouse Tutorial

Model the following lighthouse using Rhino

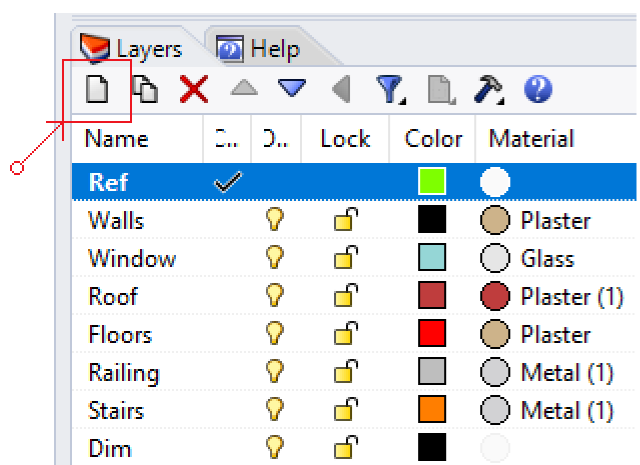


Modeling steps:

Create new layers (using the marked new layer icon) for different parts of the lighthouse and assign different colors and appropriate material.

The ***Ref*** layer is for reference geometry to help modeling.***Ref*** and ***Dim*** layers do not need material assignment.

To place a geometry inside designated layers, you can either make the layer current before creating the geometry, or select the geometry after creating and in the properties panel, change the layer to the desired one.

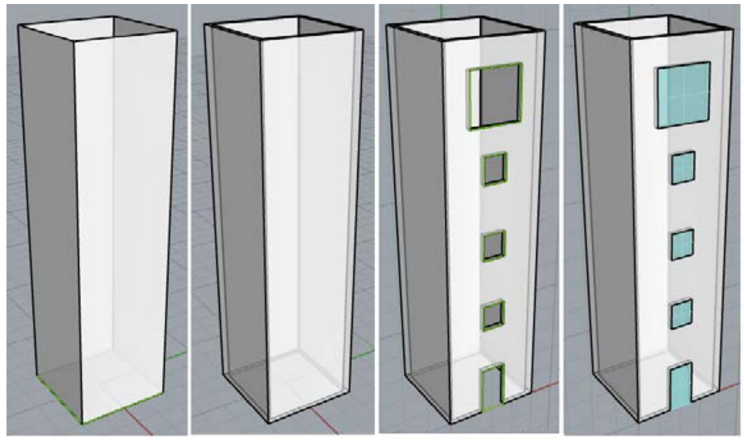


Draw a rectangle centered around the origin that is 4x4 meters in the ***Ref*** layer. Use ***ExtrudeCrv*** the base curve to create the wall.

Use ***OffsetSrf*** command to create wall thickness. Set Solid=Yes. Use ***MergeAllFaces*** to clean top coplanar faces and turn into one face.

Change ***CPlane*** to align with elevation. Draw rectangles for the openings, then use ***MakeHole*** to create the holes.

Use ***PlanarSrf*** command to create windows from the rectangles.

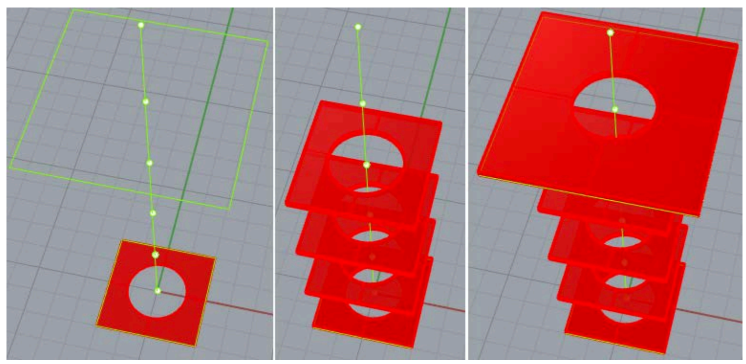


Draw a vertical ***Line*** that is 15m in length and ***Divide*** into 5 segments to mark the floors.

Create a ***Plane*** in for the base floor. Draw a ***Circle*** that is 1.2m in radius. ***Trim*** the plane with the circle. ***ExtrudeSrf*** the trimmed surface by 0.2m and set ***Solid=Yes*** option..

Copy the base floor to create the 3 other floors.

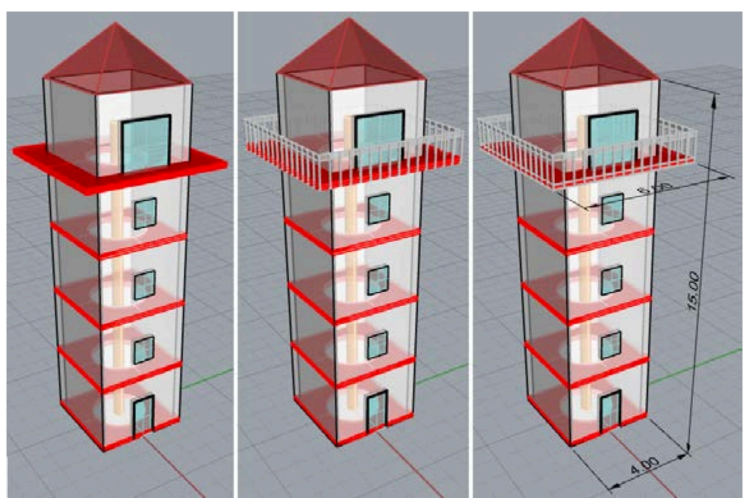
Repeat the Rectangle, Plane, Circle, Trim and ExtrudeSrf to create the observation deck.



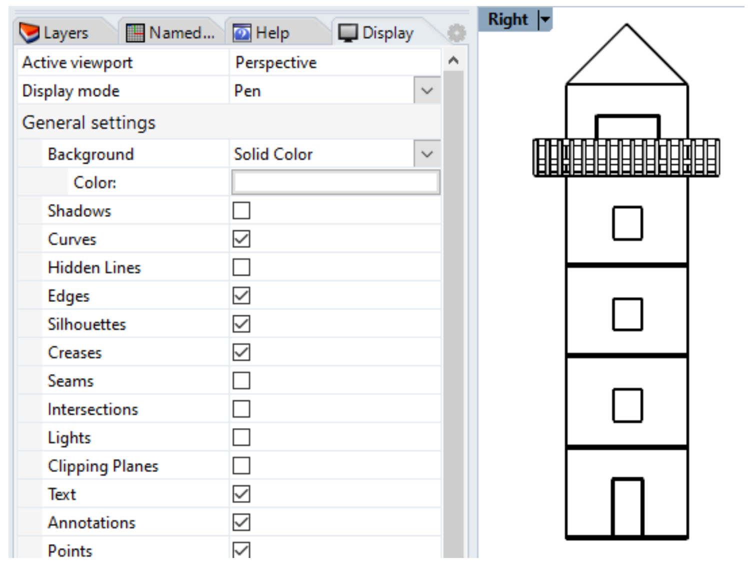
Run ***Pyramid*** command to create the roof. Run ***Cylinder*** command to create the stairs pole.

For the railing, ***Divide*** the top floor rectangle, select divide points and extrude vertically by the railing height. Use ***Pipe*** command with ***Multiple*** and ***Cap=Flat*** options to create the railing. Extrude the base curve then ***Move*** vertically to create the handle.

Use the ***Dim*** command to create the dimensions on the floor. Change ***CPlane*** to align with the elevation to be able to create the vertical dimension. Note the dimensions are always measured and created parallel to the active CPlane.

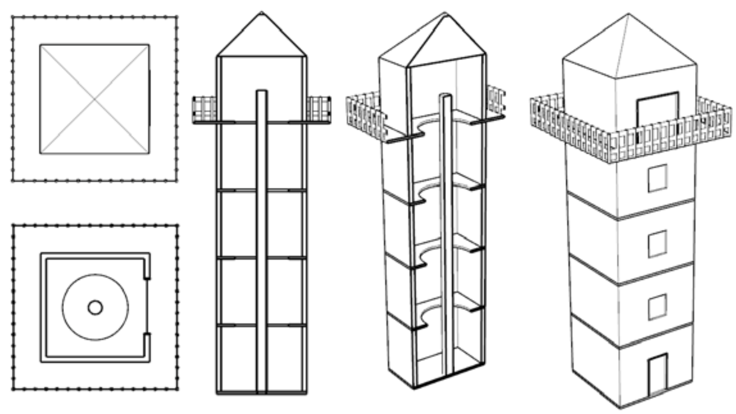


<table>
<tr>
<td>Use the Pen or Technical display mode to get raster images of different views.

You can set the Pen view to have a solid color background. In the Display panel, change background to Solid Color and set to the color you like</td>
<td>
</td>
</tr>
<tr>
<td>Extract parallel and perspective views.

Use ClippingPlane to cut out part of the model in display and create sections and section perspectives.

To extract vector drawing (curves), use Make2D and Section commands. Make2D supports ClippingPlanes and calculates hidden lines and other features.

When extract sections using Section Command, you can orient to xy-plane using Orient, Rotate or Flow commands</td>
<td></td>
</tr>
<tr>
<td colspan="2">Here is a quick reference of the commands that are commonly used in rectilinear modeling:
Geometry creation: Point, Line, Rectangle, Polyline, Circle, Divide, ExtrudeCrv, Offset, Plane, ExtrudeSrf, DupBorder, Pyramid, Cylinder, Box, PlanarSrf, Pipe, ArrayCrv, Dim.
Geometry editing: Split, Trim, MakeHole, MergeAllFaces, BooleanDifference,
Workflow control: Move, Copy, CPlane, Explode, Select, Invert, Show, Hide, Lock & Unlock, Delete, Zoom.</td>
</tr>
</table>

Part II: Modeling workflows

Workflows in architecture are typically unique to each building, and are highly dependent on the people and requirements involved. The overall flow contains specific tasks. The following sections introduce a series of tutorials that are task-based. They are grouped into four sections: Setting up, conceptual design, detailed design and prototyping workflows.

2.1 Modeling setup

There are a few steps to take with each new project to ensure proper organization and settings of your environment:

1. **Create a folder** for your model and give a descriptive name. As the project grows, you may add new folders and subfolders. Also, make sure that files are named to show in desired order, and describe the content for quick reference.
2. **Create a new template** with appropriate document properties, options, layers and reference geometry.
3. Create your model as close to the **World origin** as possible.
4. **Generate scaffolding** or reference envelope, guides, and construction planes to match model orientation.
5. **Set up views** and snapshots for effective visualization.

2.1.1 Project template

A template is a Rhino model file you can use to store basic settings. Templates include all the information that is stored in a Rhino 3dm file: objects, blocks, layouts, grid settings, viewport layout, layers, units, tolerances, render settings, dimension settings, notes, and any setting in document properties. You can use the default templates that are installed with Rhino, or save your own templates to use for your future models. The standard templates that come with Rhino specify viewport layouts, units and default document settings. You can create custom templates using different document settings for render mesh, tolerances, color scheme, layers, lights, or even include geometry and notes.

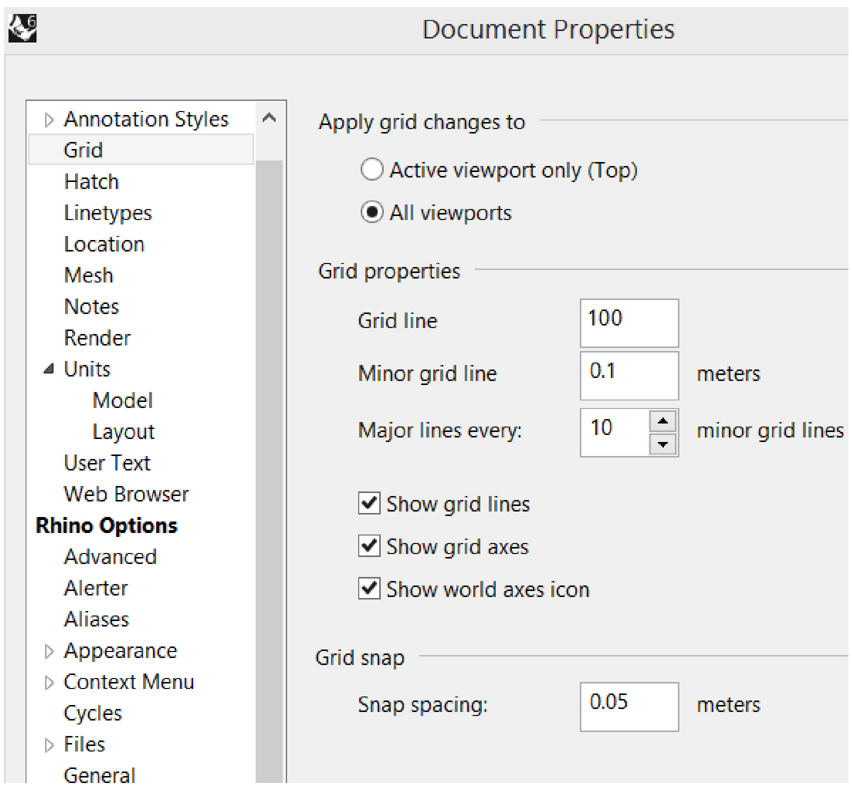
The ***New*** command begins a new model. It will use the default template unless you change it to one of the other templates. To change the template that opens by default when Rhino starts up, choose ***New*** and select the template file you would like to open when Rhino starts, then check the ***Use this file when Rhino starts*** box.

Steps to create a template:

1. Start a new model.
2. Select a Rhino template such as ***Large Objects - Meters.3dm***.
3. From the ***Render*** menu, click ***Current Renderer***, and then click ***Rhino Render***.

Steps to set the Document Properties

1. From the **_File_** menu, click **_Properties_**.
2. In the **_Document Properties_** dialog, on the **_Grid_** page, change the **_Grid line count_** to 100, **_Snap spacing_** to 0.05, the **_Minor grid lines_** every 0.1, and the **_Major lines_** to every 10.
3. On the Mesh page change the setting to **_Smooth and slower_**.
4. On the Rhino Render page, check **_Use lights on layers that are off_**.
5. On the Units menu, **_Model_** units, set **_Absolute tolerance_** to 0.001 and **_Angle tolerance_** to 0.5. Under **_Layout_**, set **_Layout units_** to Millimeter, and **_Absolute tolerance_** to 0.1, **_angle tolerance_** to 0.5.
6. For **_Location,_** change to **_Los Angeles, CA USA_**.
7. For **_Linetypes_** set **_Linetype scale_** to 100 mm.
8. Click **_OK_**.



Grid settings

Steps to set up the _Layers_ and _Properties_ panels

1. Open the **_Layers_** panel and rename **_Default_** to **_Reference_** and delete other layers.
2. Open **_Properties_** panel and set the camera **_Lens Length_** to 50 (35-50 is appropriate to architectural modeling).
3. Open **_Named CPlanes_** and **_Help_** panels and turn off all other panels

Steps to set save notes

From the **_File_** menu, click **_Save As Template_** and name House_Decimal_Meters_0.001.3dm. This file with all of its settings is now available any time you start a new model.

Steps to set the template just created as the default file Rhino opens with New:

1. From the ***File*** menu, click ***New***.
2. Select the template you want to use as the default template.
3. In the ***Template*** dialog, check the ***Use this file when Rhino starts***.

Resource 4: Templates
Template files in Rhino: https://vimeo.com/86730224

2.1.2 Productivity

Depending on the project type and your workflow, it might be a good idea to set up a quick access to your commonly used commands and options to improve workflow productivity. For example if you need to draw a lot of vertical lines, then it helps to write a macro to run the line command with the vertical option (***!_Line Vertical***) and add it as a keyboard shortcut or as a new command to the context menu or popup toolbar. For details about macros, check the wiki article Creating Macros and the Rhino help.

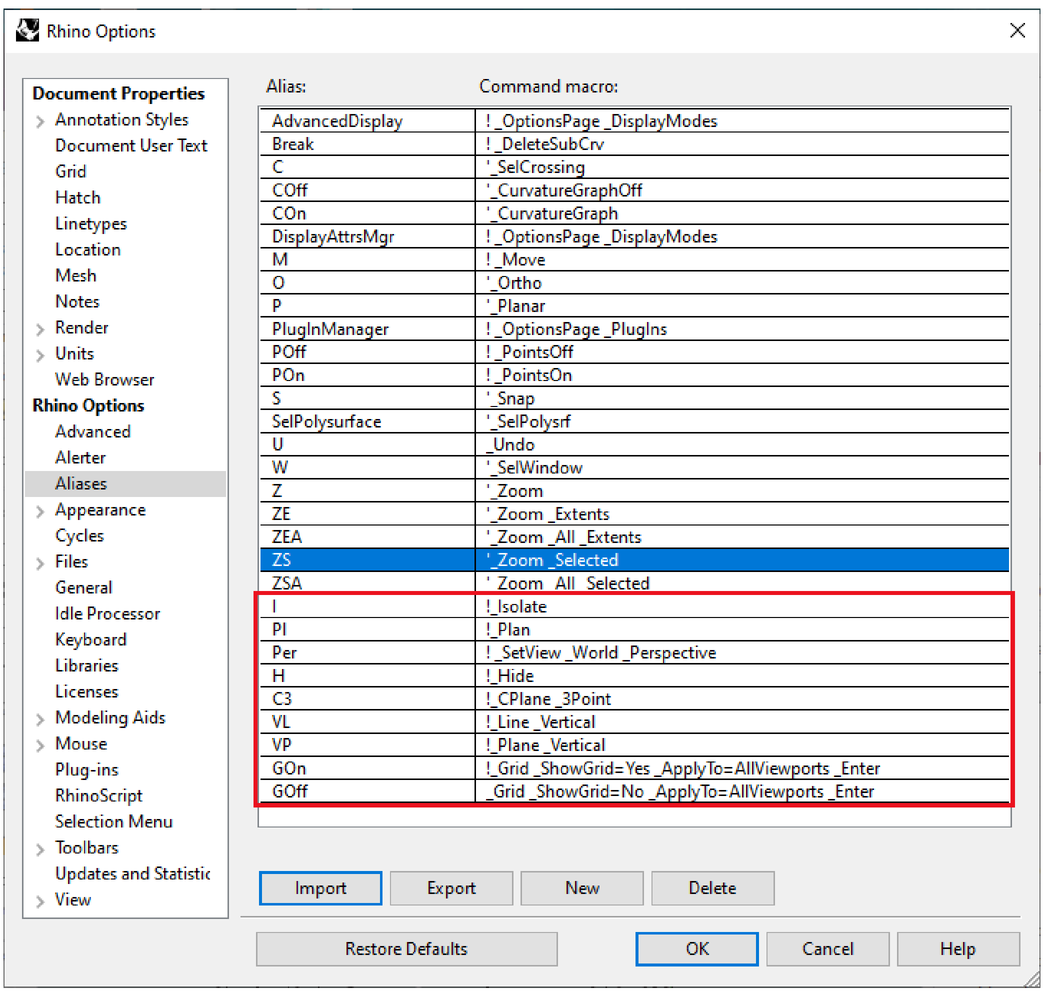
Macro character	**Meaning**	**Example**
(space)	All entries (command words and numerical inputs) need to be separated by a single space.	
Pause	Wait for user input	! Circle Pause 50
MultiPause	Wait for multiple input	! _Polyline _MultiPause
Enter	Simulate pressing “Enter” and is needed to end selection inside or to end commands	Polyline 1,1 1,9 6,9 Enter ;Same macro using relative coordinates to last point: Polyline 1,1 r0,8 r5,0 Enter
! (exclamation point)	Cancels the previous command. It is a good practice to use it before all macros (unless running nested command)	! SelAll
‘ (apostrophe)	Run as a nestable command (while running another command). Note not all commands are nestable.	!Move ‘ SelAll Enter MultiPause
_ (underscore)	Runs command as English command name This is a good practice in case you pass the macro to a non-English user	! _Circle _3Point 0,0,0 1,1,0 0,3,0
- (hyphen)	Bypass dialog box (scripted command mode)	! _SelLast -_Rebuild _PointCount=10 _Degree=3 _Enter

Most common macro notation in Rhino

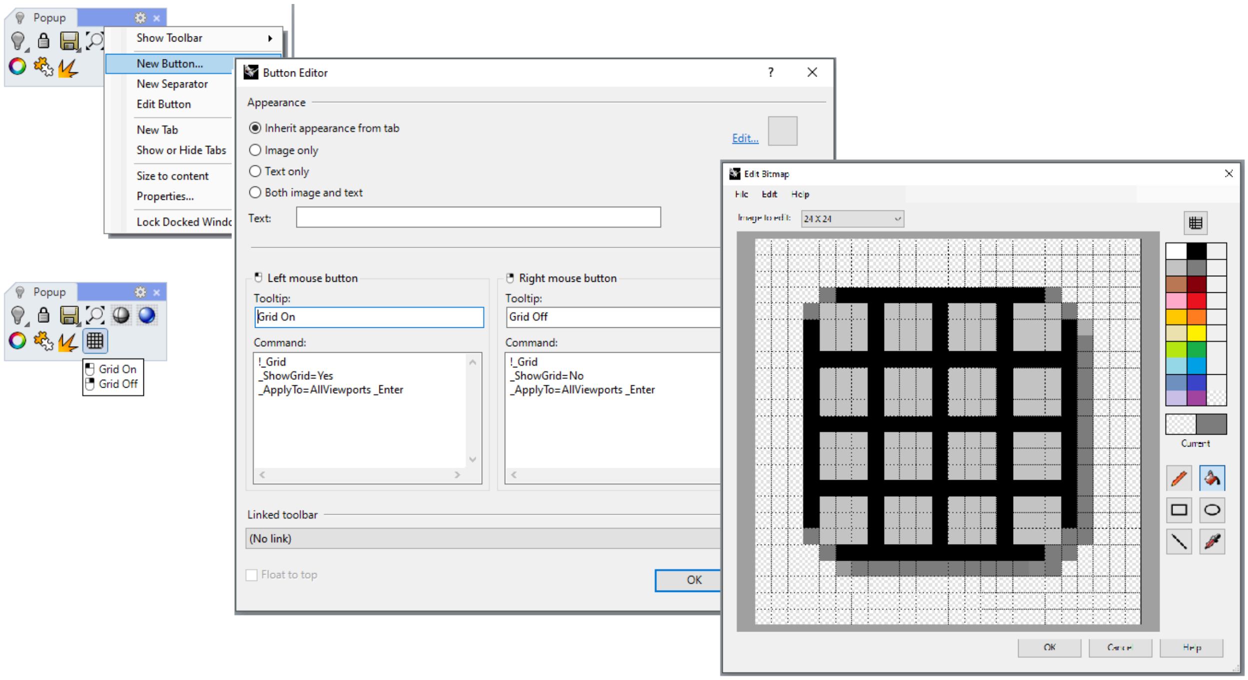
First make a list of commands and options to use and you can always change the list based on your workflow. For example, we might decide to create aliases for the following:

Command/options	Alias name	Macro
Zoom Selected	ZS (already in Rhino)	!_Zoom _Selected
Plan view / Perspective view	Pl / Per	!_Plan / ! _SetView _World _Perspective
Hide / Isolate	H / I	!_Hide / !_Isolate
CPlane 3Point	C3	!_CPlane _3Point
Vertical line	VL	!_Line _Vertical
Vertical plane	VP	!_Plane _Vertical
Grid on / Grid off	GOn / GOff	!_Grid _ShowGrid=Yes _ApplyTo=AllViewports _Enter / !_Grid _ShowGrid=No _ApplyTo=AllViewports _Enter

Here is how it looks when you add the above as Aliases (***ZS*** comes by default with Rhino)

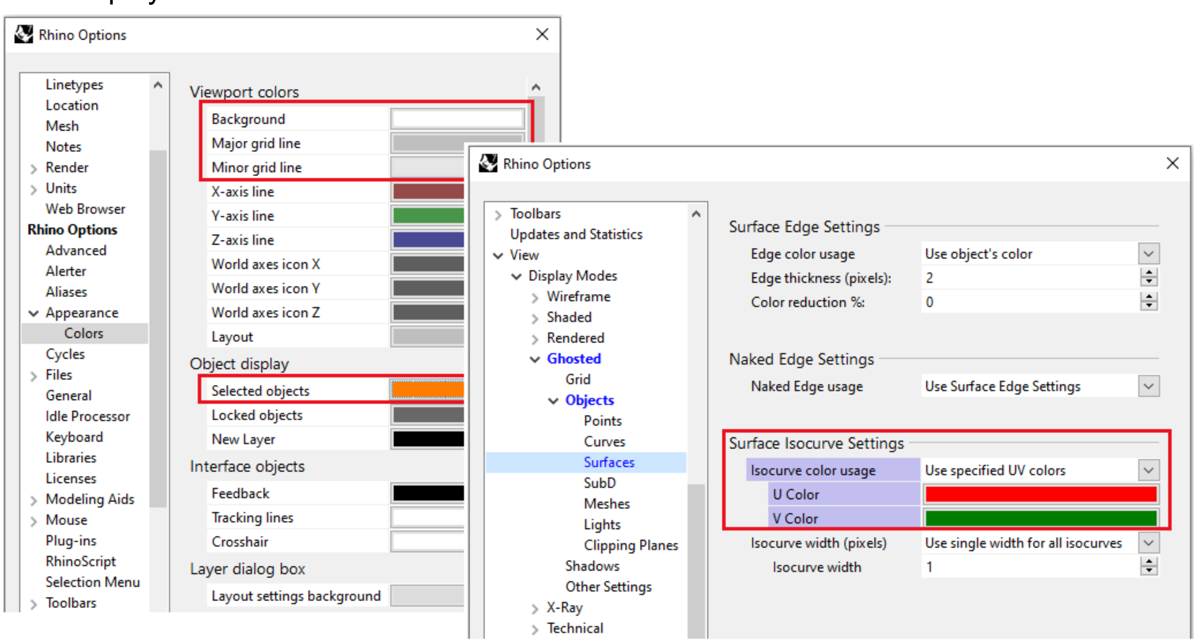


And also add a new custom toolbar button for the grid on/off to the popup toolbar.



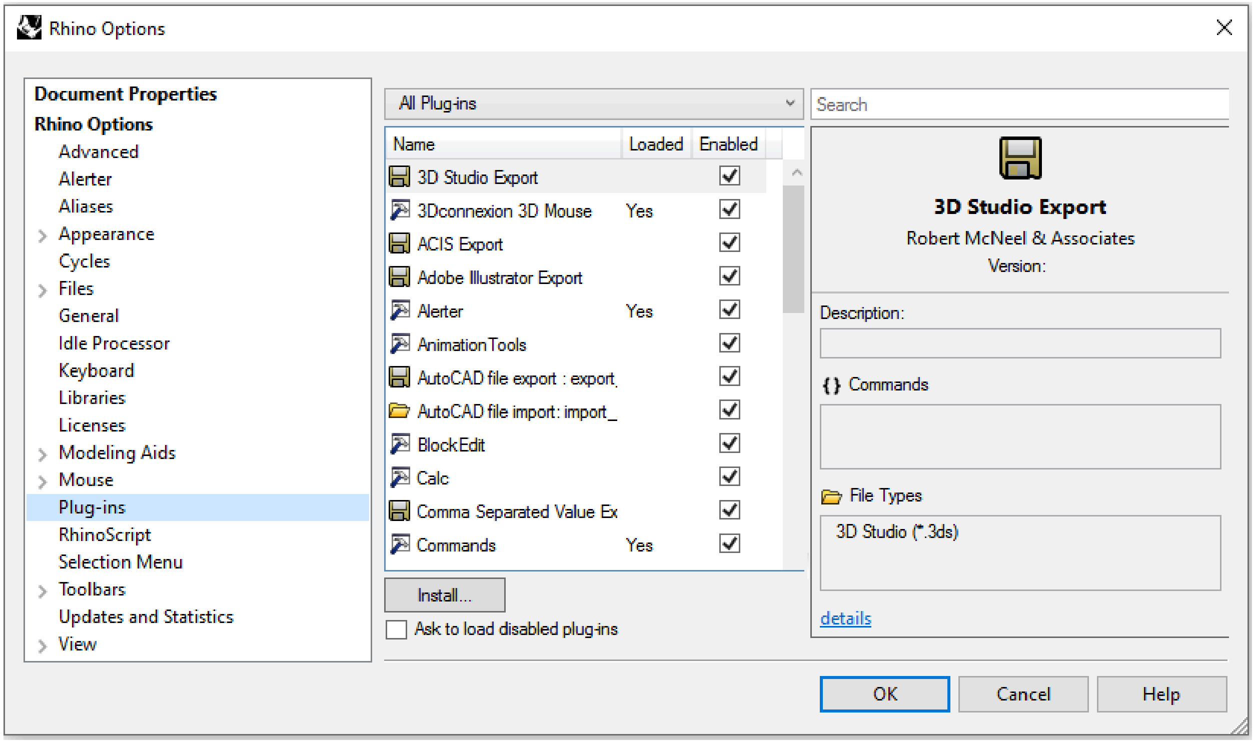
2.1.3 Appearance

You can also customize the color scheme and fonts in the ***Appearance*** and ***Colors*** under Rhino ***Options***. You can always restore defaults to revert to original settings. You can also customize the display modes to help productivity and presentation. One important setting when working with NURBS surfaces is to indicate the UV iso directions of surfaces and their normal direction by assigning distinct colors. You can also specify edge thicknesses and point sizes in each display mode.



2.1.4 Plugins

You may need to use plugins to access specialized features and workflows. Installed plugins are saved in a special folder and Rhino knows where to find them. Rhino ships with many plugins, especially those for file input/output. If you run the ***PluginManager*** command, you can see the list of plugins.



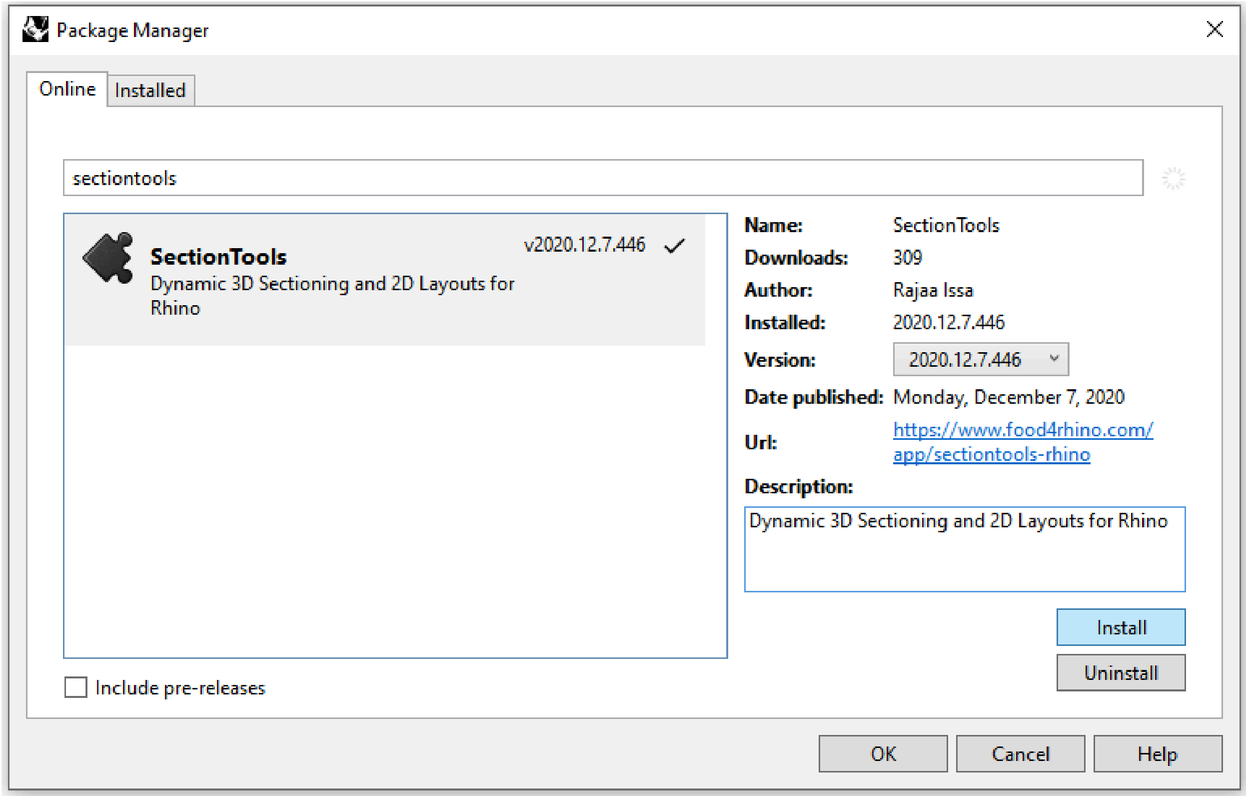
Plugin manager

Most third party plugins are available through the ***Package Manager*** in Rhino 7, or have their own installer. Rhino places installed plugins in the proper folder and loads when Rhino opens next time. Rhino also recognizes if there is a toolbar that comes with the plugin and allows loading it by the user. If there are plugins that you do not like Rhino to load, you can place load protection by un-checking the ***Enabled*** checkbox for these plugins. Next time you open Rhino, they will not load. You can always check the box again to permit loading.

There are two plugins by McNeel that we will use the do not come with the Rhino installation for Windows. Those are ***PanelingTools***[2] and ***SectionTools***. In order to install them, run the PackageManager command, search for each and click ***Install***. Restart Rhino and both plugins will load and you can see their menu at the top.

---

[2] PanelingTools for Rhino and Grasshopper ships with Rhino 7 for Mac but need to be installed from the PackageManager in Windows.

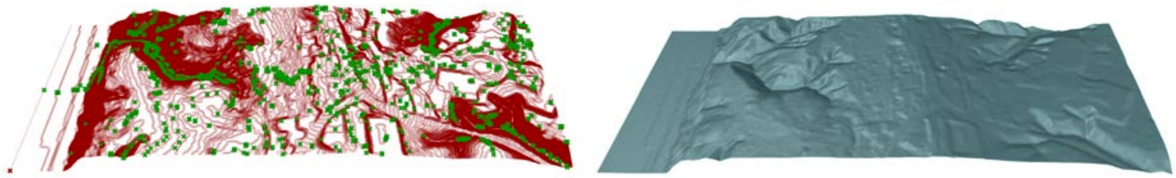


2.2 Concept modeling

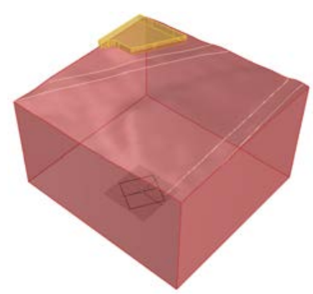
Concept generation is an uncertain and reflective stage of design. Designs go through multiple iterations before reaching the final solution. Most designers start their concepts with sketching on paper, then bring the sketches to the 3D modeling environment, model and adjust till researching a solution. It is a good practice to keep an organized record of the 3D modeling steps to be able to communicate and modify easily. A common workflow for concept design is to start with the site geometry, import sketches, generate the mass model, analyze, edit, and share. We will cover all of these aspects using step by step tutorials.

2.2.1 Site terrain and surroundings

Geographical information and land use is nowadays available in digital format with multiple layers of topographical and other information which helps bring site data directly to the modeling environment. There are many plugins for Rhino and Grasshopper to import site terrain and surroundings. Food4Rhino website is a place where most of the Apps for Rhino and Grasshopper are published and you can find relevant plugins using keywords such as ***topography***, ***GIS*** and ***site***. Another way to get terrains is to import the contours form a land survey and create the terrain surface inside Rhino using the contours as reference geometry. Appendix A shows a detailed workflow of how to import site data using two Grasshopper plugins: Elk and Meerkat.

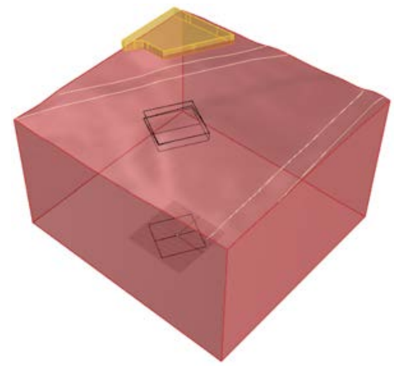
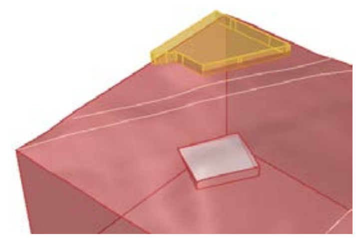
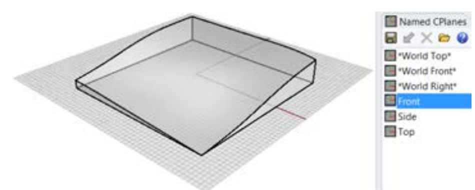
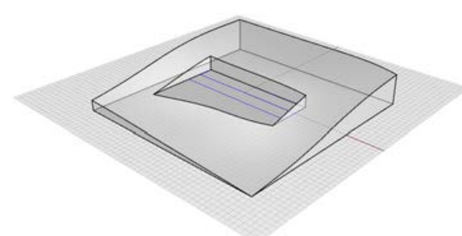


Import site GIS information

CM_01: Terrain	
Import GIS data for your location with terrain, roads and buildings information when possible. There are specialized plugins to import GIS data from surveys or images published in ***Food4Rhino.com***	
Place the imported terrain so that the center of the site is near the Rhino World origin (0,0,0). This is important to be able to model with accuracy and perform operations such as Booleans and intersections correctly within the model tolerances.	
Cut out smaller part of the terrain - Use the Box command that intersects with the terrain (make the box taller than the terrain to ensure that the Booleans operation can be calculated accurately). - BooleanSplit extruded box by the terrain. - Define site boundary around the world xy-plane origin. For this tutorial, draw a 40*40 m rectangle and rotate 30 degrees clockwise.	

You can take part of the terrain closer to the site to model the concept. The following workflow shows how to take part of the terrain and define site boundaries and flatten part of the site.

CM_02: Site

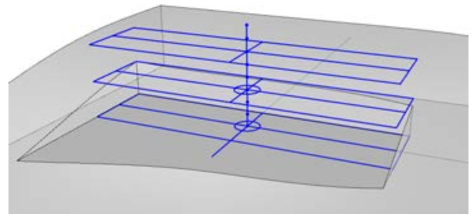
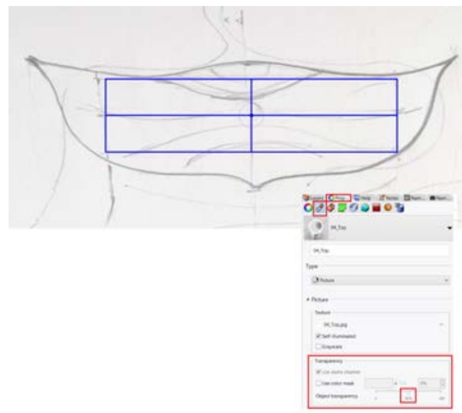
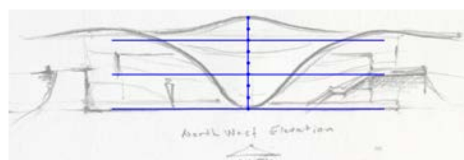
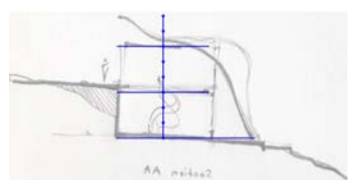
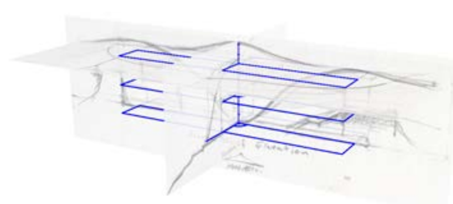
Site boundary and levels - [Project](#) the flat outline to the terrain. - Copy the flat outline to the bottom and lowest and highest points.	
Geometry of the site slope: - Extrude base curve using ***Gumball*** and Cap (or use ***Solid*** option in ExtrudeCrv command) - BooleanSplit extruded box by the terrain. - BooleanSplit the terrain by the extruded box. - Delete unwanted split parts	
Set CPlane to be at the center of the site and save three main CPlanes for quick reference. - CPlane > 3Point to set the CPlane to the Top, Front and Side relative to the site orientation - In ***Named CPlane*** panel, save the new construction planes and name ***Top***, ***Front*** and ***Side***	
Create the building base and flatten the site in the location where the building mass is located (building dimensions are 24*6*6) - Create the building base. Use Rectangle command with ***Center*** option, and set dimensions to 24 and 6 - Create a Box using the rectangle as reference and extend the side of the box facing the slope across the site so it intersects completely. - BooleanSplit to cut the site - Calculate the volume of the cut out part to estimate the cost. Use the Volume command (~420 cubic meters)	

2.2.2 Mass model

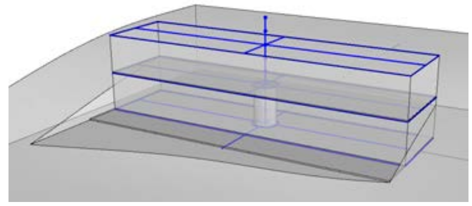
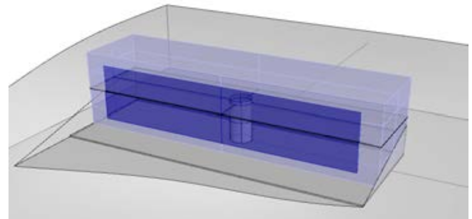
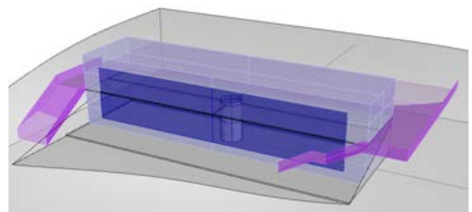
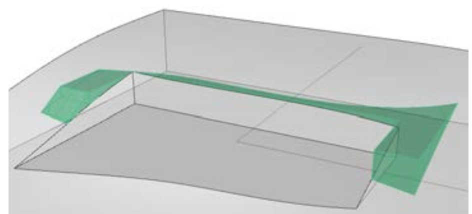
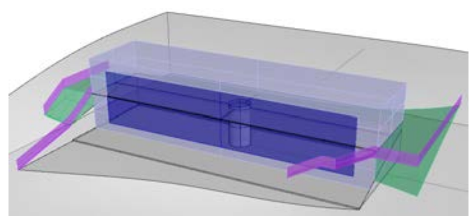
Generating building geometry involves creating scaffolding, placing sketches, and modeling the building's main geometry elements.

Scaffolding is very useful to define reference geometry to help create the building. This geometry helps define things such as orientation, dimensions, and levels. Scaffolding geometry can be used as a reference when creating the building geometry.

CM_03: Scaffolding
Scaffolding and sketches are used as a reference to create the building surfaces.

<table>
<tr>
<td>Create scaffolding for mass, levels and stairs:
- Make the Top CPlane active (double click Top CPlane in the list of Named CPlanes panel)
- Create a new layer called Scaffolding and sublayers called Mass and Sketches. Make the Mass layer current
- Rectangle, Center option (24*6) at origin of Top CPlane
- Draw Lines through the rectangle center
- Draw Line > Vertical from center with 8m length
- Divide the vertical line into 8 segments to create reference points that are spaced by 1m
- Copy base rectangle and lines vertically to level 3 and 6 m
- Create Line at the base 3 m long
- Circle (radius=1) on lower and middle levels for staircase</td>
<td></td>
</tr>
<tr>
<td>Place the top sketch
- Use Picture to place top sketch
- Use Plan to align parallel to active CPlane
- Use Gumball to locate and scale the picture to fit the mass dimensions
- Change transparency to 30%:
    - Select the picture
    - Go to Properties Panel > Materials > Picture > transparency and set to 30%
- Use Lock command to lock the sketch in place</td>
<td></td>
</tr>
<tr>
<td>Place the elevation sketch
- Create new CPlane that aligns with the model front view through the model origin and save in Named CPlanes and call Front
- Place elevation Picture and Move/Scale to fit dimensions
- Change transparency to 30% (same steps for top)</td>
<td></td>
</tr>
<tr>
<td>Place the section sketch
- Create a new Side CPlane and save in Named CPlanes
- Place section Picture and Move/Scale to fit dimensions
- Change transparency to 30% (same steps for top)</td>
<td></td>
</tr>
<tr>
<td>Go back to the perspective view to verify that all of the sketches are oriented and located correctly and to scale</td>
<td></td>
</tr>
</table>

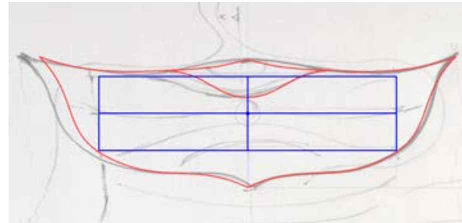
CM_04: Massing
Massing involves creating base geometry for the mass model with minimum details.

Create initial massing geometry - Switch to ***Top*** CPlane (in Named CPlanes panel) - Create a new ***Building*** layer and a ***Floors*** sublayer, and make it the current layer - Box to create first and second floor slabs (~0.2 m) - ExtrudeCrv base curve and put in a new ***Walls*** layer - Cylinder to place staircase mass, put in ***Stairs*** layer	
Add openings. - Switch to ***Front*** Cplane - Create openings and place in a new ***Windows*** layer: draw a rectangle on the side face where the openings are to be created, use Split command, then select the split portion of the face and change the layer to ***Windows***. - Calculate Area of walls and windows (e.g. windows area = ~140 sqm, walls = ~220 sq m)	
Add front and side platforms/balconies - Draw curves, ExtrudeCrv, then Join and MergeAllFaces - Use ortho angle = 30 to define stairs' slope (30-37 degrees is the recommended stairs slope)	
Generate fill - ExtrudeSrf platform surface in the -Z direction, then BooleanSplit with the site to generate the fill solid - Volume = ~54 cubic meter for the fill	
Clean up - Trim rails and platform surfaces	

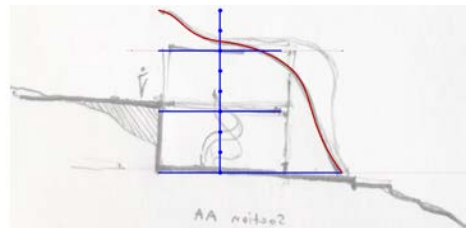
CM_05: Roof
The design uses a free form for the roof. Use reference sketches to create main curves of the roof in each picture plane, then edit the curves to place correctly in 3D space. Use the curves to generate the roof surface.

Trace roof outline in the top picture. Snap to other curves and scaffolding when relevant.

- Move the ***Top CPlane*** to align with the top picture
- Use the ***Curve*** command to trace one side
- Use ProjectToCPlane to ensure curves on plane (snapping can cause control points to go off the plane)
- Use ***Mirror*** to reflect the curves and ensure symmetry



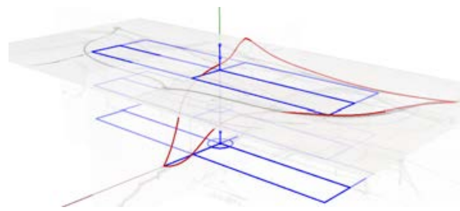
Repeat the tracing process for the section sketch

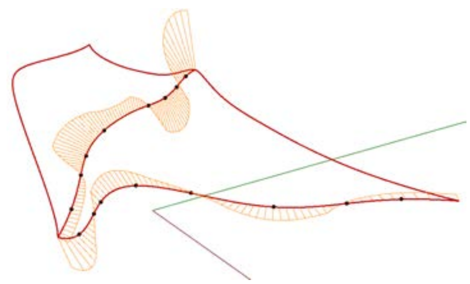


Adjust top curves to match elevation contour

- Need to adjust only half the curves, then mirror them
- Set CPlane to ***Front***.
- Set selection filter to ***ControPoints*** only.
- ***Move the*** points ***Vertically***
- Adjust using the ***Side*** view as well
- ***Mirror*** curves

Test curves' smoothness with curvature graph for curves. The points mark significant locations on the curve where curvature changes from concave convex, or the middle of a long span.





Roof surface - ***Patch***
Having boundary curves and cross section curves might be enough to create a good patch surface.

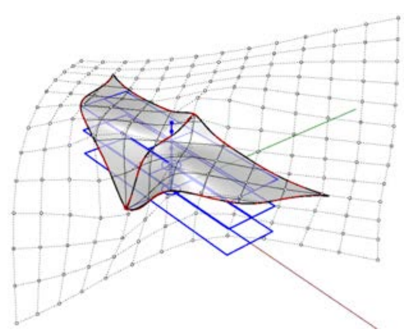
- Use ***Patch*** command and select roof 3D curves. Use the ***Preview*** option to see how the patch surface looks and adjust settings
- ***Pull*** curves to surface
- ***Trim*** surface with pulled curves

Notice that the surface parameterization (direction of isocurves) may not be in a desired direction. Also the surface is not symmetrical.

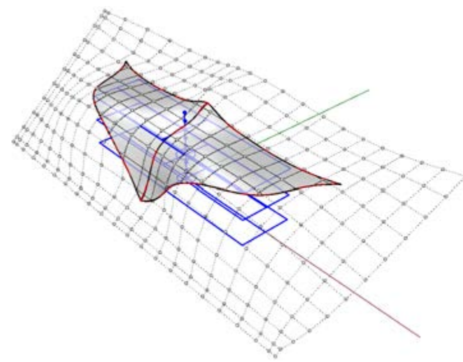
Repeat the ***Patch*** command with a starting surface.

Create a tilted ***Plane*** through the three corner points and make the plane slightly bigger. Use that plane as a starting surface in Patch command the repeat steps above to create the patch surface.

Notice that parameterization followed the plane direction. Also when comparing the mean curvature of the two patches, the second comes out smoother (notice the transition from blue to red).



Notice that both surfaces are trimmed surfaces.

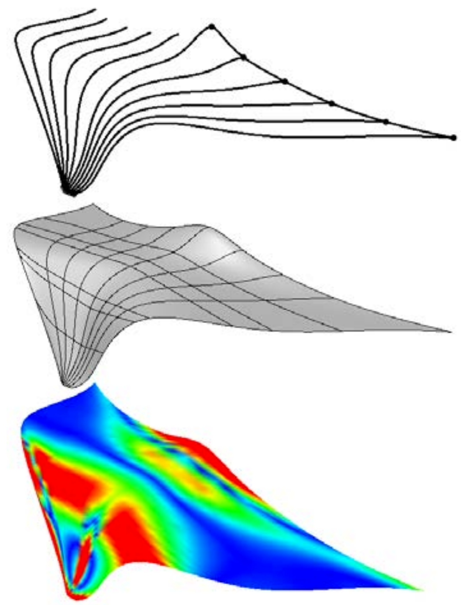


Roof surface - ***Loft*** surface

Loft surfaces are typically easy to set up, construction curves are in one direction and are easy to adjust and match. History also allows more interactive editing until reaching a desirable solution.

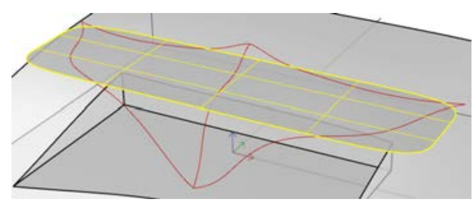
- ***Divide*** the front curve into 5 segments.
- ***TweenCurves*** between middle and end, then adjust curve ends to align with the points. Use ***PointsOn*** command to move control points. Alternatively, and if you have a temporary surface from patch use ***Project*** to locate and trim tween curves on the surface.
- ***Mirror*** tween curves.
- ***Loft*** curves with ***History*** on to be able to adjust the surface as needed. Turn on history recording for a specific command by clicking on the ***Record History*** in the ***Status Bar*** before running the command.

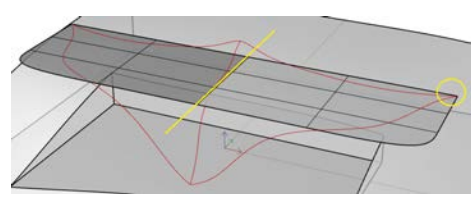
- Test mean curvature using ***CurvatureAnalysis***

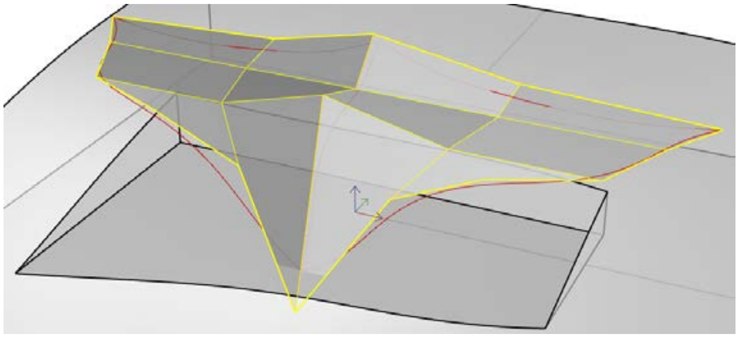
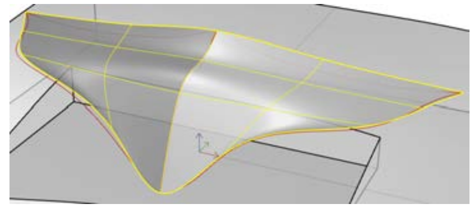


Roof surface - ***SubD*** surface.
- Create ***SubDPlane*** from center and set X=4, Y=3
- Use ***Reflect*** before editing to apply symmetry.
- Use ***Crease*** and select the end vertix to get a sharp corner
- Sub-object select curves and vertices of the subD surface (use Shift+Ctrl down), then move to create the roof form using the 3D curves as a guide. You only need to work on one side.
- You can convert the subD surface into NURBS surface using the ***ToNURBS*** command.

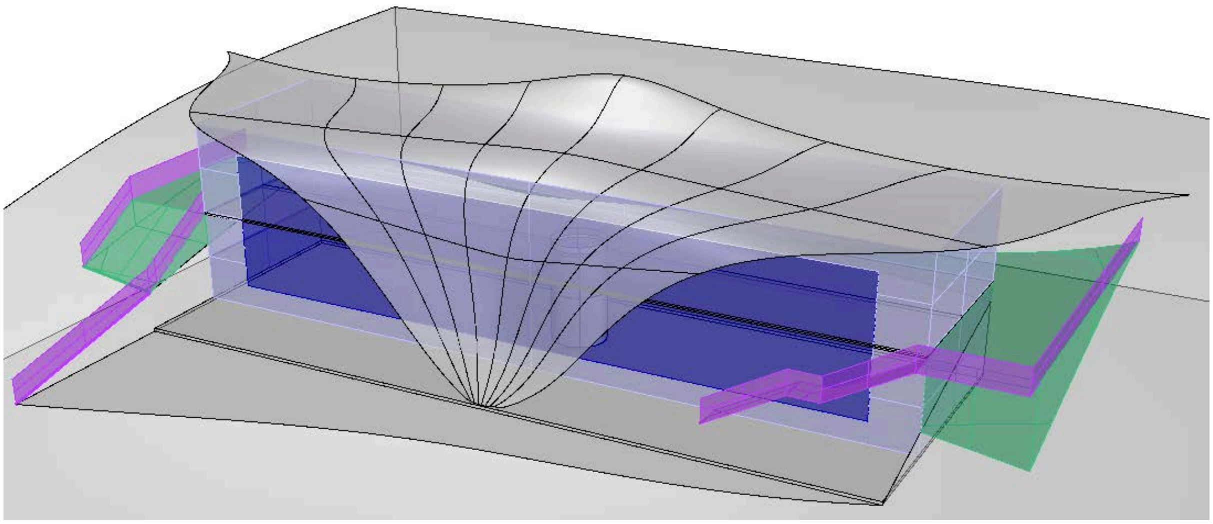
<u>Tip</u>: Use Tab key to toggle between ***Box*** and ***Smooth*** modes of the SubD preview. It is easier to pick vertices in Box mode, but you can only see the final smooth surface in Smooth mode.





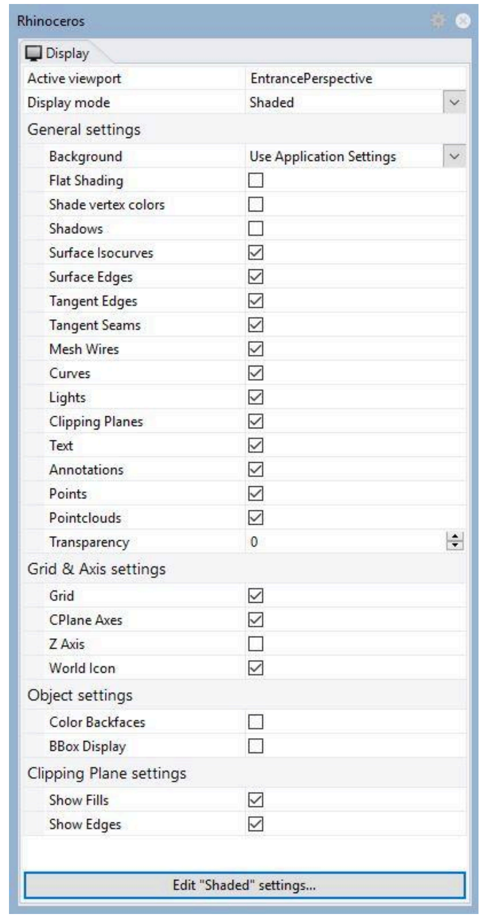
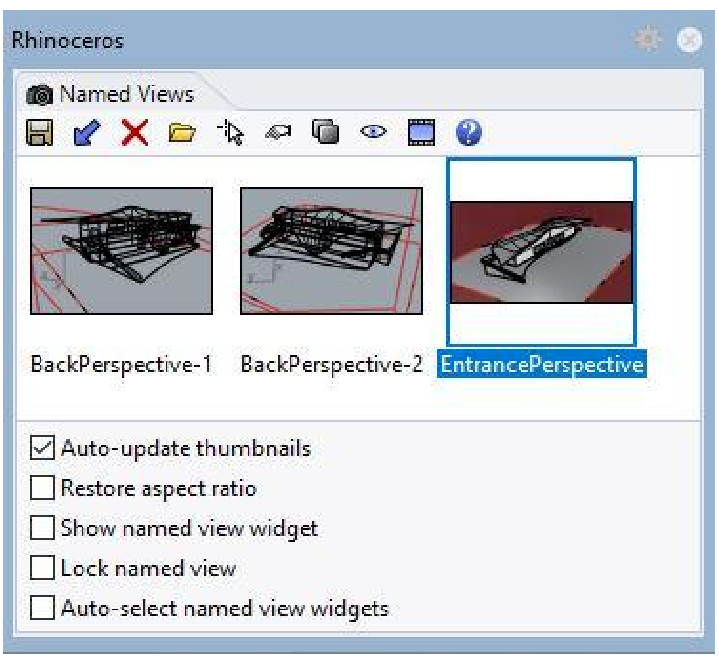
The concept geometry putting the mass and roof together:



2.2.3 Concept visualization

Visualization is an essential part of concept development and communication. Rhino has a rich set of tools for quick visualization, but also to produce high resolution renderings. Viewport display modes offer quick access to a variety of display modes. They all are working modes where you can continue to edit and navigate your model. There are also tools to assign materials, create renderings and capture images that we will explore in the next tutorial.

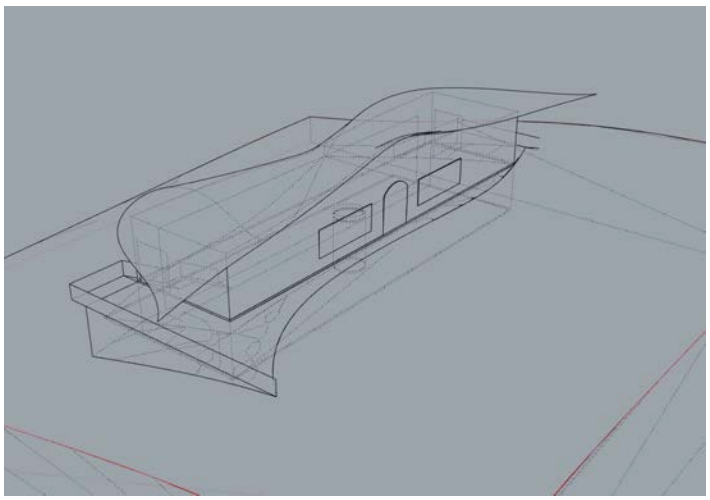
CM_06: Visualization

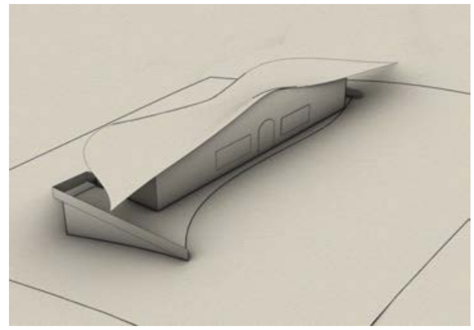
Display modes: A variety of built in display modes are included with Rhino and these are accessible in the Display panel which can be found in the Panels drop down menu if not already shown. Many commonly changed settings can be altered within this panel. Click the button at the bottom of the display panel to open the full options for the mode chosen. Display mode settings are system specific and are not part of the 3dm file. To export and import customized display modes, use the Options command and navigate to View>Display Modes. Any existing display mode can be copied as the starting point for a new user-defined display mode.	
	Named Views: You can save your views in the Named Views panel You can show the Named Views panel with options including to save the CPlane. Once saved, you can restore at any point during modeling. Double click any thumbnail image in the Named Views panel to restore it in the active viewport. Use the icons at the top of the panel to access various named views functionality. Among these controls you can import named views from another 3dm, edit a named view by simply rotating the viewport as well as animate the transition between named views.

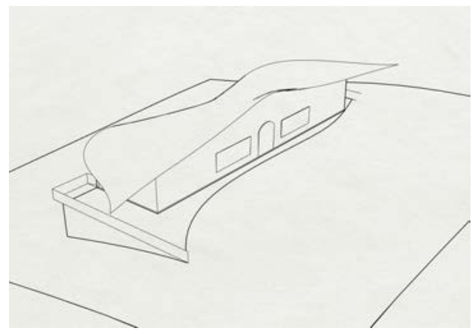
Display mode features:

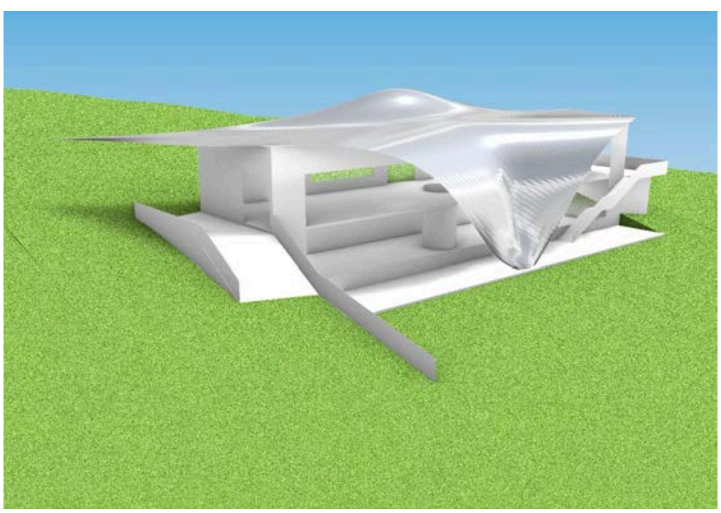
Some similarities can be found between certain default display modes and these are good to understand as you experiment with customization.

Technical, Artistic and Pen are all derivatives of the Technical display mode. These modes require an initial calculation time dependent upon the complexity of the scene. Hidden lines and Silhouettes can be visualized in these modes making them unique.

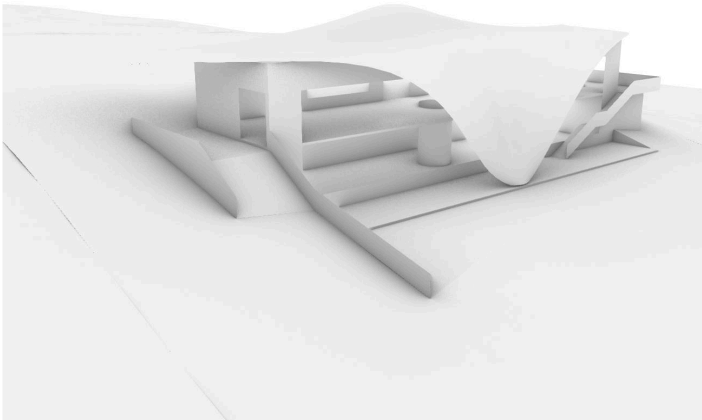
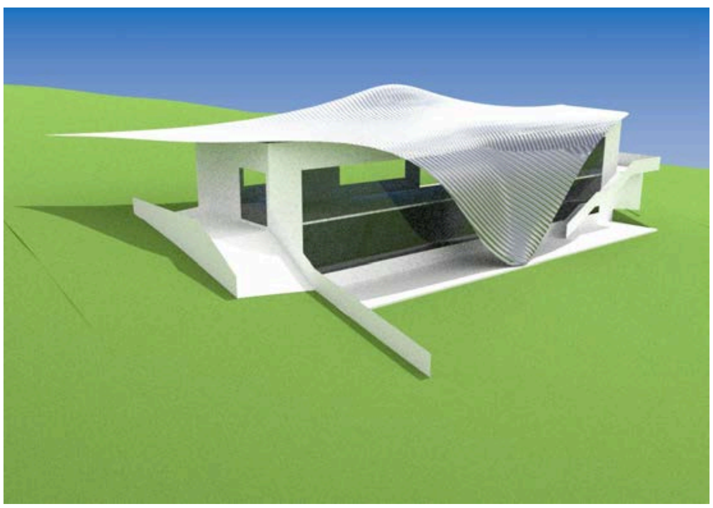
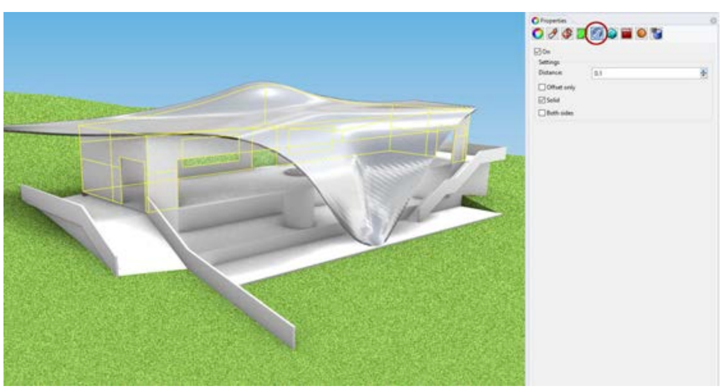




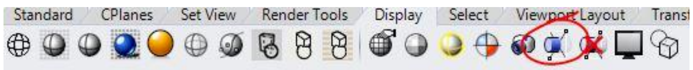




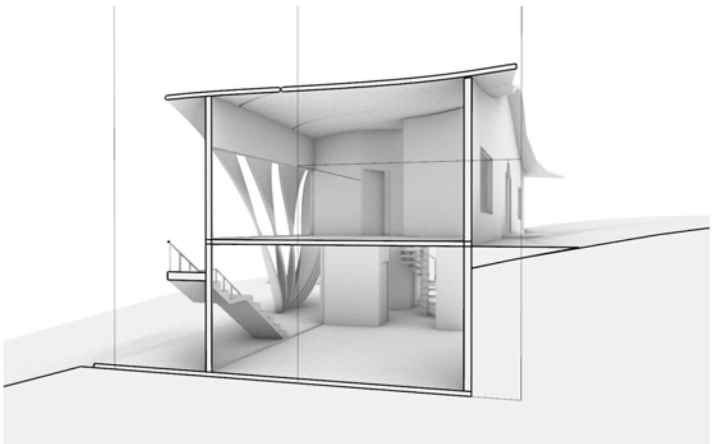
Rendered mode uses many of the settings found in the Rendering panel which can be shown via the Panels drop down menu if needed. Namely, the display of assigned materials and cast shadows distinguish this mode from a standard Shaded display.

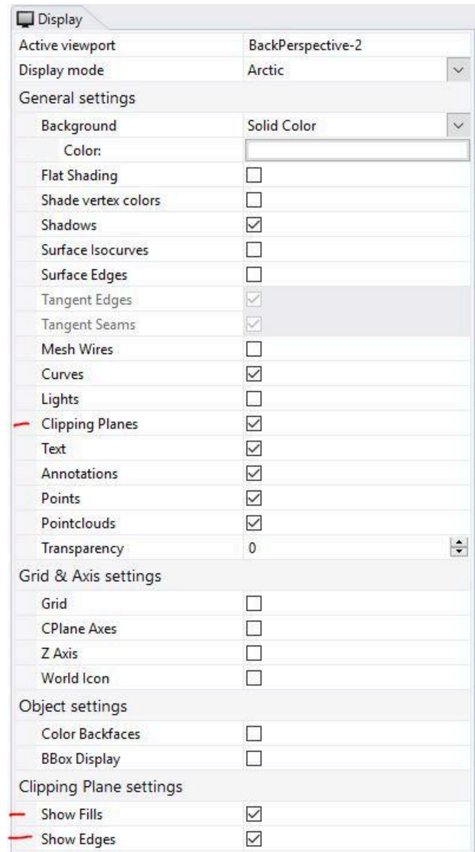
	**Arctic** mode will override any material assignments and custom lighting to produce a soft shadowed grayscale display mode. Materials remain in the file and are still assigned. By default, transparent materials such as glass will remain transparent in the Arctic mode. This option is configurable within the display mode settings.
	**Raytraced** mode is unique among the display modes in Rhino. This mode actually renders the model interactively in the viewport. Reflections between objects as well as indirect illumination (the bouncing of light) are calculated.
	Render mesh modifiers can be used to change the look of the model without impacting the actual geometry. One example is the thickness modifier for surfaces which when enabled will make any surfaces appear as solid (with thickness) in shaded modes. It is accessed when selecting surfaces, then in the ***Properties*** panel, select the ***Thickness*** button.

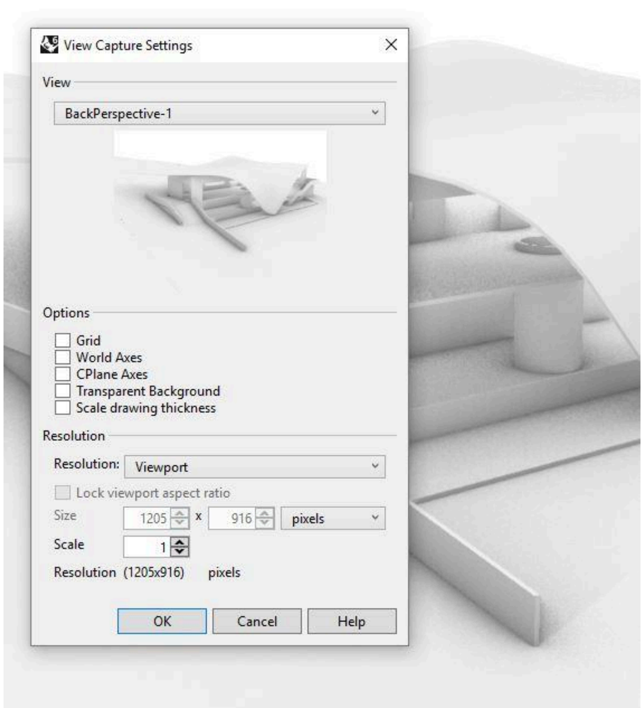
Another feature that can be quite helpful in concept visualization is the use of Clipping Planes. You can create new clipping planes with the ***ClippingPlane*** command and enable at any the active viewport using ***EnableClippingPlane*** command.



Multiple clipping planes can be active at any viewport. Select the clipping plane object, and in the ***Properties*** panel you can see which viewports are affected. Other clipping plane options such as showing thickness for edges or adding a solid fill can be set inside each display mode.





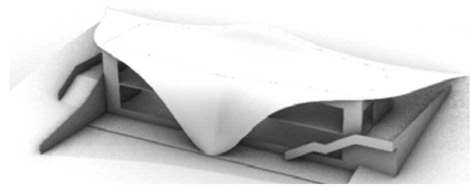
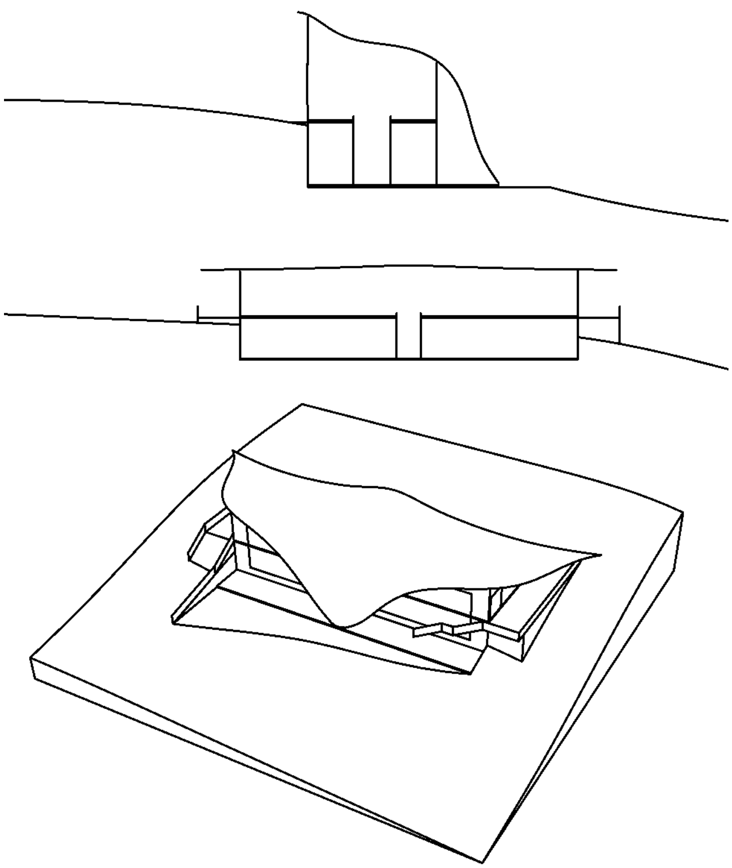


To save a high resolution image of the viewport in any display mode, you can use the commands ViewCaptureToFile or ViewCaptureToClipboard. Both commands offer additional options..

The view capture allows setting custom resolution for printing or other presentation purposes. Note that scaled Raytraced view captures will require a re-render of the viewport to account for the added pixels. Other modes, however, will scale up during capture relatively instantly.

2.2.4 Concept analysis

Analysis helps evaluate the design option and develop the concept. We have done some analysis above for the roof surface continuity, the overall volume of cut and fill in the site and building orientation. Rhino supports many workflows for concept analysis and the following includes some of those.

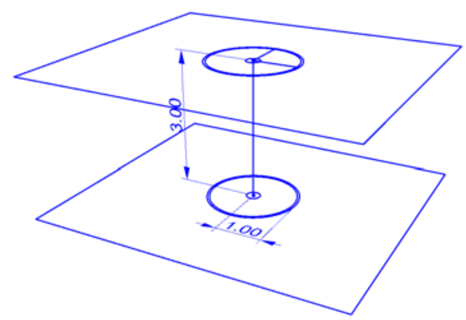
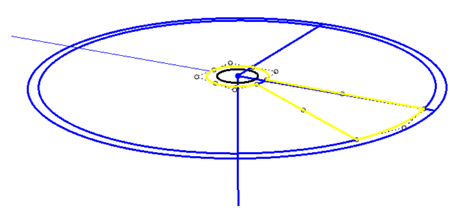
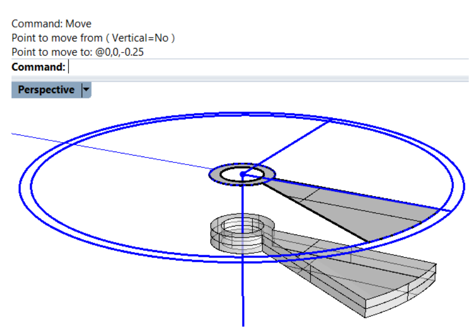
CM_07: Concept analysis	
Shadows study - Use SetOneDaySunAnimation, then PlayAnimation to view and RecordAnimation to save.	
Basic scheduling: - Make sure you create separate surfaces and solids for the walls, floors, glass, outdoor pavements, fills, etc. - Add data (name and user data) in the object properties panel to calculate overall area or volume for example.	
	Basic line drawings - Use Section on Top view to create sections - Use Make2D of the current view. Note that Make2D recognizes clipping planes and you can extract things like section perspective

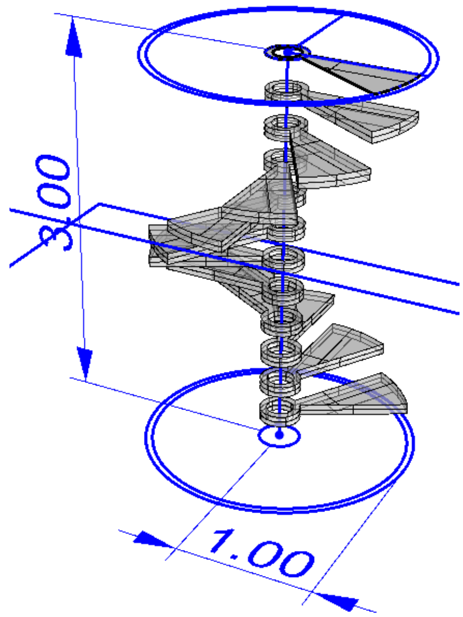
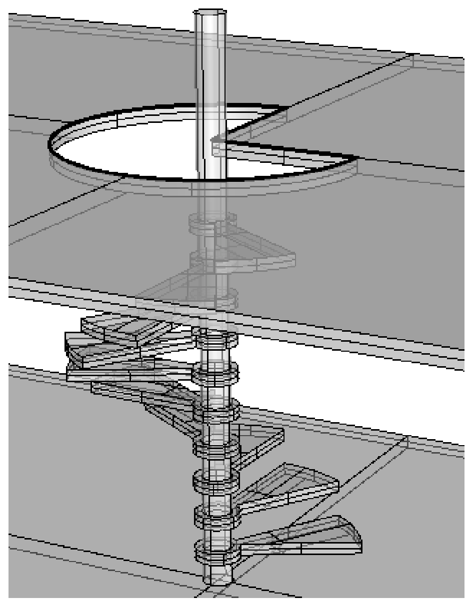
2.3 Detailed modeling

In this section, we will discuss workflows past concept stages including advanced geometry, rationalization, visualization and documentation. We will also discuss some parametric features in Rhino such as blocks.

2.3.1 Building details

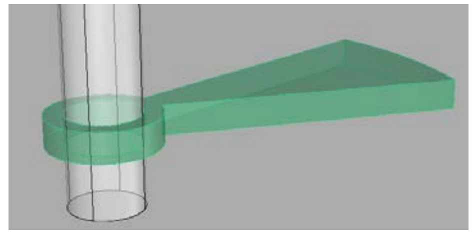
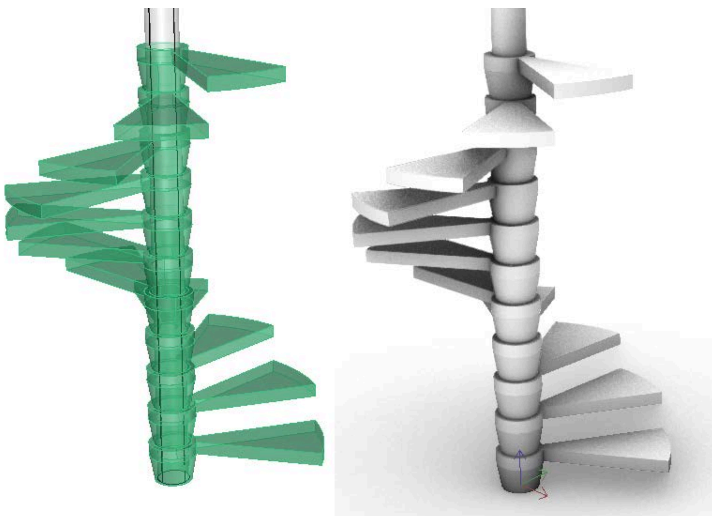
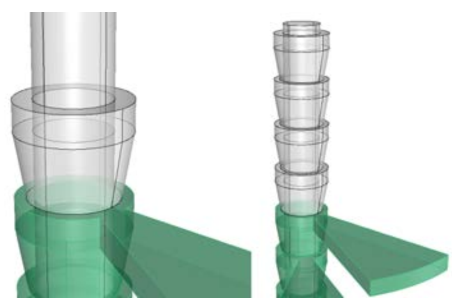
Building details include circulation, layout, materials, structure and floor plans. This work involves accurate modeling and considerations of building standards and material specification.

<table>
<tr><th colspan="2">DM_01: Spiral staircase</th></tr>
<tr><td>Create the stairs’ scaffolding with the desired dimensions.
- Locate the staircase center to be on top of the lower floor where it intersects the staircase vertical centerline. Adjust the CPlanes origins to be at that point parallel to the floor
- Create 3 circles with radiuses equal 0.1, 0.15, and 0.95. This makes each step width equals 0.8m.
- Use Distance and Dim commands to measure and mark the dimensions.</td><td>
</td></tr>
<tr><td>Create the curves for one step using the scaffolding
- Draw the first step outer Arc. Calculate the angle to be 360 divided by the number of steps per full rotation. Assume you have 12 steps, then the angle =arc degrees.
- Draw lines connecting endpoints of the arc with the center
- Create step outline: Trim then Join step curves</td><td></td></tr>
<tr><td>Create step solid
- Use PlanarSrf to create the step surface
- ExtrudeSrf by 0.08 in the negative vertical direction to create the step solid
- Calculate step rise: floor to floor height is 3m and number of steps is 12 steps, so the rise is 0.25m
- Move the first step up by 0.25</td><td>
</td></tr>
</table>

Create all steps. - [ArrayPolar](#) command with 12 steps StepAngle=30, set offset option ***ZOffset=0.25*** to create all the steps	
Add staircase pole and upper floor slab - Use [Cylinder](#) with radius = 0.1 - Fill the quarter circle of the top floor that meets with the stairs and ***BooleanUnion*** with the floor slab.	

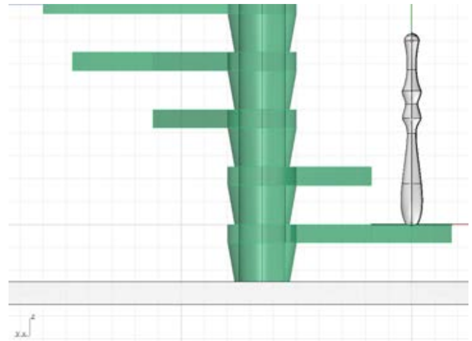
Models that involve copying the same geometry a number of times, such as the steps in the staircase tutorial, are best modeled as blocks. This helps keep the model size smaller and allows you to change the geometry, and update without remodeling. The following tutorial creates the staircase using blocks.

DM_02: Blocks

Create a block - Select the step geometry, then run Block command to create a new block. The selected step will turn into a block instant. - Run What command to confirm that it is a block instant.	
Edit the block - Create additional geometry to add to the step block - Run BlockEdit to add the new geometry to the step block	
	Recreate the rest of the steps in the previous tutorial (run ArrayPolar with Z offset by 0.25)
Create a new block to finish the rest of the pole - Create a new block - Array (or Copy) the block vertically to cover the whole pole.	

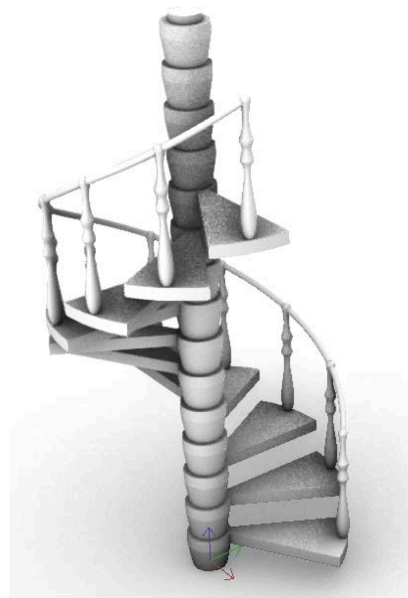
Create the spindles

- Change CPlane to ***Front***, then bring origin to the spindle center (CPlane, and set origin point).
- Change to Plan view
- Create profile curve
- Use ***RevolveCrv*** to create the surface
- ***Cap*** to create the solid
- Create a block from the revolved surface an a reference point for the handrail
- Change CPlane to ***Top***, locate one spindle on the first step, then run ***ArrayPolar*** with the same options as the steps.



Add the handrail

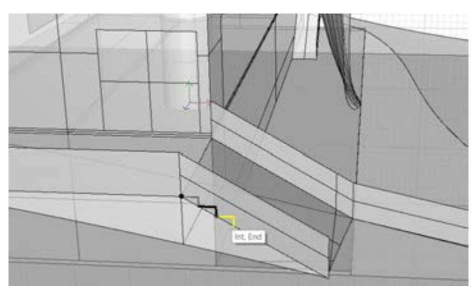
- Use ***InterpCrv*** to connect the reference points in the spindles.
- ***Pipe*** through the curve for a circular cross section (r=0.02), or ***Sweep*** for custom cross section.
- ***Cap*** the ends of the pipe

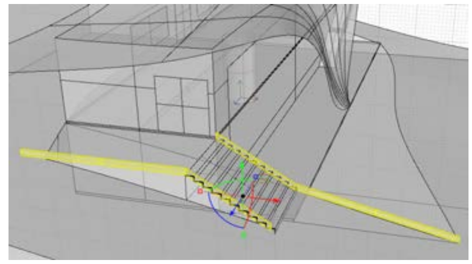
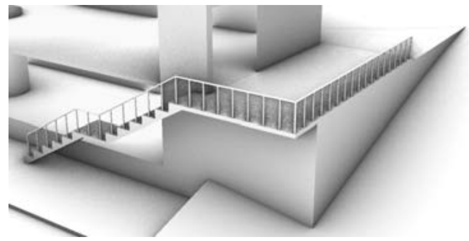
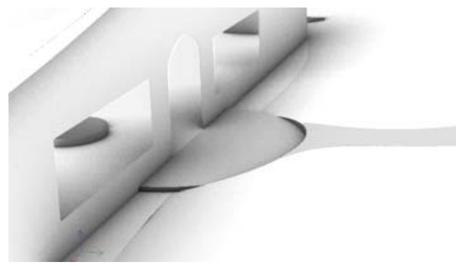


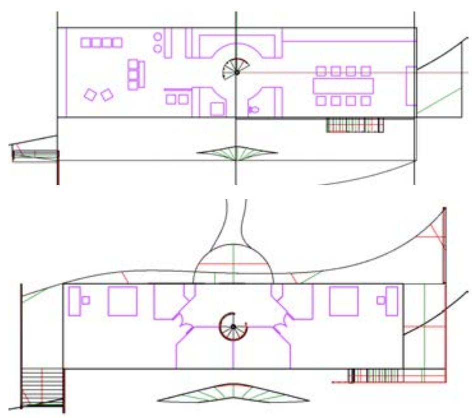
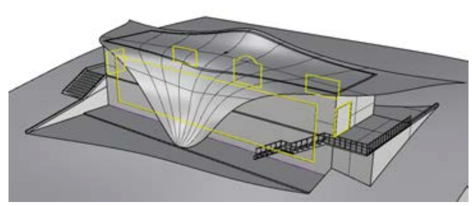
DM_03: Outdoor details

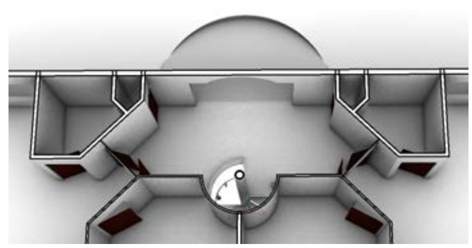
Create outdoor stairs

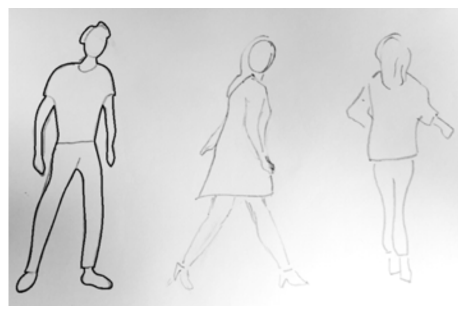
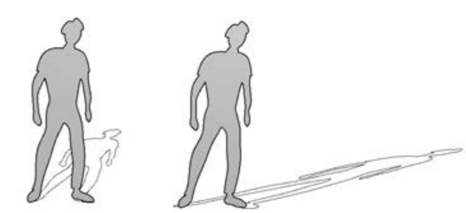
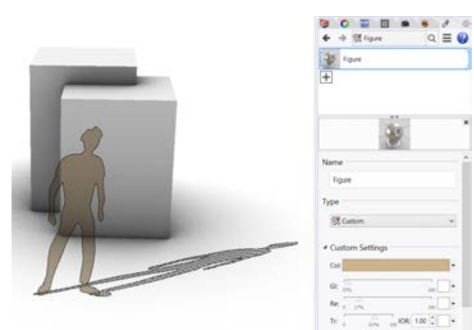
- Change CPlane to align with the stairs profile
- Use ***Polyline*** to draw one step
- ***Copy*** to create all the steps
- ***Join*** all polylines
- ExtrudeCrv to create the steps



Create stringer - Adjust stringer surfaces to appropriate height - ExtrudeSrf to create the thickness	
Create rails - Create a ***Cylinder*** on the first step, then turn into a block. - ***Copy*** and Array to create the rails. - Use ***Polyline*** to create the curve for the handrail - ***Pipe*** through the polyline to create the handrail	
Create entrance surfaces - Draw entrance shape using ***Arc*** and ***Line*** then ***Join*** - ***ExtrudeCrv*** with ***Solid*** option to intersect with terrain - Create the pavement path and project to the terrain surface - Copy terrain surface in place, then trim with pavement curve to create the pavement surface	

DM_04: Floor layout	
Sketch floor layout - Set CPlane to each floor - Use ***ClippingPlane*** to be able to view and work on each level. - Use ***Plan*** view - If you need to use other viewports, use ***SynchronizeCPlanes*** - Draw the general floor layout using curves	
Add exterior walls and windows - Use ***Slab*** command to create the exterior wall. Make it high enough to intersect the roof, and ***Trim*** with the roof surface. Put the walls in new ***Walls*** sublayer - Align the CPlane with each wall, draw the openings and use ***MakeHole*** to create holes in the wall slabs - Move the openings outline curves inwards, and give thickness and put in new ***Windows*** sublayer	

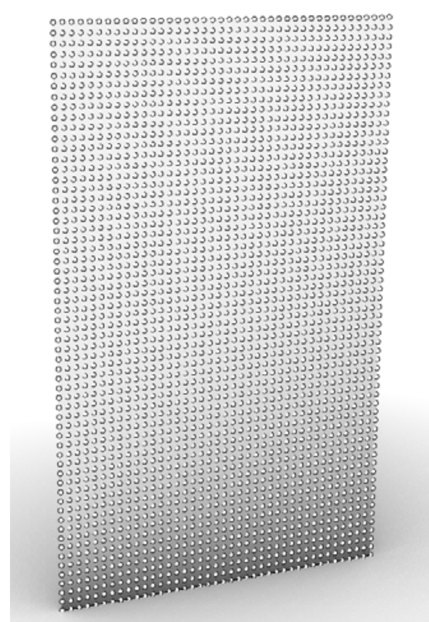
Add interior partitions - Use ***Slab, Offset and ExtrudeCrv*** commands to create interior walls from layout and put in new ***Partitions*** sublayer - Trim walls with the roof surface using ***BooleanSplit***	
Add doors. The general steps are: - Align CPlane to the wall, draw the door rectangle, then use ***MakeHole*** command to create the door openings. - ***Offset*** all sides of the rectangle, except the bottom, then close the curve to create the frame outline. Then ***ExtrudeCrv*** into a solid. - Use ***PlanarSrf*** on the original rectangle and move to the center of the frame to create the door geometry. Put the frames and doors in new sublayer	

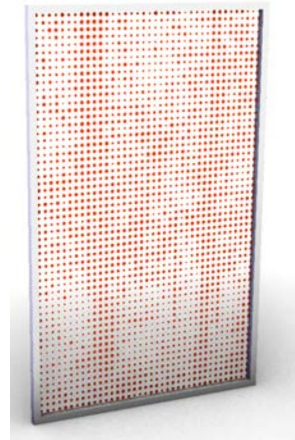
DM_05: Figures	
Create figures outlines in Rhino - Insert the figures sketch using [Picture](#)command - Lock the image (use the [Lock](#) command) and you may need to turn off GridSnap - Trace the figures using the control point curve command ([InterpCrv](#) or [Curve](#) commands). - If you end a curve and need to continue from where you left, then use ***ContinueInterpolateCrv***. - Make sure you create a valid closed curve (check with ***What*** command).	
- Copy and rotate the figure (use alt + rotate 90° on the gumball widget). - Select the vertical figure and create a surface (use ***PlanarSrf*** command) - Scale the horizontal figure in one direction mimicking the sunset long shadow (use ***Shear*** or ***Scale1D***).	
- Create a ***Hatch*** inside the horizontal curve - Choose your prefered Hatch pattern and rotate it at 90° using the pattern rotation in the Hatch dialogue. - Hide your curves - Add transparency to your figure (use the material Editor). You can also place all figures in a new layer and assign material and transparency as a layer property.	

2.3.2 Rationalization

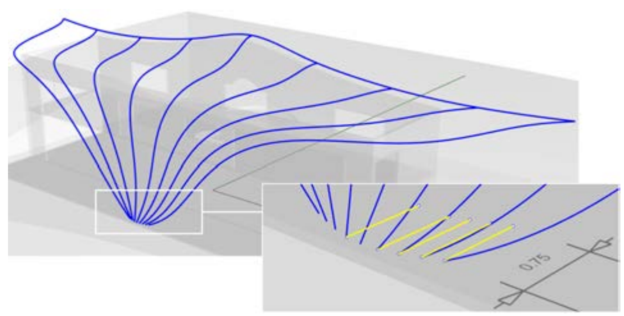
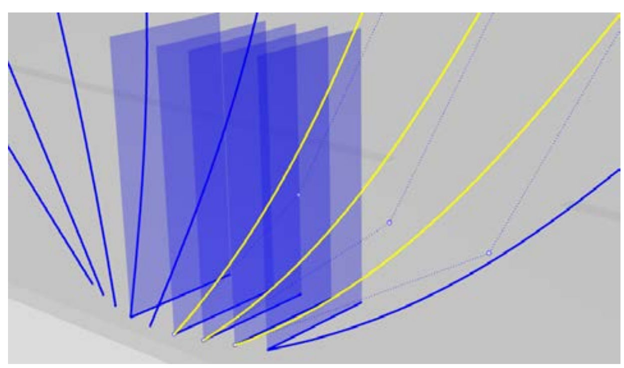
Design rationalization is closely tied with building and fabrication processes. Understanding the materials and building workflows is essential to turn models into buildings. Sometimes designers start with abstract form with little thought about buildability and cost. This can potentially lead to compromising the design intent and form. It is very useful to be informed about fabrication processes and workflows to improve communication with other building professionals and build design ideas that can turn into buildings. The following tutorials explores a couple of common workflows. One to generate custom perforation, and the other is to panelize a free-form.

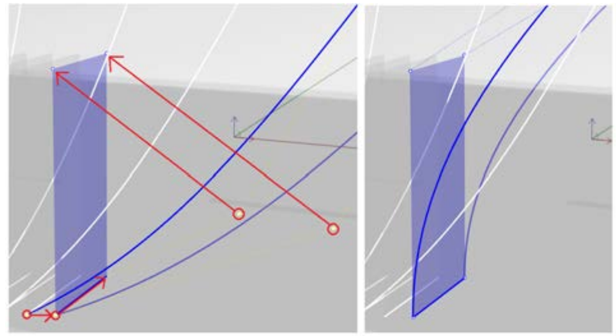
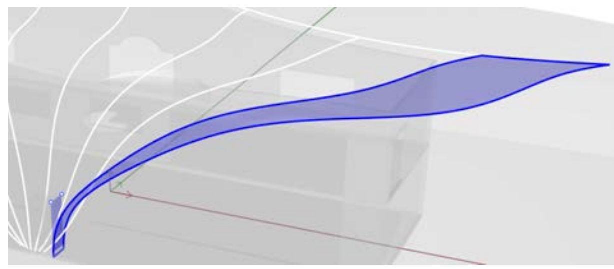
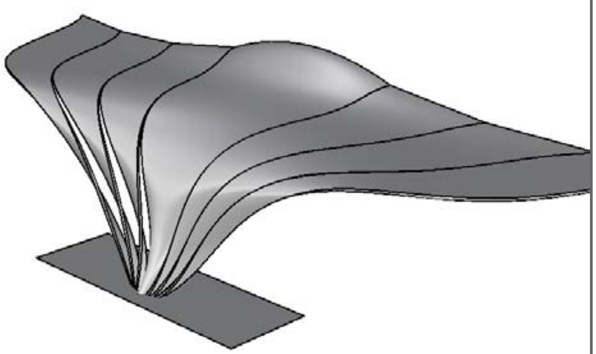
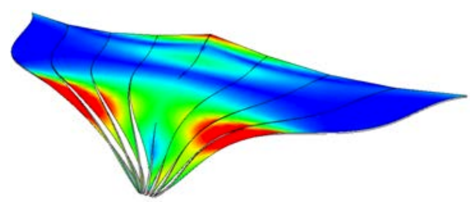
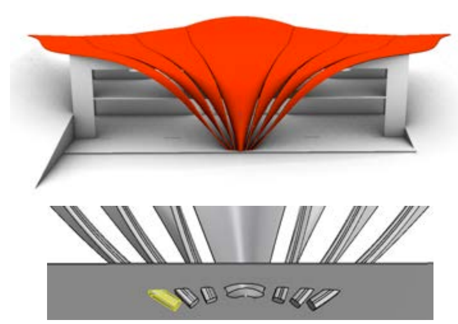
<table>
<tr><th colspan="2">DM_06: Custom paneling</th></tr>
<tr><td>Use an image to print on a screen. This image is taken in the Torrey Pines area in San Diego.</td><td></td></tr>
<tr><td>Create the interior partition with perforation based on regional picture
- Install PanelingTools plugin from <u>here</u>
- Create partition surface (1.8x2.9 m)
- Create paneling grid using ptGridSurfaceDomainNumber with 40x60 u and v spans

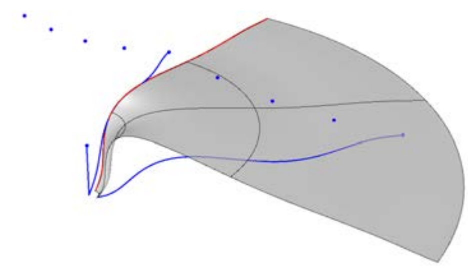
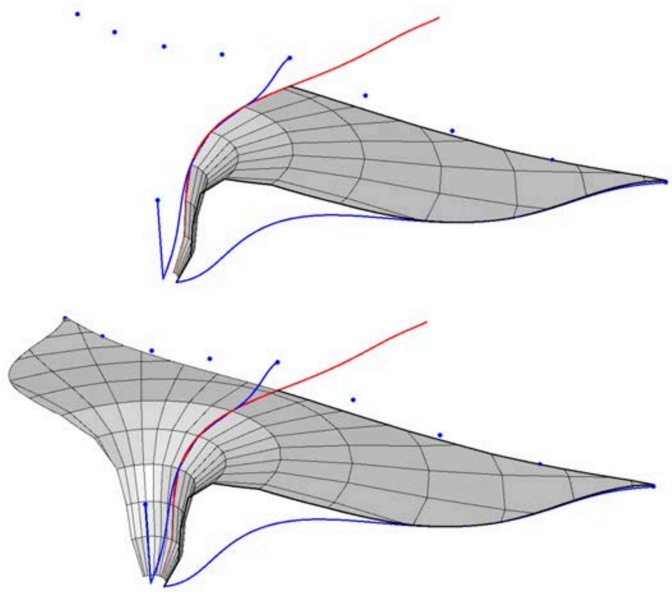
Note: to increase the amount of detail in the partition, you can use more dense grid</td><td></td></tr>
</table>

Create custom perforation - Run ***ptPanelGridCustomVariable*** with ***Scale*** and ***Bitmap*** options. Use a ***Circle*** with a center point as a module. - ***Split*** the surface with the curves and put circle surfaces in one layer and the remaining surface with holes in another - Change layer material for the surface with holes to be glass, and the circular surfaces to be painted. - Create a frame (***Offset*** outline square, ***PlanarSrf*** the 2 squares, then ***ExtrudeSrf***)	

With complex free forms, it is usually a compromise between the form and cost. For the roof, we will first create curved panels, then explore strategy to build it with flat panels (more economical, but will involve changing the form).

DM_07: Roof panels	
Create the roof reference curves - Extract the iso curves on equal intervals including the two edge curves of the roof surface from the loft surface. - Draw short lines (0.75m) on the ground floor at the location where the roof panels intersect with the ground - Lower the lines a small amount to be able to trim the panels after adding thickness.	
Edit reference curves to outline the fins geometry - Extrude the base short curve vertically by 1.5m into short reference walls. - Duplicate the 3 inner curves. - Split the top horizontal curve.	

- Adjust the lower two control points of each pair of the fin curves to align with the extruded short walls to allow a see through gap that works as a vertical louver.	
- Use Sweep 2 to generate each of the fins. Use base and top smaller curves as rails and the side curves as cross sections. - For the middle fin, use three cross sections to create the center fin.	
	- Repeat for all 3 panels - ***Mirror*** to complete
Perform curvature analysis to check the continuity across the fins. You usually want to see a gradual continuation of colors to make a smooth transition.	
- Add thickness using ***OffsetSrf*** with ***Solid*** option and ***FilletEdge*** to soften the edges and add joints between the bays - ***Trim*** out geometry below the platform level with BooleanSplit. - You can leave gaps between panels and turn into a skylight.	

DM_08: Flat panels	
Redefine roof surface so that it can be rationalized with planar panels - Extract the center curve and adjust it if needed - Create revolve surface to cover half the roof	
	Create roof panels (use PanelingTools plugin) - Create the grid with ***ptGridSurfaceDomainNumber*** - Use ***ptPanelGrid*** with ***Faces*** output - ***Split*** the faces the lie outside the boundary of the reference roof surface and delete the outside panels - ***Mirror*** panels to complete the coverage

2.3.3 Advanced visualization

Realistic renderings help present design ideas to colleagues and clients. In this section, we'll define rendering as the assignment of real world materials and the accurate calculation of lighting, reflections and refractions.

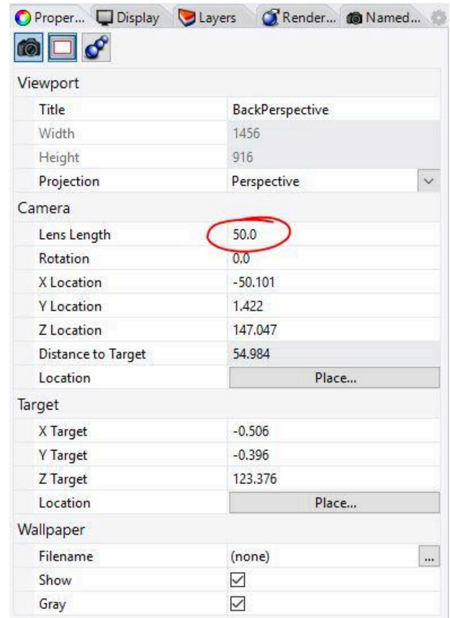
Rhino includes two rendering engines, the ***Rhino Render*** and the ***Raytraced*** as part of the display modes. Both of these utilize the same material library, camera controls, lights, texture mapping and other render settings, with some exceptions.

DM_09: Visualization

Cameras/Views:

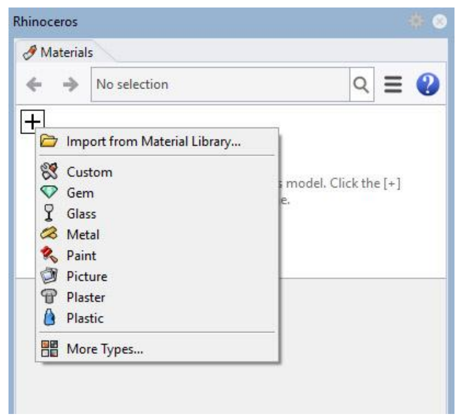
You can set desired named views and assign descriptive names. Clicking anywhere inside your view, check the ***Lens length*** field of the view camera in the ***Properties*** panel. By default the lens length Rhino uses is 50, which matches the perspective you'd have looking at something on a desk in the real world. Let's change the lens length for this named view to 35 instead, which will give the view more of an architectural perspective.

Then go into your named views panel and save the named view again using the same name to replace it so you can use the 35 lens length anytime you restore the named view.



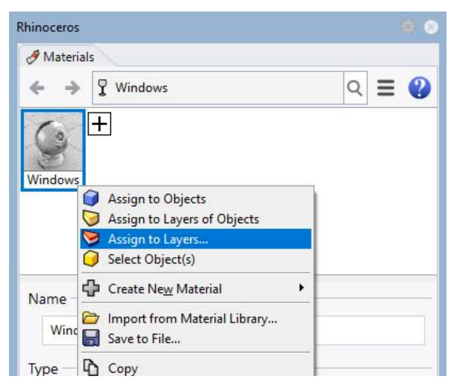
Materials and texture mapping:

To assign materials to the model we first need to create some in the [Materials](#) panel. Click the + symbol to load a standard material type like Glass first.



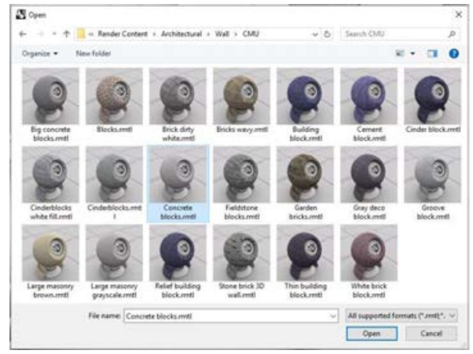
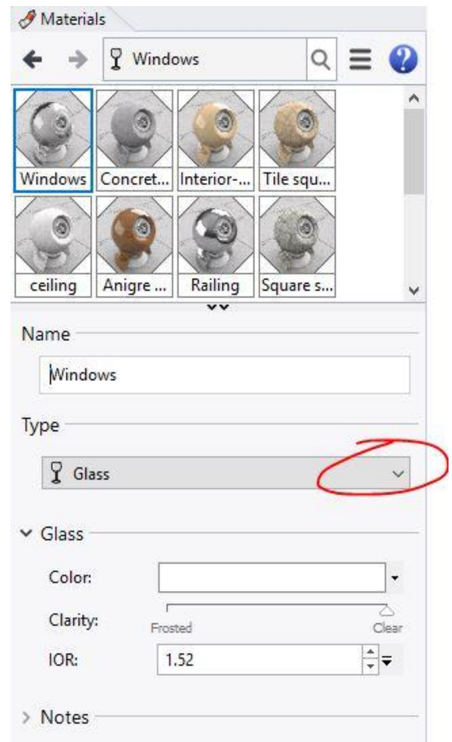
You can rename the Glass material to 'Windows' as soon as it's created by simply typing. Then right click the material thumbnail and choose 'Assign to Layers…'. Choose the 'Glass' sub layer under Openings to assign the new glass material.

Drag and drop of a material swatch onto objects in the viewport or pre-selection of objects followed by 'Assign to Objects' in the right click menu of the material thumbnail are alternatives for applying materials.

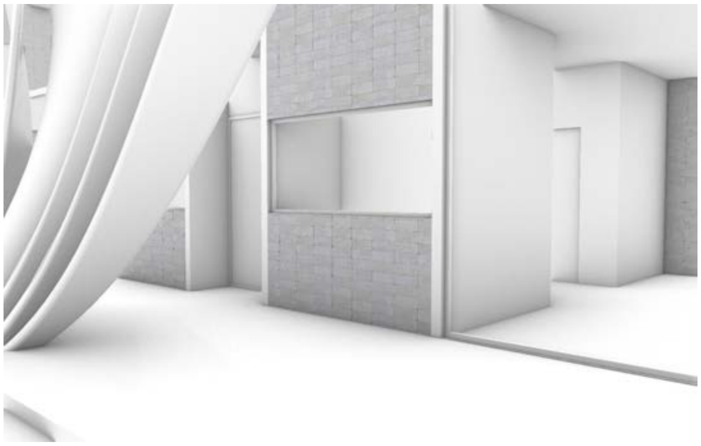


<table>
<tr>
<td>Next, use the ‘Import from Material Library’ option when making a new material in the Materials panel. In Architectural > Wall > CMU select ‘Concrete blocks.rmtl’ and click Open.

The Rhino material library contains many materials that utilizes images also known as texture maps. These are downloading on demand when opening a material for the first time. If you will be offline and want access to all the Rhino material library textures, use the command DownloadLibraryTextures beforehand.

Also note, the window that opens when importing materials is a standard Windows file browser which allows for thumbnail viewing and searching with text filters.</td>
<td></td>
</tr>
<tr>
<td>Standard material types like glass, metal and paint can be converted into “custom” materials if and when you want greater control over their look. Simply click the type drop down menu in the Materials panel to change a material’s type.</td>
<td>
</td>
</tr>
</table>

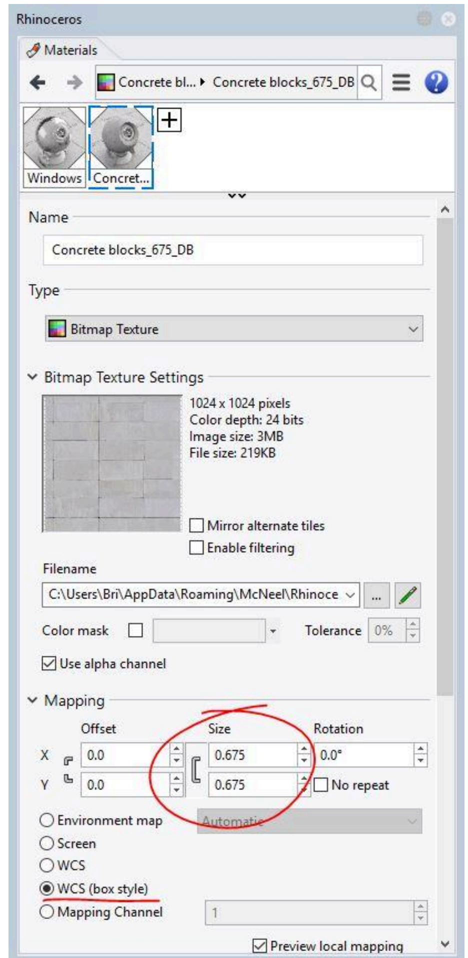
Assign the Concrete blocks material to the ExteriorWalls layer. Then zoom in to see the scale of the texture map. By default, any imported materials from the Rhino material library will be using what is called “World Coordinate System” mapping or WCS for short. This mapping method uses a set real world size for the image textures within the material.



Click the texture name in the color channel for the material to see the settings for the texture map. Note that the size of the texture is set to .675, the model is using Meters as the unit of measurement so this is the scale of the texture shown in the preview once applied to the model. The texture is also mapped onto objects using WCS and the box style of projection. This is great for linear objects like walls.

If you want to change the texture scale you can do so globally here by altering the size. If there is more than one texture in the material, you will need to adjust each one independently. This will impact all objects with the material assigned.

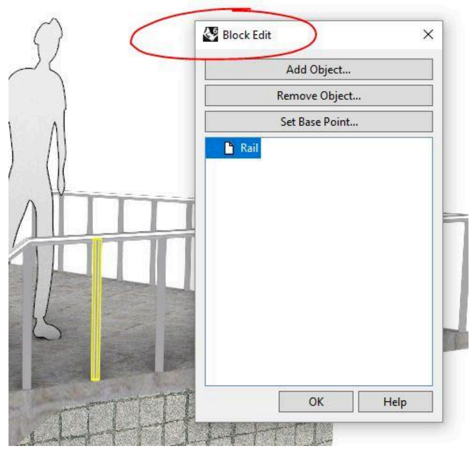
To use non-WCS mapping methods such as planar, cylindrical or a custom unwrap for more complex forms, choose ‘Mapping Channel’ in the texture’s settings and edit the selected objects texture mapping properties in the Properties panel.



Experiment with adding more materials to the model from both the standard material types and the library. Here I’ve added a Paint material to the interior walls layer and a Tile material from the library’s Architecture > Floor category.

It is sometimes necessary to assign different materials to the same object. The slab for the second floor for instance in this model also represents the ceiling for the ground floor. To assign a separate material to the bottom surface of this polysurface, Hold the Shift + Ctrl keys to sub-object select just that surface. You may then right click any material swatch to assign that material only to the selected sub-object. A default Plaster material will work well here as a flat ceiling white paint.

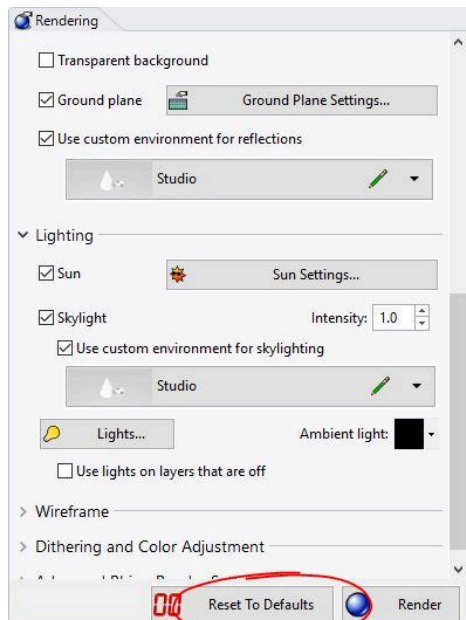
If you are assigning materials to block instances, such as the balusters for the exterior railing, you must do those assignments within the block editor. The advantage of using blocks here is that assigning the material once during an edit will also update the material of all other instances of that block in the scene.



Lighting / Exteriors:

If a model is created in Rhino 6 it will automatically use a collection of new default lighting settings. These can be seen in the Rendering panel and include a default skylight environment for both the color of the lighting as well as reflections.

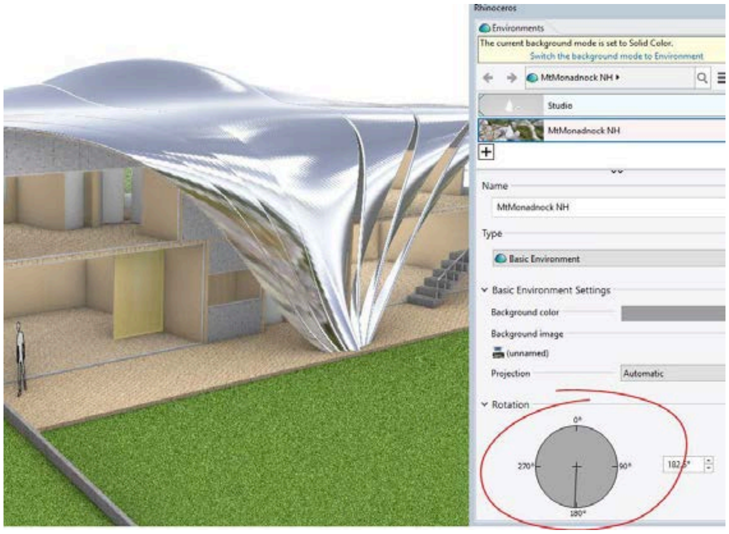
If a model has been imported to Rhino 6 from an earlier version of Rhino, you can click the ‘Reset to Defaults’ button at the bottom of the Rendering panel to use the Rhino 6 defaults.

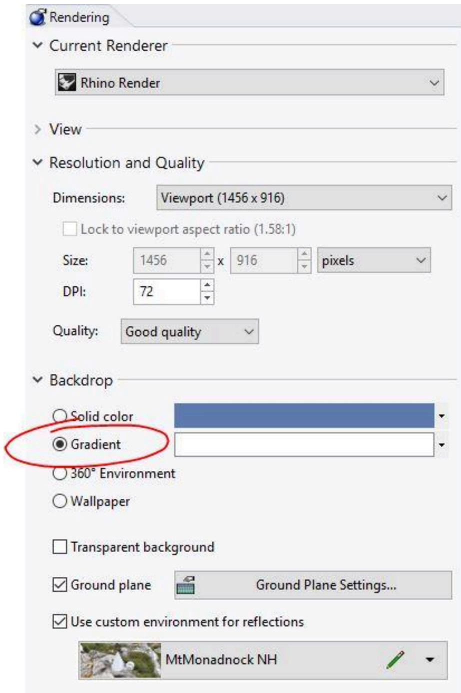


Exteriors are much easier to set up and will calculate faster than interior shots. Let’s first look at exterior views of the model to render. Additional materials have been applied as before using sub-object materials where needed such as the interior wall surfaces of the exterior wall polysurfaces. The Grass bright material from the Organic > Grass category of the material library has been used for the basic ground cover.

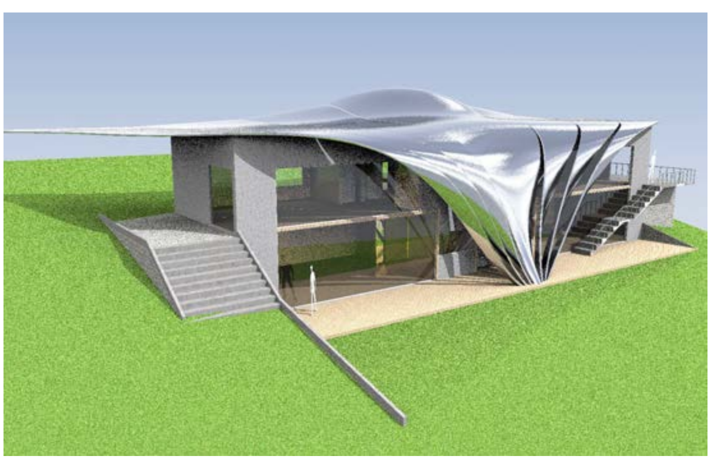
For the roof, a standard metal material with the ***Hatch*** bump texture selected was used. The reflection of an environment sky becomes important with reflective roof material. In the Rendering panel, change only the reflective environment to one from the library by using the drop down menu and choosing ‘use new environment’ >’Import from environment library’, the Mt. Monadnock one for example should work well. Open the Environments panel to adjust the rotation of the environment reflections.

In the Rendering panel you can use the reflection environment as the visible backdrop but let's use a simple gradient instead to keep the focus on our building.





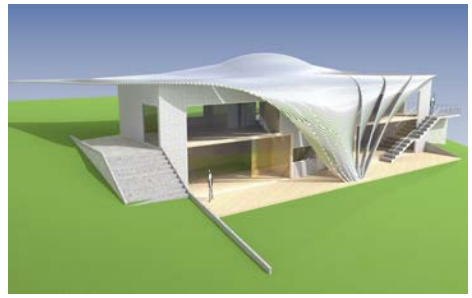
Exteriors are more realistic when the [Sun](#) is enabled as well. In the Rendering panel > lighting section, turn the Sun on and open the Sun settings to set the location and time. Our site is in San Diego so either enter the exact lat/long or use Los Angeles which will come up in the search.



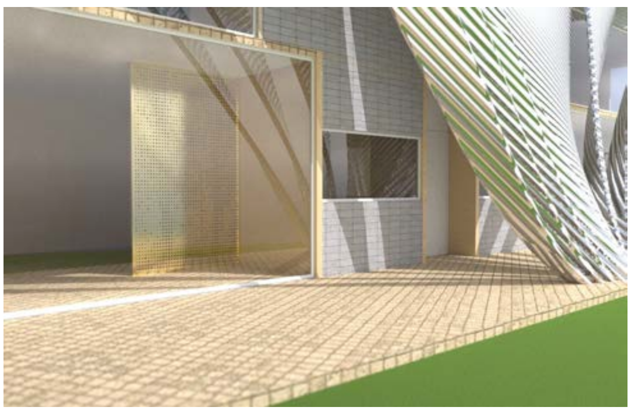
With Rhino Render set in the Rendering panel, use Low quality and do a test render to check overall lighting and contrast. Shadows, reflections and bump maps will appear different from the Rendered display mode. The speed of the rendering is determined by several factors. The pixel dimensions of the image rendered and the quality set in the Rendering panel being the most significant. Here's a comparison of low quality taking 3 mins. versus good quality taking 33 minutes.

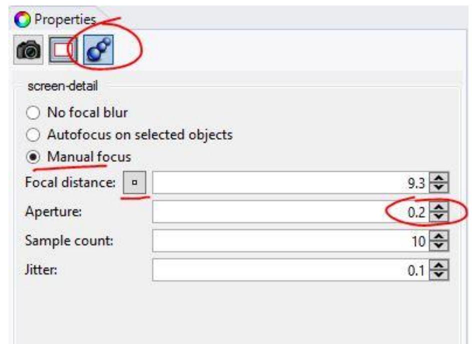
Another option for Rendering in Rhino 6 is the Raytraced display mode which interactively uses a rendering engine called Cycles. You can pick the devices in your computer to calculate the rendering in Rhino Options > Cycles. If you have an Nvidia GPU with Cuda in your computer, your speeds can increase greatly. Multiple GPUs can also be used in unison. Here's the same scene rendered to 500 samples in 3 mins in Raytraced mode.

Note that since Raytraced mode uses Cycles, color saturation and materials won't look exactly the same as Rhino Render and overall will produce a different look.

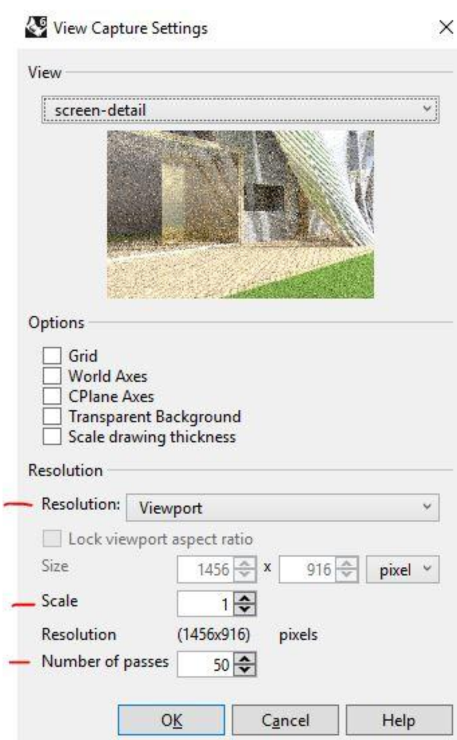


The Properties panel has focal blur settings when nothing is selected and that view is active. These focal blur settings get saved in any named view. Raytraced mode (shown here) as well as Rhino Render can use this feature. Focal blur helps direct the viewer's eye to an area of interest in the composition.





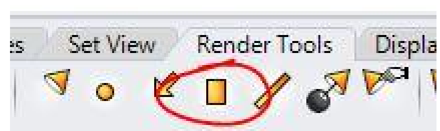
To save a Raytraced image use the ViewCaptureToFile or ViewCaptureToClipboard commands. Make certain that the settings for the resolution are set to Viewport, that the scale is set to 1 and that the samples field does not have a higher value in it then the number of samples achieved in the Raytraced viewport prior to running the command. If these settings differ, a reprocessing of the Raytraced view will begin.

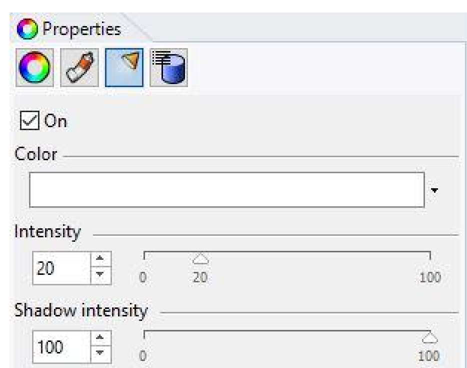


Lighting / Interiors:

Rendering interior scenes often requires the addition of light objects which makes set up a bit more difficult and will also increase rendering times. You may have already noticed in experimenting that Rhino Render and Raytraced mode differ in terms of contrast with interiors. This is because the Raytraced mode / a.k.a. Cycles, calculates what is called indirect illumination or more simply the bouncing around of light. Rhino Render does not. Here are two views without any additional lights for a comparison of Rhino Render and Raytraced.

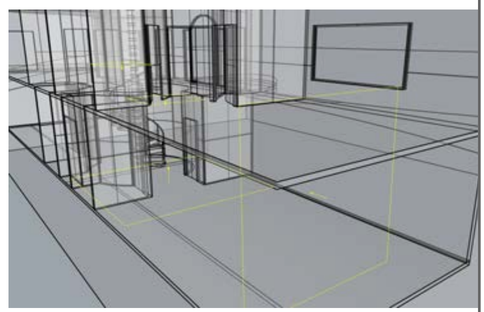
You can add a variety of lights in the Render Tools toolbar group. Follow the prompts in the command line and edit any lights via the Properties panel > Light section when a light is selected.

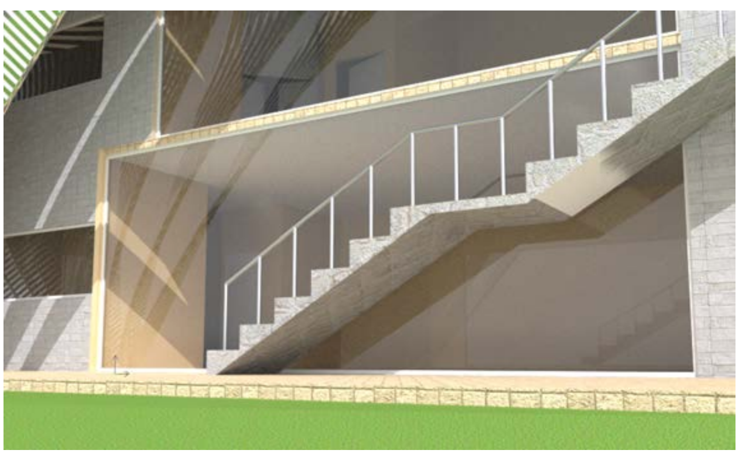


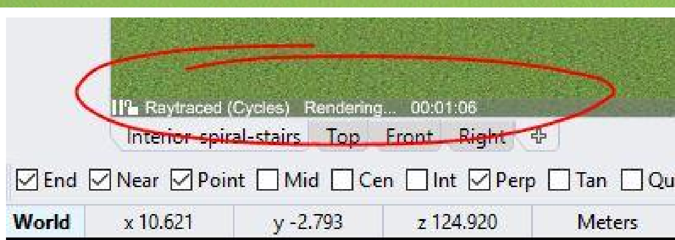


In these two renders, the first with Rhino Render and the second with Raytraced, four additional rectangular lights have been added. Due to Raytraced calculating the bounced light or indirect illumination, the intensity of the lights was lessened to 20% from the intensity of 60% used for Rhino Render.

The image on the right shows the locations of the rectangular lights highlighted in the scene.







An additional note on the Raytraced display mode. At the bottom of the viewport using Raytraced there will be a set of controls for pausing the calculation, locking the viewport from accidental rotation and at the right end the sample count which can be clicked and edited directly while running Raytraced. There's no target sample count that means "done" for all scenes but in general 500 to 1000 samples will produce a high quality image.

2.3.4 Drawings

Extracting 2D data out of the 3D model is useful for documentation and communication. Delivering 2D drawings is a requirement in most architectural projects. Drawings might also be needed to issue building permits and to share with contractors for construction. Rhino models are placed in 3D space to the true full scale of the model in real life. The viewports can be set to have parallel projection to look at the model from different directions. Rhino also supports paper space that can be used to organize 2D data and print to scale. The workflow to create 2D documentations involves two main parts. The first is extracting the 2D drawings, and the second is to lay them out to scale in paper space. The following tutorials show a workflow for extracting and printing drawings.

Resource 6: 3D printing tutorials
Rhino Layout workflow document: https://wiki.mcneel.com/rhino/layouts5

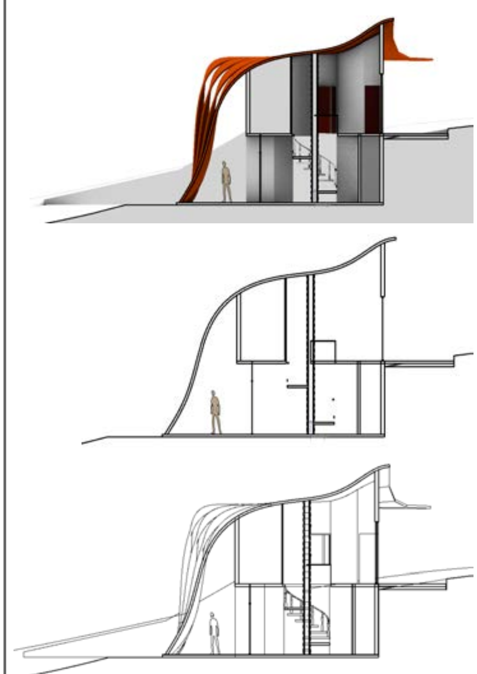
DM_10: Sectioning

You can use Rhino built-in commands to extract sections and elevations using ***Section***, ***Contour*** and ***Make2D*** commands in Rhino, and view using ***ClippingPlanes***. While very effective for quick snapshot of the 2D views, there are some limitations:

- Extracted sections are static (they don't change when the model changes),
- Extracted Sections and Contours need some work to project or reorient the sections from the true 3D location to and other flat plane that the drawings are placed on (usually World Y-Plane).
- Can extract only curves, and need extra work to clean curves, join and create caps, hatches, etc.

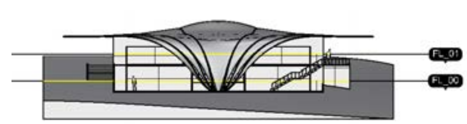
Using Rhino drawing commands:

- Run ***ClippingPlane*** and draw at desired location. You can render the view with the clipping plane active. This gives a raster image of the model.
- Run the ***Section*** command to go through the model in the desired direction, then put output in a new layer. Output is vector drawing (can print with high resolution or export to other vector based applications).
- Run Make2D to extract objects outlines behind the section. Note that Make2D output is placed on WorldXY. You can set one of the views to be World-Top to view output. Note that the Make2D output is vector based.



SectionTools plugin supports dynamic sectioning that updates with model changes. It also resolves some of the workflow limitations described above.

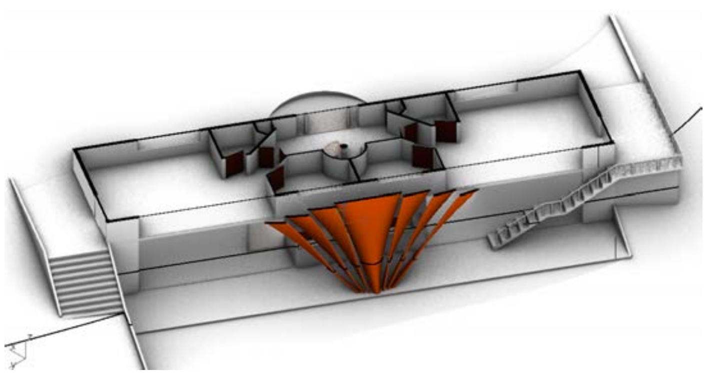
- Install **SectionTools** plugin for Rhino
- Use stCreate, and select objects to section (you can press "Enter" to section through ALL visible objects. This is a good option if you know you need to update the model).

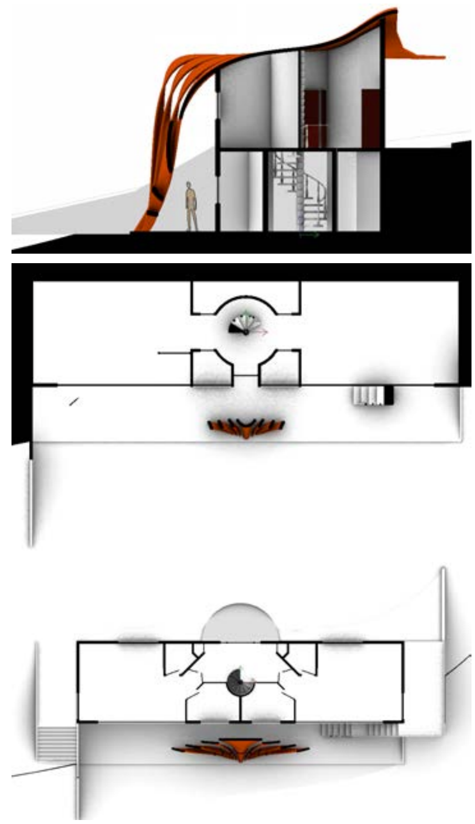


View sections in 3D space
- Run stViewSections, select the section and set options to “Yes”. Select the view to view the section in.
- Use stClearSectionViews to clear the clipping from a selected viewport

Edit sections
- stEditSections to reset options
- stEditSectionsObjects to select specific objects to section, or section through all visible objects
- stEditSectionsHint to edit what part of the section hint to view
- stMoveSections to move sections

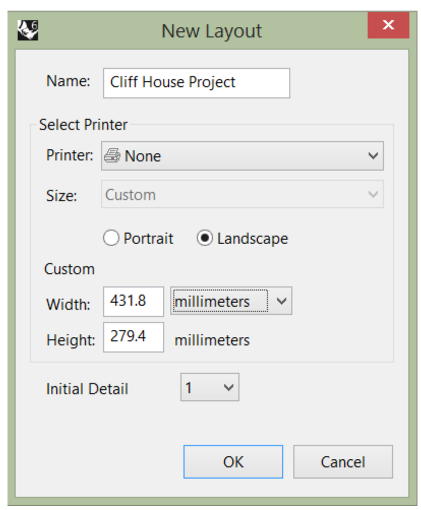
Save desired views as NamedView to be able to set to when needed in the paper viewports





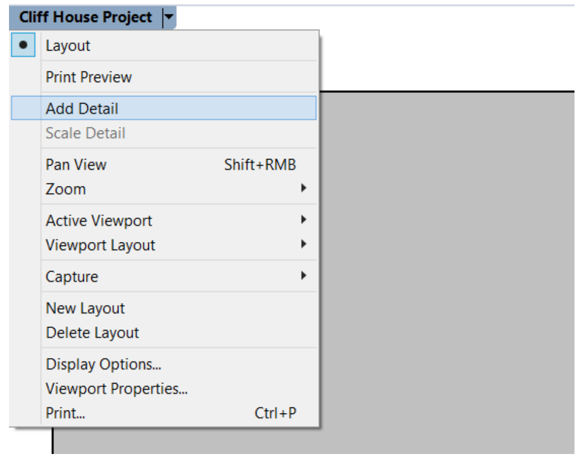
DM_11: Layout

Create a new layout:
- Use the ***Layout*** command to set up a new layout.
- Enter the layout name, paper dimensions and the number of initial details.
- You can use any units to set paper size. That will not affect the actual units of your layout space (which we have set up to millimeters).



Add new Detail:

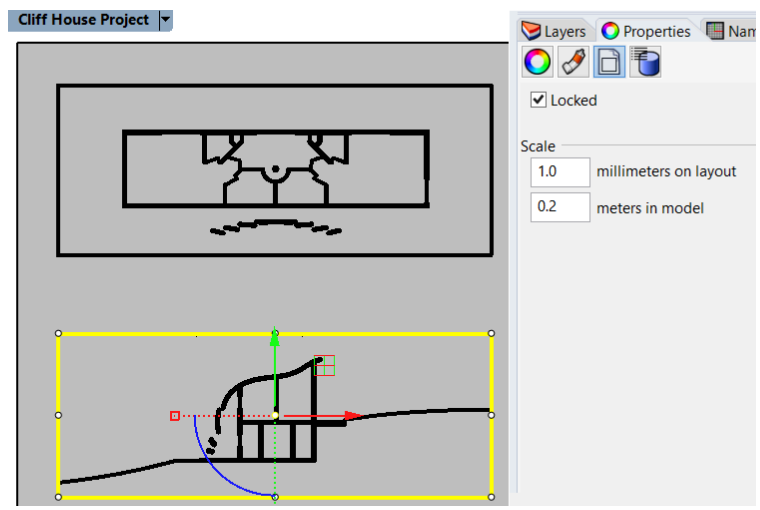
- Press the arrow in the Layout title, and select “Add Detail”
- Drag a window to define the boundary of the detail.
- Double click inside the detail to activate the viewport and be able to Pan and Zoom to the part of the model you need.



Set the detail to scale:

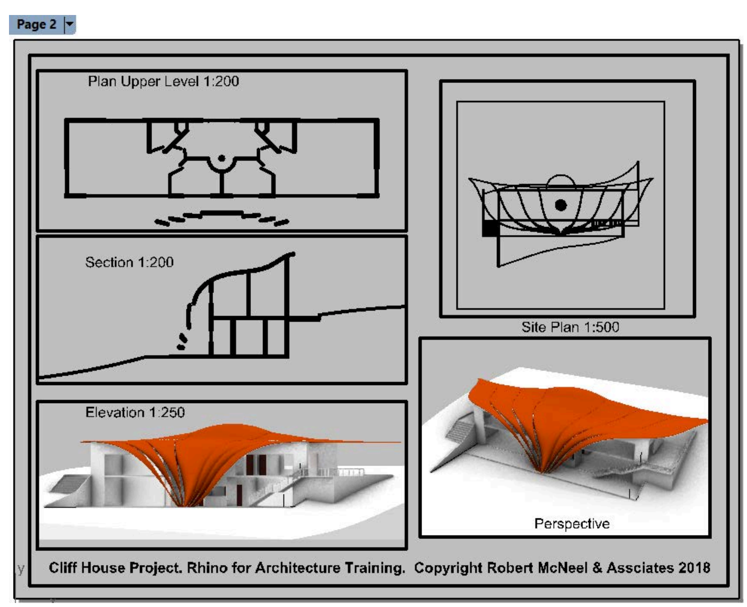
- Select the detail outline
- In the Detail properties, set the scale (1:200) and check the Locked box. Once you lock, you cannot zoom or pan the view to ensure that the scale remains consistent.

Note that you can set different scales for each detail.



Rendered views and labels:

- Enter other details and set the view and projection.
- You can use any display mode for the details.
- Add Rectangle as a border for the layout
- Inset labels using Text command



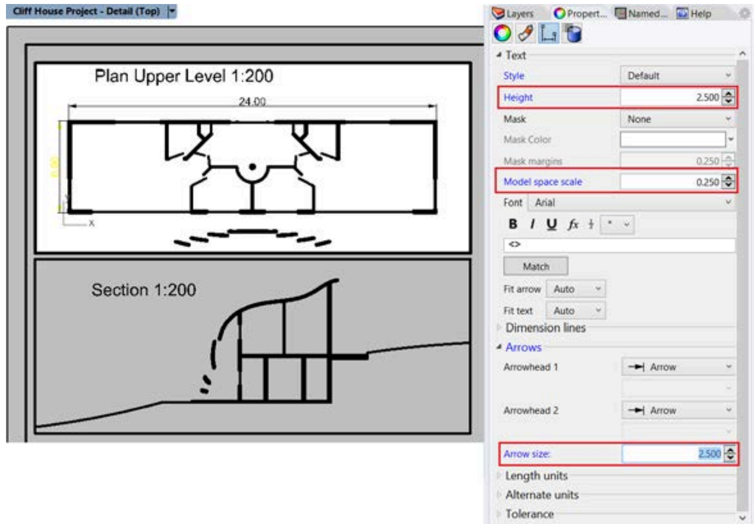
DM_12: Annotation

It is best to do dimension in ***Layout*** view, and you put them in a separate layer. This way you can control their visibility. Note that dimensions <u>DO NOT</u> update dynamically with model changes.

- Create a new layer for Dimensions.
- Create dimensions using Dim command.

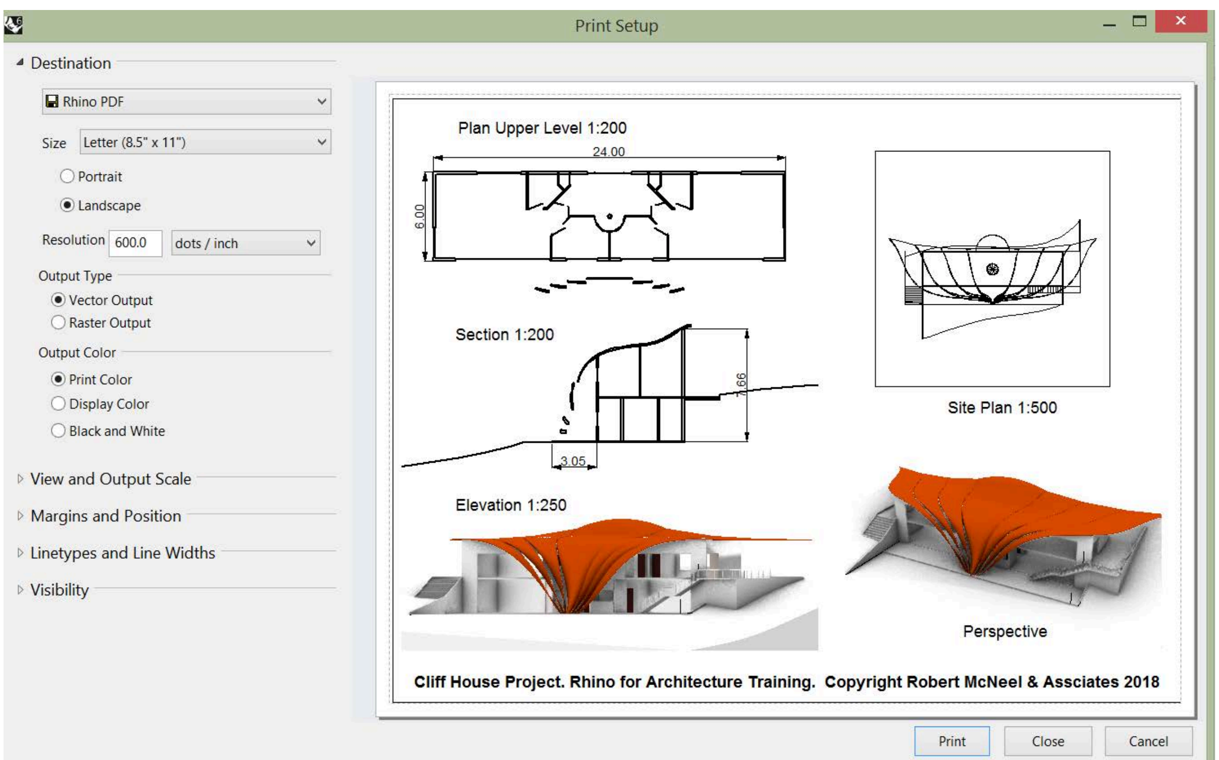
The default scale might need adjusting:

- Select the dimension, and in properties, set Height to a bigger number (e.g. 2.5 mm).
- Pumping the paper space scale of the dimension results in bigger size dimensions in model space. Set "Model space scale" to be a fraction of the paper space (0.25)
- Make sure to set the Arrow scale to match the Height.



DM_13: Printing

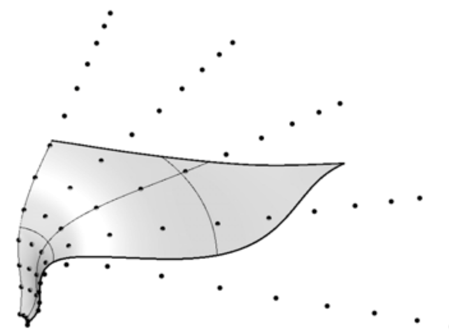
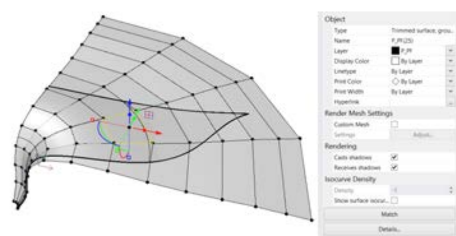
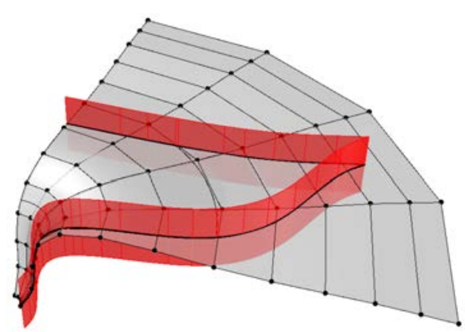
Print PDF using Print command. You can set the resolution and other options.

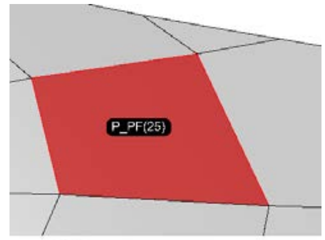
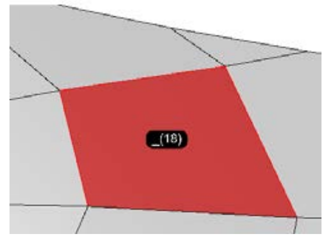
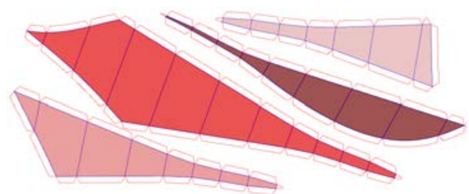
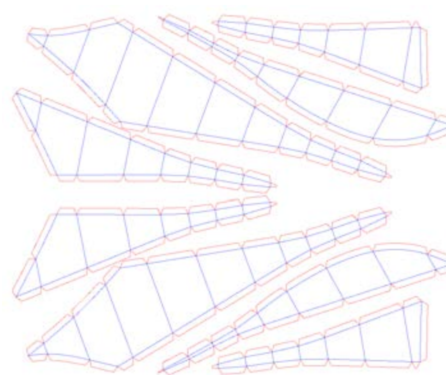


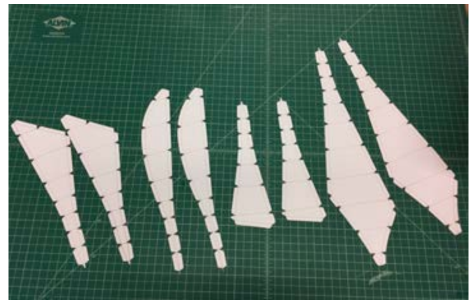
2.4 Prototyping

Prototyping is an integral part of concept development and detailed design. It is as much a design tool as it is for communication and presentation. The nature of digital modeling with virtual screen based representation is bound to miss important aspects that are only noticed through physical modeling. Prototyping uses CAD/CAM systems such as laser cutters, CNC machines and 3D printers. We will discuss two workflows using 3D printing and laser cutting, and CNC routing.

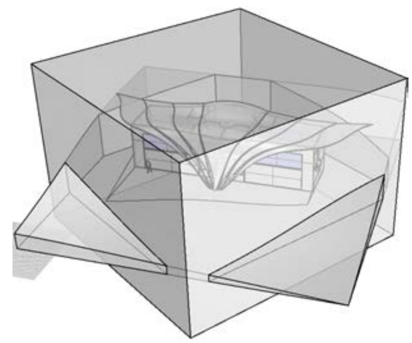
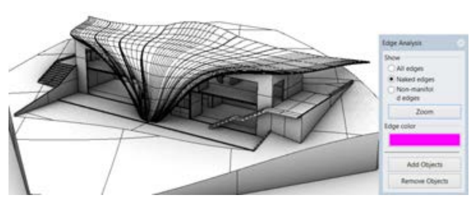
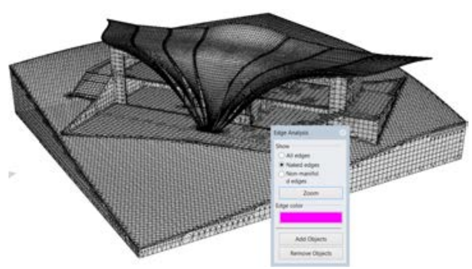
2.4.1 Laser cutting

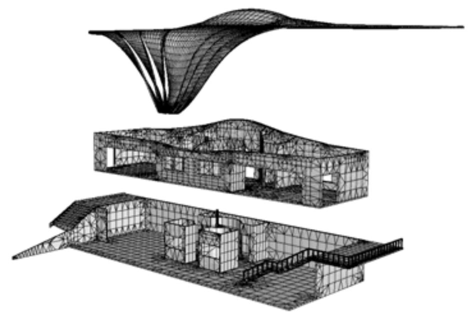
P_01: Laser cutting	
Use rationalized roof panels (flat panels). For this tutorial, we will rationalize with reduced number of panels. - Project the outline of the original roof onto the revolve surface and trim with it (without shrinking the surface). - Divide the surface into 4x12 spans using ptGridSrfDomainNumber (PanelingTools plugin)	
Create the panels - Use ptPanelGrid to generate the panels as Faces - Notice that faces are grouped and each face has a unique name	
Split panels - Extrude outline curve - Use Split to split all panels - Delete the part of panels outside the boundary	

Labeling - If the autolabel for panels is not desirable, you can relabel it using ptSerializeObjectsName - To tag objects with their name, use ptTageObjects	
Unrolling: you can unroll, tab and glue individual panels, but also look for a more efficient way if the form allows it. - Join each step of panels, and make sure the polysurface normal faces outwards. - Unroll together using UnrollSrf - Change the color of steps to make it easier to distinguish	
Create tabs - Create new layer and choose color RED and make it the current layer (many laser cutting programs designate the red color for cutting and blue for etching) - Arrange your strip to fit in the laser cutter bed (check your laser cutter dimensions) - Use ptTabs with Recess option to create the cut outlines	
Extract etch curves - Explode trips, the DupBorder to get the curves - Explode the curves then remove duplicates (SelDup, then Delete) - Put curves into a new layer and make its color BLUE - Mirror all curves for the second half of the roof - Select all cut and etch curves, and Export using file format suitable for your laser cutter Note that if you need to bend panels in 2 different directions, then use a dashed line red cut curves for etching instead of blue continuous curves. Also, take into consideration the final scale of the printed part.	

Cut panels using the laser cutter Put it all together by gluing corresponding tabs	

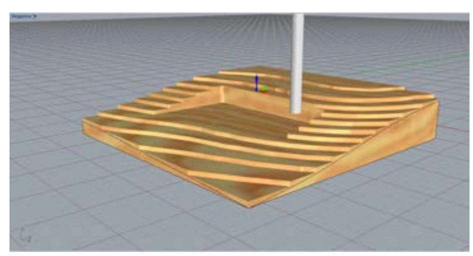
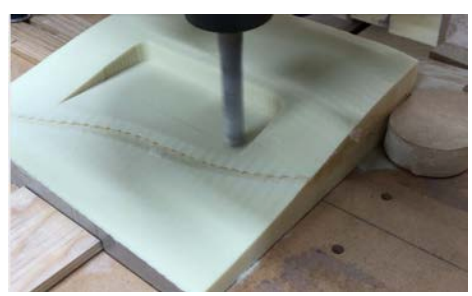
2.4.2 3D printing

P_02: 3D printing	
Check your printer tolerances. 3D printers have a minimum thickness that they can print successfully. - Set model scale to fit the final print size. Usually set units to millimeter. - Make sure all thicknesses are within tolerance. Recreate the solids to have proper thickness even if it is a little different from the design intent.	
- BooleanUnion as many polysurfaces as you can. Ideally end up with exactly one polysurface. - Make sure you have CLOSED polysurfaces. Note that you may need to do extra operations to Boolean successfully and make thickness within printer tolerance. You may use OffsetSrf, Move objects to overlap, etc.	
- Extract a mesh with reasonable face count. 3D printers have a limit of how many faces they can print, and minimum size. - Check the mesh to make sure it has no naked edges or holes. Repair when needed using MeshRepair tools and other Rhino mesh commands - Export using proper format acceptable by your 3D printer. Usually as STL or OBJ	
- It is good practice to supply the Rhino file with the original polysurface as well as the mesh to the 3D printing service. This way if there is a scale or thickness issue, they can modify the original model and export for better resolution.	

- Total volume and support material can be an issue for cost and profitability of the parts. - For the final print with Resin, the option to break the model into 3 parts gave best results in terms of cost and ability to print with min support. The cost was reduced from ~$500 to $75.	

Resource 7: 3D printing tutorials
3D printing workflow document: https://wiki.mcneel.com/rhino/3dprinting 3D print tutorial for a product design example: https://vimeopro.com/rhino/preparing-to-3d-print

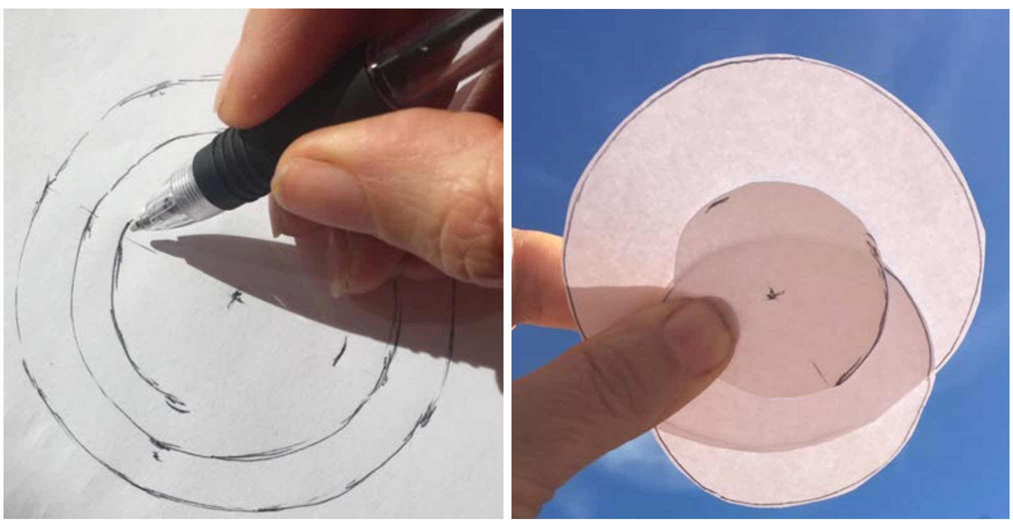
2.4.3 CNC routing

P_03: CNC Routing	
Scale the model and adjust model to work well with your router tolerances Check for angles and adjust undercuts and angles when necessary depending on your CNC machine specifications and tools Depending on the CAM tool used, you might need to export as a mesh. Note that the mesh does not need to be closed like in 3d printing. CNC routing model is very forgiving. However, you need to know you machine to adjust angles and scale.	
CNC settings and simulation using RhinoCAM	 ***[Show simulation video]***
Final cut site model using CNC machine	 ***[Show milling video]***

Part III: Modeling methods in Rhino

Architectural design involves creating ideas using a language of expression such as geometry and materials; and a medium of representation such as drawings or modeling. The media of representation tends to influence our design thinking and methodology. This is particularly true in the digital medium of design. Different digital tools are designed to support the different ways of modeling.

To understand the various modeling methods that Rhino and its plugins support, let us start with a simple example. Suppose you want to create a composition out of circles with varying radii. You might start with drawing on paper, cut and experiment with different compositions interactively. This method uses a very familiar “tool” (pen and paper), and a basic understanding of geometry (circles have a center and circular curve). One also needs a bit of practice to come up with good hand sketched circles.



Hand drawing circles

Benefits of using pen and paper to represent geometry are plentiful. Drawings are cheap, available, easy to pass around, and we all learn how to draw with pencils from a very early age. Drawing medium is also great at keeping a record of all the attempts. However, there are some disadvantages, for one, it is hard to draw accurate circles by hand. Also, drawings inked on paper are hard to change. Designers have invented many tools and methods to help with drawing challenges. For example, a compass to improve accuracy, tracing paper to layer sketches, and so on. Just like drawing, digital representation of design ideas has its advantages and challenges. To fully understand those, we will examine a range of digital design methods.

Digital modeling supports a wide range of workflows for different stages of design; from intuitive modeling with vague ideas about the final result, to more structured and well-defined designs.

For example, you might start with some surface, have not decided yet if it is a wall, a ceiling, or simply a guiding geometry for something to follow. At that stage, you need flexibility to shape and mold your ideas. If your tools force you to decide on the material or building part, then your flexibility to change your mind and morph your initial thoughts into new ones will be hindered. On the other hand, if your design has matured and you need to deliver your model to engineers for evaluation, you need to have made detailed decisions about building parts and materials.

Creators of digital tools examine these needs and workflows closely and create digital tools that suit different stages and modes of design. In general, the creators of digital tools tend to support one of four modeling approaches: direct, algorithmic, object-based or parametric. Some of these overlap to a certain degree, especially the parametric one which we will address in a separate section. The main characteristics of the first three approaches are summarized in the following.

	Direct Modeling	**Object-based Modeling**	**Algorithmic Modeling**
Representation	Uses abstract geometry with no explicit decision about that building part the geometry represents	Uses well-defined objects such as building parts that embed information about geometry materials, and other attributes	Process driven focusing on describing the steps whether to create some geometry or a building part
Modeling approach	No restriction on the forms created or processes used to create them. Highly adaptable to designers preferences	Uses a library of parts to assemble a model	Incorporates mathematics, logic and algorithmic processes to define forms and relations
Design stage	Suitable for intuitive conceptual design	Efficient when modeling specific building types and styles, and for production.	Desirable for parametric design
Interoperability	Abstract geometry is usually highly portable across digital tools, which favors its inclusion in many design workflows	Rigid. You are usually limited to the workflow designed by the environment itself and its family of tools. There are some industry standards to help interoperability.	Can communicate with external tools, but is dependent on the environment it is developed with.
Communication	Easy to understand and manipulate across team members	Relying on standard objects is a significant advantage to help consistent modeling among team members.	Well formed and clear algorithms is key to collaboration. It is easy to generate unreadable scripts that are hard to understand and manipulate.

Comparison of digital modeling methods

The Rhinoceros core modeling environment supports direct modeling through its intuitive geometry creation methods and the rich set of tools to manipulate, analyze and share geometry. The Rhinoceros core is very easy to extend into specialized functionality through plugins. It has an open source file format (openNURBS), and much of its core is built using the same development tools available to all third party developers. Grasshopper, which started as a Rhino plugin and now ships with Rhino, has become a standard tool for algorithmic design. Its intuitive visual programming method, coupled with the powerful Rhino geometry engine make it very popular among designers and building professionals. Many plugins for Rhino and Grasshopper support specialized workflows. For example VisualARQ uses object-based modeling with standard libraries of building parts. ArchiCAD, which is a stand alone object-based tool, has a plugin linking Rhino and Grasshopper in real time.

Note: VisualARQ, BIM plugin for Rhinoceros and Grasshopper
VisualARQ makes it very easy to work with building parts (walls, windows, etc.). It is tightly related to both native Rhino geometry types (easy to transition between the two), and the algorithmic environment of Grasshopper (for example, it can define blocks algorithmically, and build relational models of the building). For more information about VisualARQ, go to http://www.visualarq.com

The accessibility and ease of adding to Rhino and Grasshopper resulted in a growing ecosystem containing hundreds of specialized tools for analysis, interoperability, robotics, visualization, and others. Most of these tools are shared for free and present an incredible resource. The inclusive nature of the Rhino and Grasshopper and its affordability make it a great choice in research and practice.

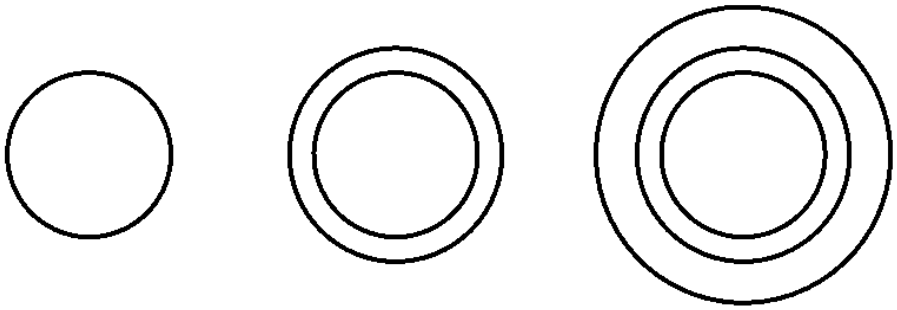
Resource: Food4Rhino lists plugins for Rhinoceros and Grasshopper
Hundreds of plugins for Rhinoceros and Grasshopper can be downloaded from the food4Rhino website. Most are free: http://www.food4rhino.com

Direct modeling

Direct modeling is very comparable to pen and paper. After you familiarize yourself with Rhinoceros user interface (2 hour tops), you can start modeling with simple geometry such as curves and surfaces. For example, creating circles in Rhinoceros involves simple steps:

- Find a computer that has Rhinoceros installed,
- Run Rhinoceros and start a new file,
- Run “Circle” command (find it in toolbars, menus, or type the word “circle” in the command line)
- From here, you are guided through the steps (instructions are typed in the command line). You’ll be asked to specify a center, radius, all with some nice preview to see what you’ll get before you commit or accept.
- Run “Circle” command a couple more time snapping to the same center point, but with different radius.
- For now, ignore the many different ways you can create a circle that are offered by the “circle” command, or else you’ll be faced with more things to learn and decisions to make.

You end up with something like the following:

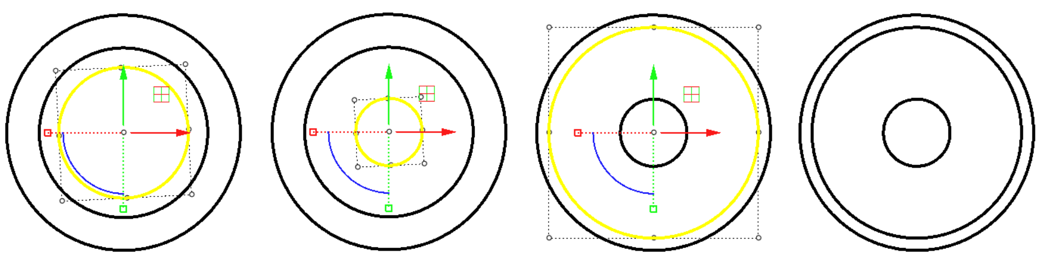


Modeling circles in Rhino

Designers usually get the hang of direct modeling fairly quickly for few reasons:

- The general workflow is similar to using pen and paper.
- There is not too much upfront work that the designers need to do other than opening the application and starting drawing, albeit digitally.
- Actions (or commands) are typically intuitive and easy to remember. You can guess many of them (type a couple of letters, and the smart autocomplete will pull a list for you to choose from). All tools are also grouped in a logical arrangement in menus and toolbars.
- Once you run a command, it usually guides you through the process step by step.

Once you're comfortable making circles with Rhino, you'll find that there are many advantages that come with direct digital sketching over the good old pen and paper. For example, it is trivial to interactively scale until you are happy with the circles! But beware, if you are not disciplined, you can easily lose your early iterations. Unlike on paper, no trace is left on screen. You can always "undo" but that will only take you a step or two back.



Editing circles in Rhino

Both methods mentioned above (hand drawing, and direct digital modeling) have limitations:

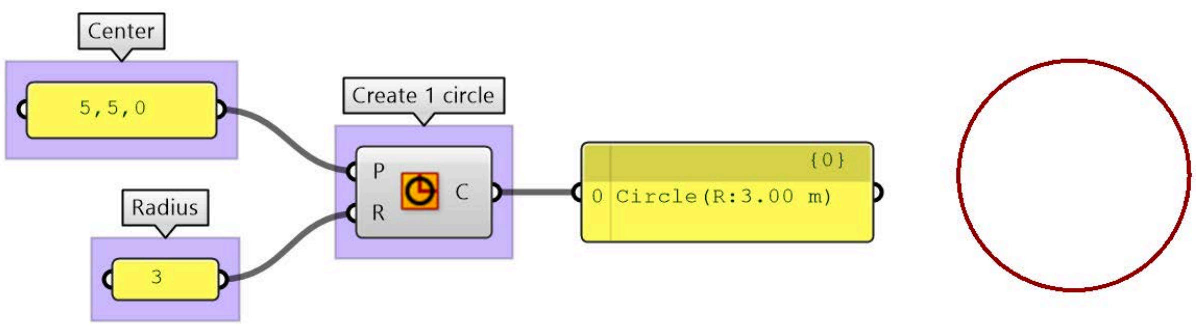
- Remodel every time you need to create the same thing.
- Hard to keep and compare variations.
- Hard to define dependency or relationships between the parts.
- Hard to embed knowledge about the process or the logic of design.
- Involves more work when transition to detailed design and documentation.

If you hit these limitations frequently at your work, then you should consider using algorithmic or object-based modeling.

Algorithmic modeling

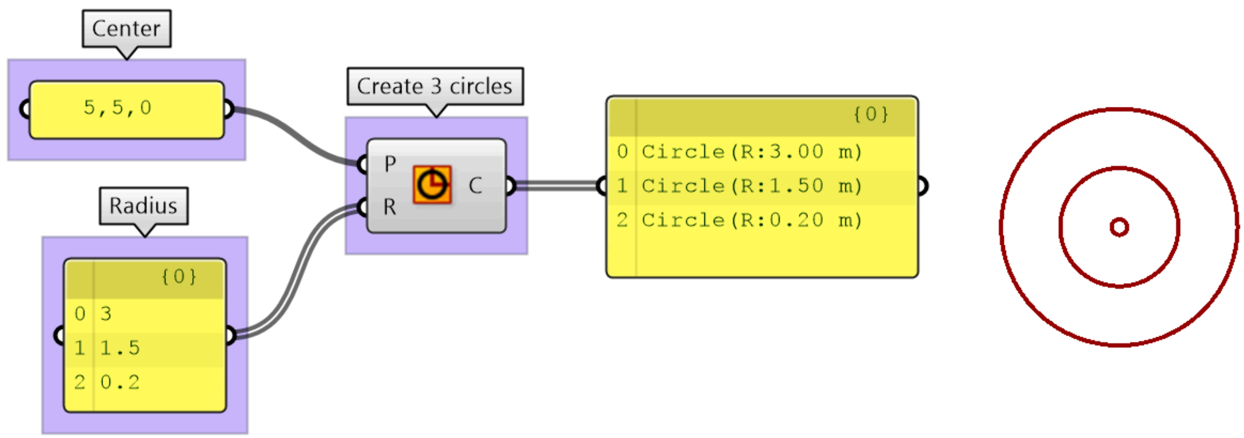
Algorithmic modeling requires clear articulation of design problems and the steps to reach the solution. Once you wrap your head around the algorithmic design workflow, you will be able to overcome many of the limitations inherited in other modeling methods. The main advantage is the ability to work on projects early because it is fast to change input requirements and make new updates. It also helps with scalability and redoes of similar problems. There is, however, an upfront cost to learn algorithmic modeling. Knowledge of basic math, geometry, and logic design is essential. More importantly, developing algorithmic design skill requires discipline and commitment to learn. The good news is that you can acquire the skills of algorithmic design with practice and time, and once you get it, it is hard to lose.

It is best to explain algorithmic modeling workflow through an example. We will use Grasshopper to show a typical workflow. Let us start by exploring few different ways you can “define” a circle, or a bunch of them using Grasshopper. Notice I used the word “define” instead of “draw” or “model”. The reason will become clear later. First, we will define one circle with a center point and radius. Then we move on to constructing different ways to define concentric circles.



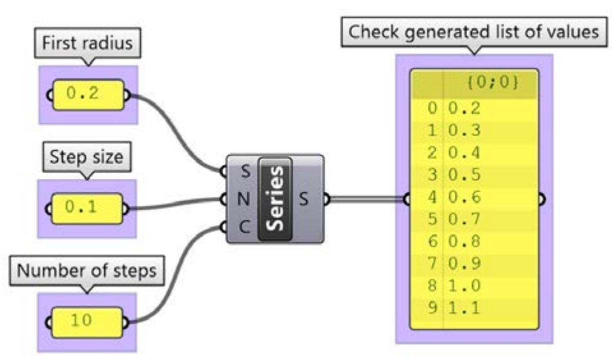
Define one circle with fixed (constant) center and a fixed radius

Note: How input is processed in the Grasshopper component
Grasshopper supports many ways to define “input” (the values on the left that feed into the circle battery). The input can be one or multiple values, and hence the circle may execute any number of times (one for each input combination). In the above example, the “circle” runs exactly one time using the one (center, radius) combination. The output (coming out of “C”) at the right side of the component, show that we have created exactly one circle. A preview of the circle is drawn on the Rhino viewport.



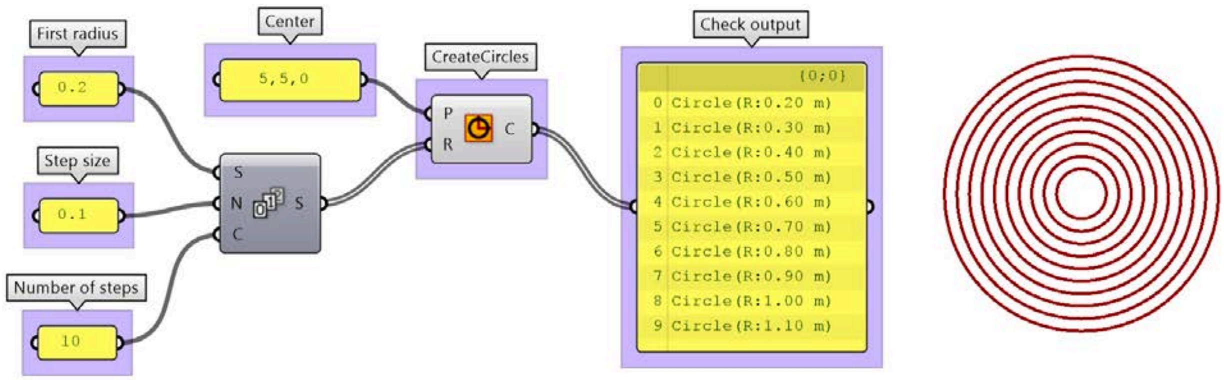
Define three circles with a constant center and constant list of radia

Grasshopper supports multiple ways to define or generate input values. You can directly supply them, but you can also generate them. This is where the power of algorithmic methods shows. For example, if you need 10 circles starting at radius equal 0.2, and increasing by 0.1 all the way to the tenth circle with radius equal 1.2, you can do that in many different ways. You can create a separate definition for each circle with one radius (similar to example 4). You can also supply a list of typed radia (similar to example 5). These two methods are correct, and they mimic the “direct modeling” method of repeating each circle. This is tedious to change and does not exploit the power of algorithmic design. A better way is to generate the list using a starting radius, step size and number of radia. This way if you like to change the initial radius or step size, then you can do that efficiently by changing only one value. You’ll notice that definitions resemble building blocks connected to each other to generate the desired output. First, create the list:



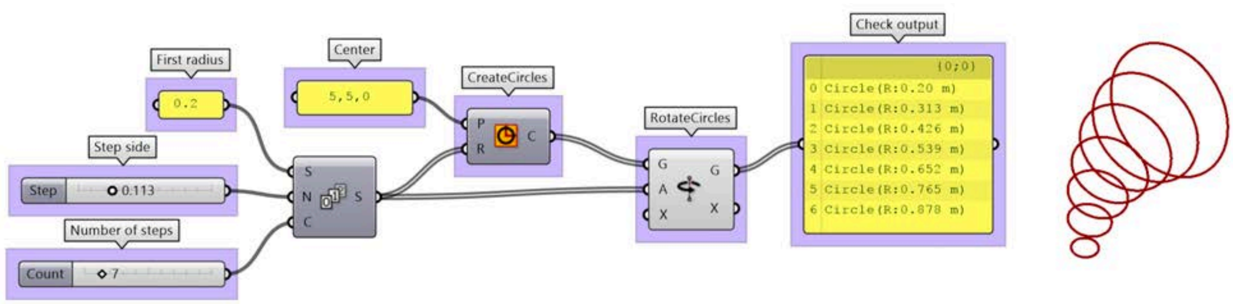
Use Series to generate a list of values

Then use the generated list as input to the circle radius:



Generative circles

One big advantage of using algorithmic modeling is that it naturally support parametric design. For example, instead of one value, you can define an input parameter as a range of values. You can then interactively change input and observe the effect on your output.



Parametric circles

Note: What do components use as input when lists are involved?
Grasshopper has a specific way to generate the list of combinations from input. This is a very advanced topic and has to do with data management, but it is worth mentioning here to give you the heads up about how it works. The general case is easy to understand. If a single value is supplied to each of the input, then those make one combination, and the component is executed once. If one of the input is supplied with a list of values, then the component executes once for each value on the list and is combined with the same other single value input. For example, the radius input consists of three values and only one point for the center. Subsequently, the circle component executes or runs three times using the following combination for (center,radius): ((5,5,0), 3), ((5,5,0), 1.5), ((5,5,0), 0.2). The output coming out of “C” at the right side of the component shows three circles. Now if a list is supplied for each input, the values of the same index are combined. If one list is shorter, the last value on the short list continues to be used. For more control, Grasshopper makes many tools to manage how data is matched. For example, there are ways to combine each value in a list with **all** other values from the other list, but we can leave this for another time.

At this point, it is useful to define some terms commonly used in algorithmic modeling:

Data	Includes all the values (numbers, text, geometry, color, etc.) that are processed to create the output. These values take two forms: **variables** (fixed or certain) and **parameters** (change, uncertain). Data containers in Grasshopper are all called "parameters".
Data structures	Grasshopper organizes data within three structures: **single**, **list** and **tree.** Properly representing data structures and managing them is a big part of algorithmic modeling.
Functions, methods, and operations	Those typically take input, perform some operation, and produce output. In Grasshopper operations are encapsulated inside what is called "components".
Algorithms	They are the steps (recipe) that define the sequence of operations. It has three main parts: input, steps, and output. Grasshopper files contain one or more algorithms, which are commonly referred to as "definitions".
Parametric design	A method to create design solutions through a set of logical steps (**algorithms**) and values that can be changed (**parameters**) to help efficiently generate design variations. Parametric design is highly supported by algorithmic modelers.

Table (2): Algorithmic modeling concepts and their definitions

So far, we learned that Rhinoceros supports direct modeling familiar to most designers, while Grasshopper is an algorithmic modeling tool that helps articulate the design logic using algorithms. Here is a summary of what Rhinoceros and Grasshopper are good at, and why you might want to use them.

Rhino (direct modeling)	**Grasshopper (algorithmic modeling)**
Captures the intuitive workflow of traditional design medium of pen and paper	Based on computer programming principles, but is made intuitive through visual, rather than text-based scripting
Uses NURBS to represent and manipulate geometry accurately. But also support other types of geometry such as meshes	Rhino geometry commands are embedded in the components, combined with data management tools to support visual algorithmic modeling
Design decisions can be implicit and reflective	The workflow forces explicit definition of all modeling steps
In large, designers do not have to deal with or manage geometry data.	Understanding data types and data structures is essential. Designers have to be aware of data and actively manage them
Offers direct interaction with geometry. Typically work directly in model space.	There is separation between logic and geometry. The design logic is created in a separate space from that where the geometry is displayed.
Making changes may involve remodeling. It is not easy to generate and compare design variations.	The ability to change designs and create variations is perhaps the most pronounced advantage.
Hard to leverage mathematics and algorithmic logic. Also hard to build model dependencies.	Mathematics and dependencies are in the nature of this method.

Affordable, stable, accessible in Windows and Mac operating systems.	Fully integrated inside Rhinoceros in both Windows and Mac.
Flexible user interface. Customizable tools using macros. Also, supports many scripting languages (RhinoScript and Python) and plugin development (C++ and DotNet framework).	Provides access to the platform scripting and development libraries and functions. All can be accessed through text-based editors (Python, VB, and C#). Grasshopper functionality can also be extended with compiled Add-ons using the DotNet framework.

Table (3): Characteristics of Rhino and Grasshopper

Object-based modeling

Rhinoceros does not support object-based modeling as part of the core application. However, there are a few plugins that are integrated tightly with Rhino that do support this modeling method. One example is VisualARQ.

Object-based modeling almost never deals with abstract geometry. Designers typically model with objects such as walls, windows, stairs, and roof. These objects behave in predictable way and hold information about materials, cost, 2D representation, etc. Designers use standard objects or create their own. The main advantages of using an object-based modeling application such as VisualARQ can be summarized in the following:

1. IFC is the industry standard format to store objects and their properties. Many object-based applications use this format; therefore, there is good interoperability between them. Many other building applications for analysis and construction support IFC format which makes it more convenient to exchange files.
2. Objects can embed a lot of information about materials, 2D representation, cost, etc. This makes it more straightforward to generate 2D documentation and bills of materials.
3. Consistency in using standard styles and building parts across an organization or in future projects. Once these libraries are established, it is very productive to work with object-based modelers, especially for similar style projects.

Parametric modeling

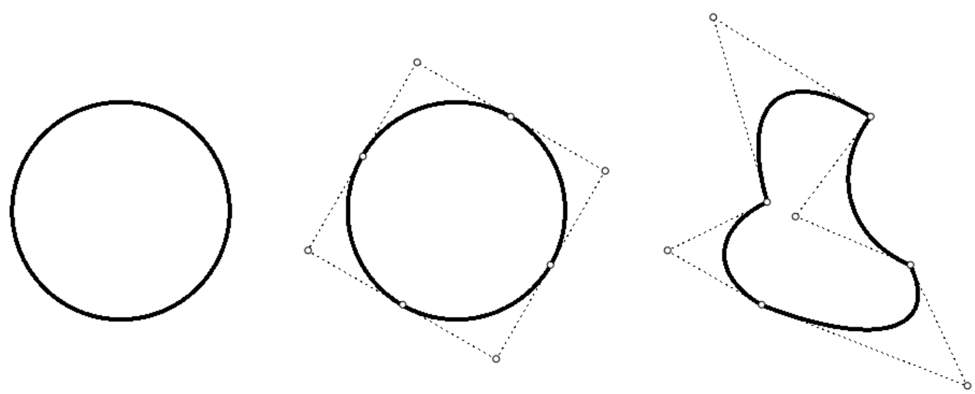
Parametric modeling is a specialized modeling mode that is gaining popularity. It overlaps with all other modeling methods. In algorithmic modeling tools such as Grasshopper, all input values can be turned into parameters. Object-based modeling is also parametric by nature where changing specifications of any style (height, thickness, material, etc.) leads to updating all instances that use that style. It is less obvious how parametric design is supported in direct modeling tools. That is said, the Rhino core does support parametric design in three ways. The first is embedded in the very nature of the NURBS geometry, the second through blocks and finally when recording command history.

Note: Transformations and parametric design

Certain parameters are common in all digital tools. All modeling tools allow rotating, scaling, stretching and changing the geometry location. Such operations are called “transformations”. The angle is the parameter for rotation, the scale factor is the parameter for scaling, and so on. Rhino supports all basic transformations, but also a rich collection of advanced ones. Geometry can be stretched, twisted, bent, and flown along a curve or a surface. All Transformations are listed under the Transform menu.

NURBS geometry is parametric

One of the main appeals of NURBS modeling is that it defines geometry using parametric curves and surfaces that are easy to define and manipulate. One of the parameters that describes NURBS curves is called “control points”. Once you create a curve, these points can be dragged to modify the curve intuitively. The same thing is true for NURBS surfaces.

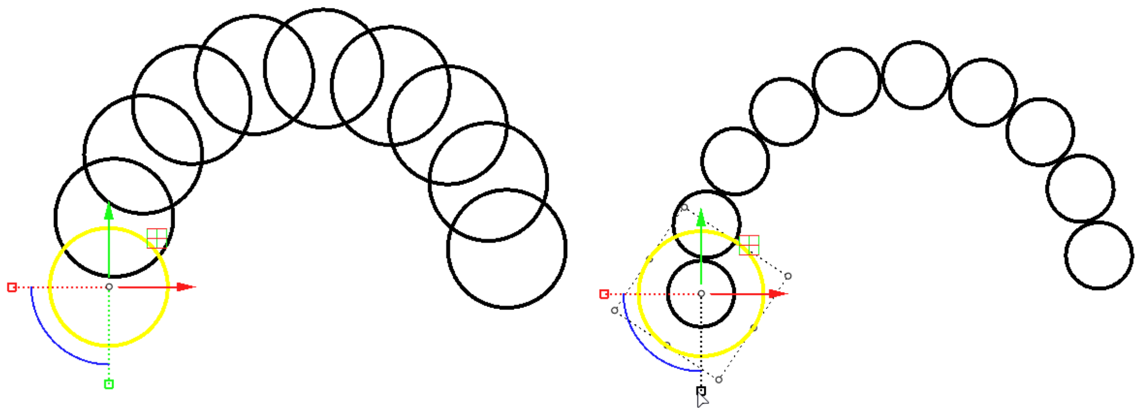


Control points in NURBS geometry as parameters

Note: What is NURBS geometry?
NURBS stands for non-uniform rational b-splines. A detailed description of parameters curves including NURBS is described in the “Essential Mathematics for Computational Design”: http://developer.rhino3d.com/guides/general/essential-mathematics/parametric-curves-surfaces/

Blocks as parameters

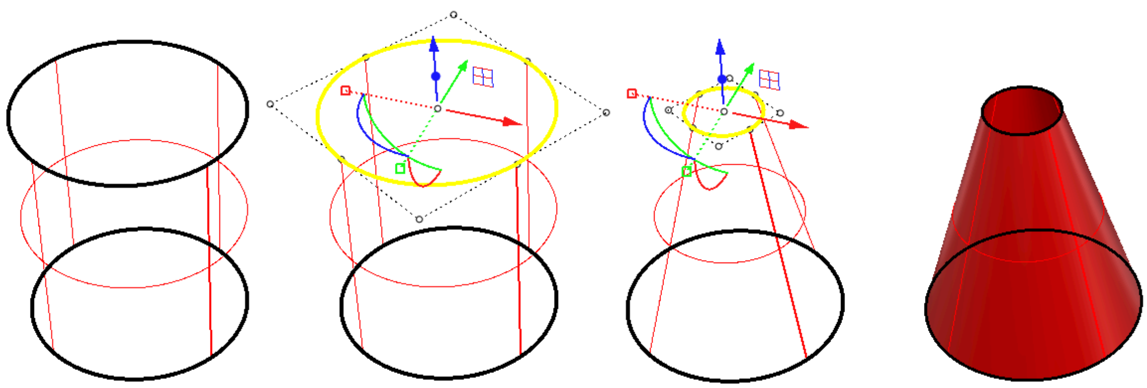
Rhinoceros supports blocks. Once a block is defined, any number of instances can be placed in the model. They all refer to the original block geometry and update when the block is changed.



Blocks as parameters

Command history and parametric workflow

Most Rhinoceros commands support history. History is recorded only when you choose to *Record History*. Input geometry becomes parameters. For example, if you record history before extruding the base curve of the tower, you can change the base curve to control the extruded form.



Using ***History*** in Rhino is a form of parametric design

Resource: History recording in Rhino
History tutorial: https://vimeo.com/261535716

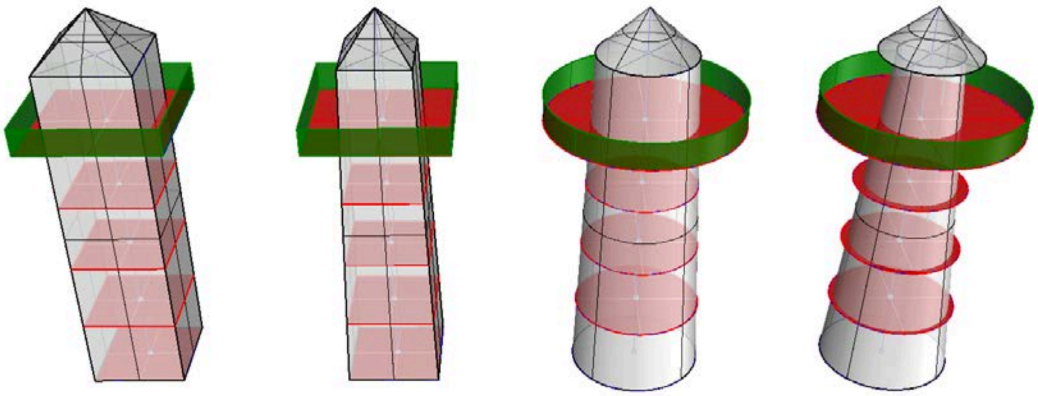
Tutorial to compare modeling methods

All modeling methods are widely used in architectural design, and they can complement each other. Each has its strengths and utility within the design and building workflow. Designers well versed in all methods are usually more productive and competitive.

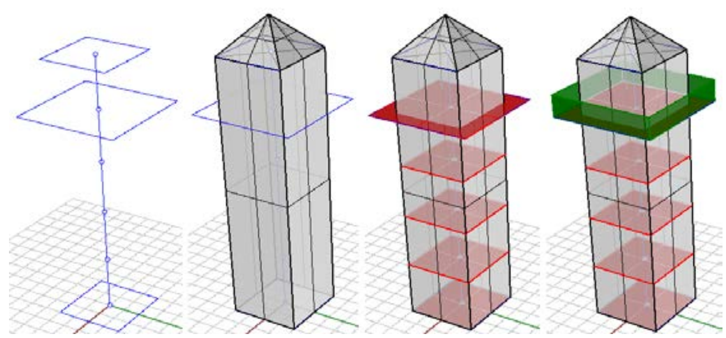
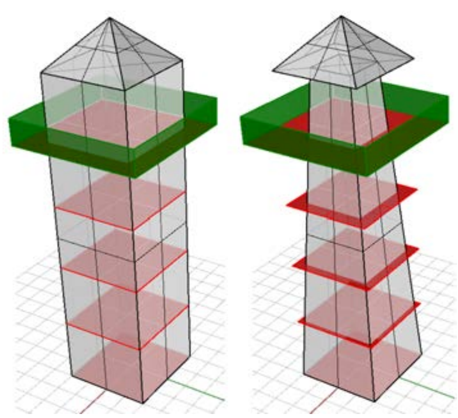
To gain an appreciation of all three modeling methods, we will model the lighthouse using Rhino, Grasshopper and VisualArq (Rhino plugin), and compare strengths and shortcomings.

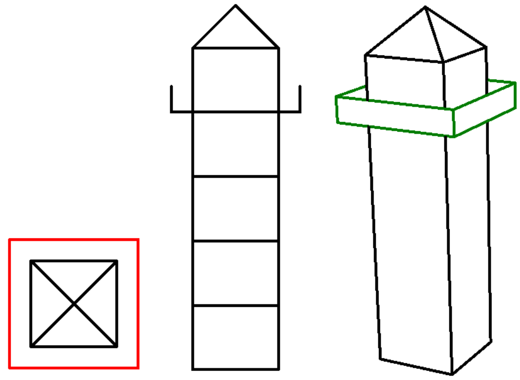
Direct Modeling using Rhino

Rhino modeling allows quick creation of geometry and ability to reflect and recreate without having to assign too many details or make material and other decisions early in the process. It is well suited for concept exploration and when the remodeling of variations is not cupersome.



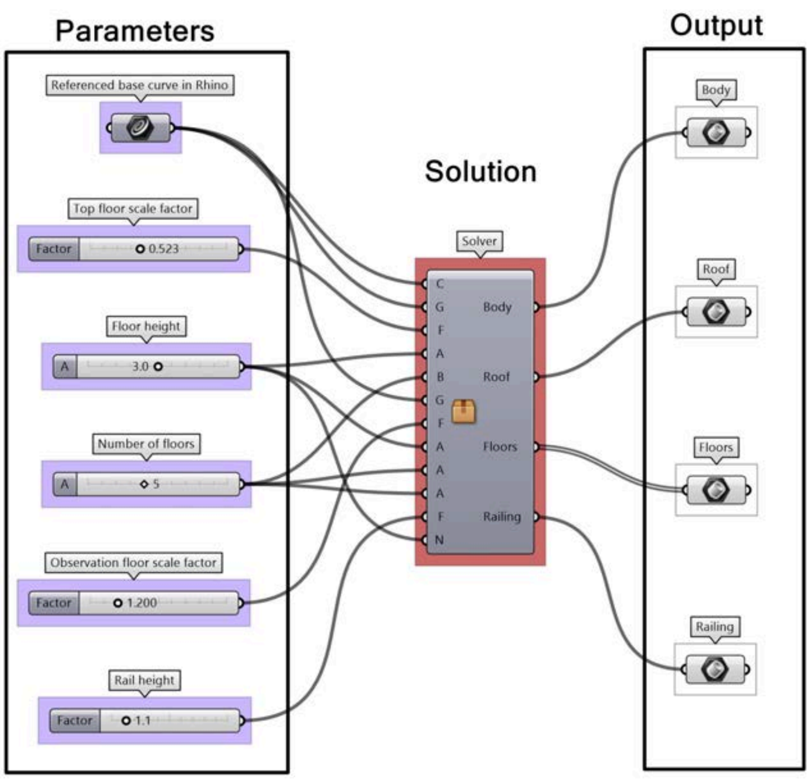
Modeling steps

Create reference geometry for the building shape and levels. Reference is very useful to set the scale the model and define rough outline Initial form can be quickly created with commands such as [Rectangle](), [Box](), [Plane]() and [ExtrudeCrv]().	
Changing the form involves some remodeling No detailed decisions are made about materials, building specifications and others at this stage	

Extract basic 2D drawings using Section and Make2D commands. Detailed models and documentation that includes wall thicknesses, openings and materials require additional modeling steps.	

Algorithmic Modeling using Grasshopper

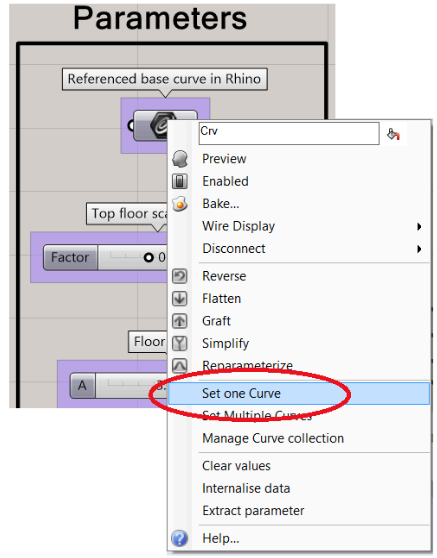
Parametric design is well suited for parameter-based modeling where the logic of creating the model is explicitly documented. The biggest advantage is to be able to change initial parameters and have the full model update without any need for remodeling. Parametric environments are also well suited for algorithm based modeling.



Modeling steps

Parameters can include the base curve, height, number of floors and if the walls are straight or tapered.

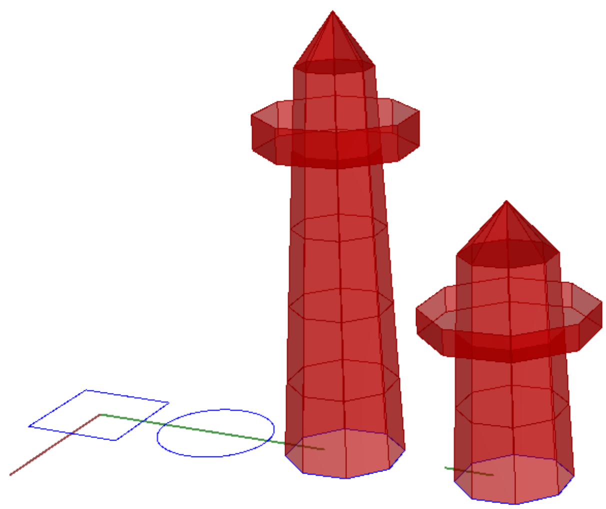
The base curve can be referenced from the modeling environment



You can create variations by changing the base curve and parameters.

All geometry is created as a preview. You can Bake output to create Rhino geometry.

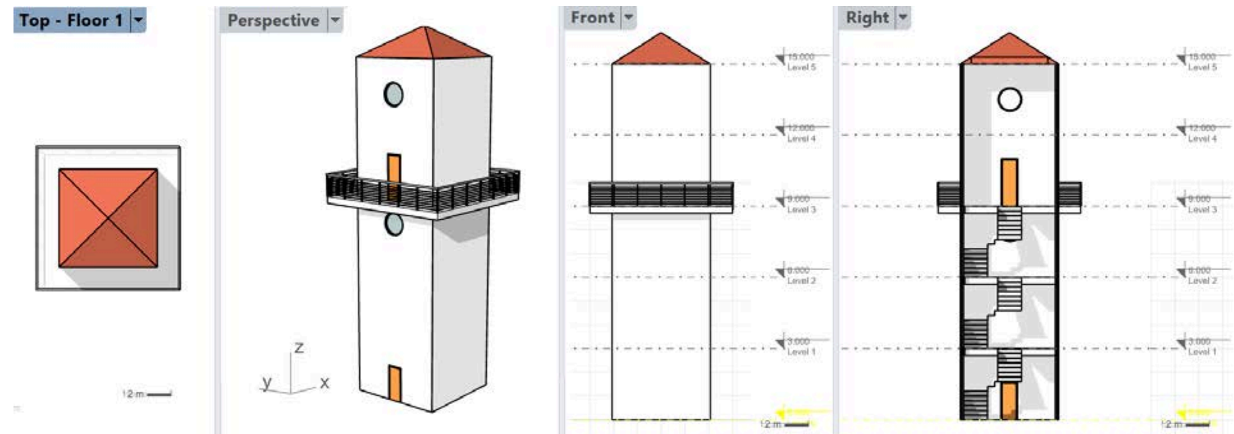
More sophisticated parametric models can include full details of the building, routines to generate and extract panels and connect to advanced analysis plugins to design the structure and respond to environmental constraints for example.



Object-Based modeling using VisualARQ

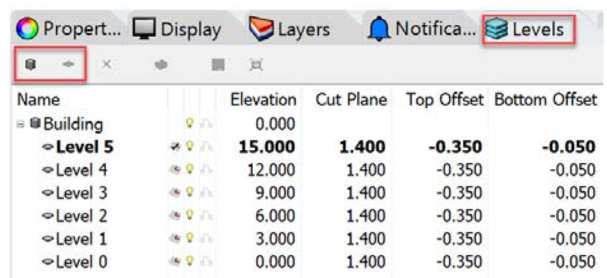
Object based modeling is very intuitive to create specific types of buildings and can be optimized to use inhouse building codes and specifications. It is most suitable as an assembly environment that uses building parts to create models.

Parameters are embedded in the building parts and models can be updated quickly. Documentation and schedules can be extracted quickly once the model is completed.



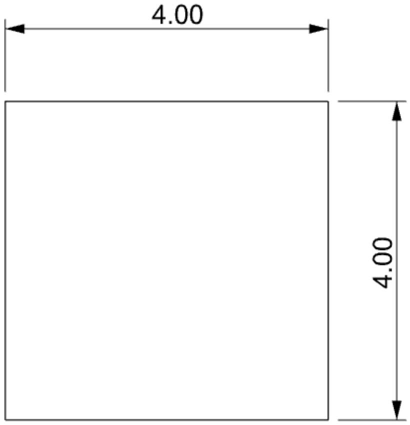
Modeling steps

Open a new VisualARQ Template - Meters.
Go to the Levels Panel.
Start adding Levels, make sure the building layer is highlighted.
Each level has its own CPlane and elevation height.

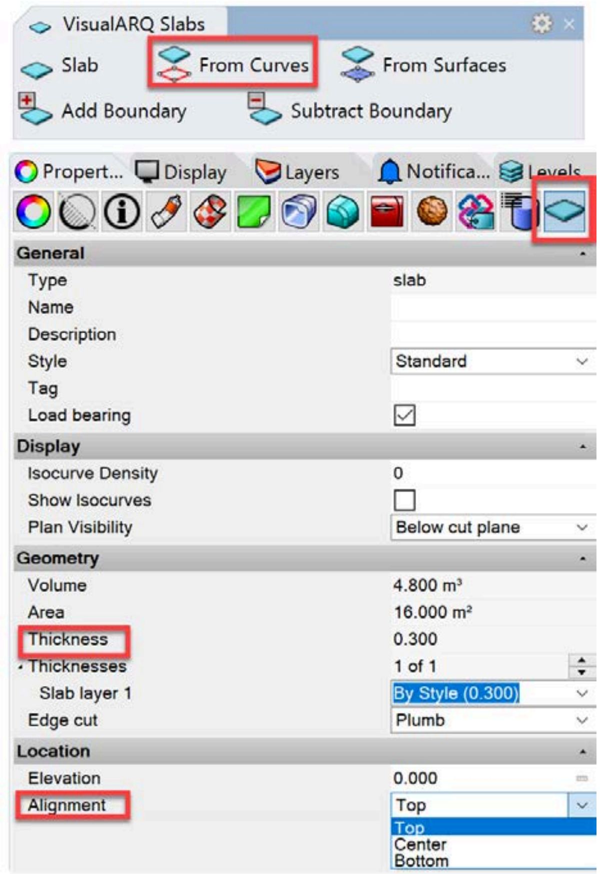


Name	Elevation	Cut Plane	Top Offset	Bottom Offset
Building	0.000			
Level 5	**15.000**	**1.400**	**-0.350**	**-0.050**
Level 4	12.000	1.400	-0.350	-0.050
Level 3	9.000	1.400	-0.350	-0.050
Level 2	6.000	1.400	-0.350	-0.050
Level 1	3.000	1.400	-0.350	-0.050
Level 0	0.000	1.400	-0.350	-0.050

Make sure Level 1 Cplane is active.
Draw the lighthouse plan.

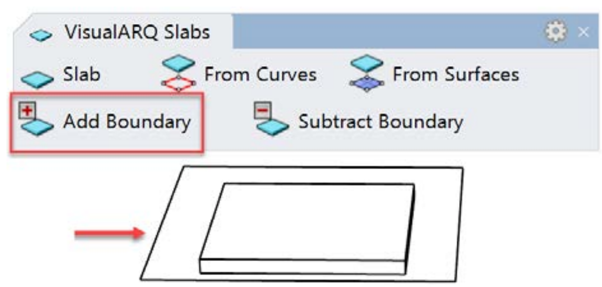


Now select the plan curve and choose (vaWall) and choose (From Curve) option on the command line.



Copy the slab to all four levels.
On the fourth level offset the curve 2m.

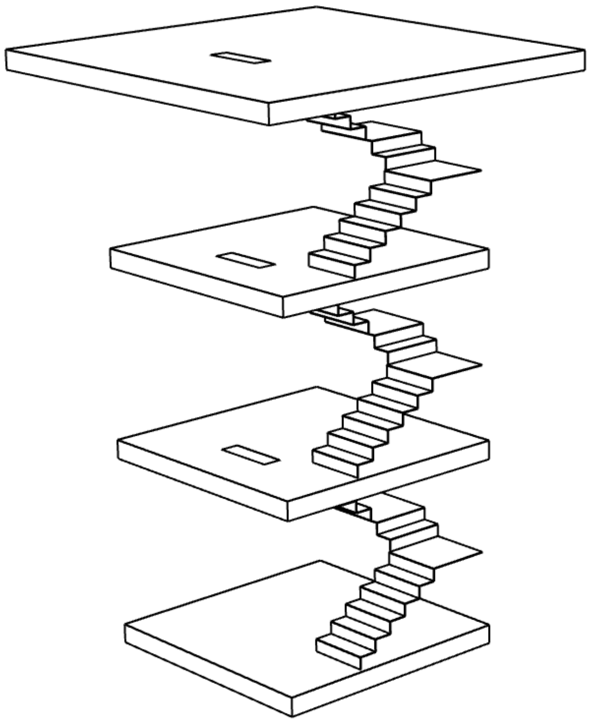
From the slab menu, choose (Add boundary) this will extend the slab to the desired curve.

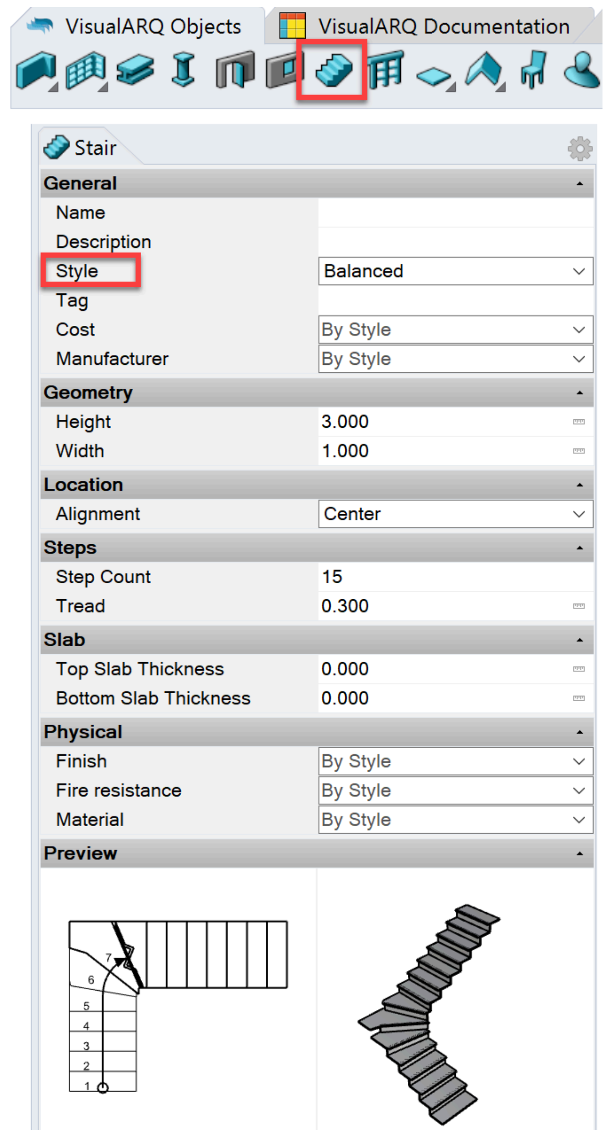


Click on the stairs from the VisualARQ tab.
Follow the command line to complete the stairs.
Note: You can always turn the control points on to change the stair's insertion point.

From the properties panel, you can change the stairs' style.

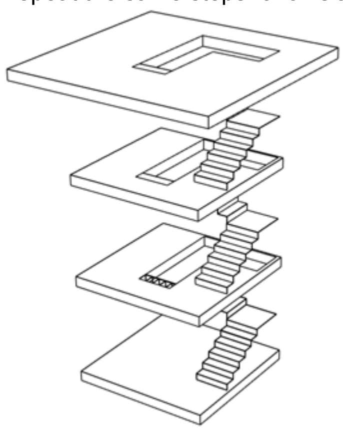
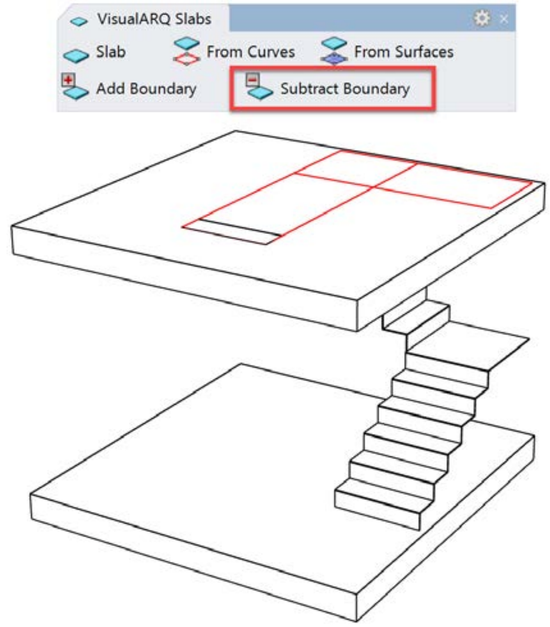
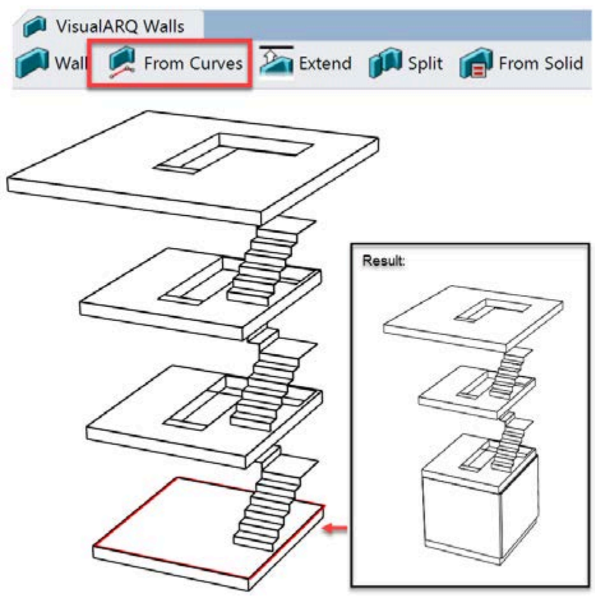
Copy the stairs to all floors.



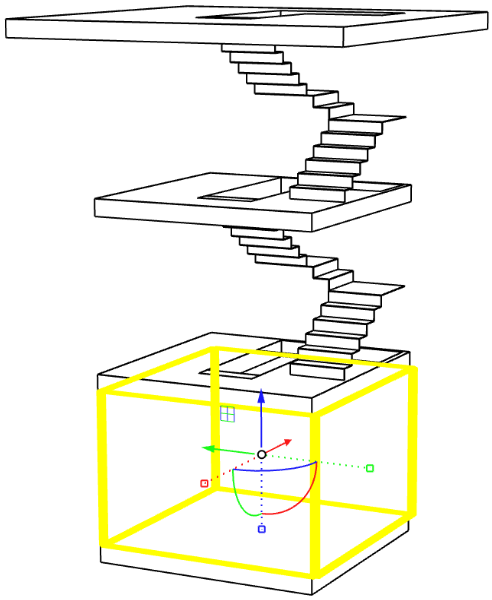


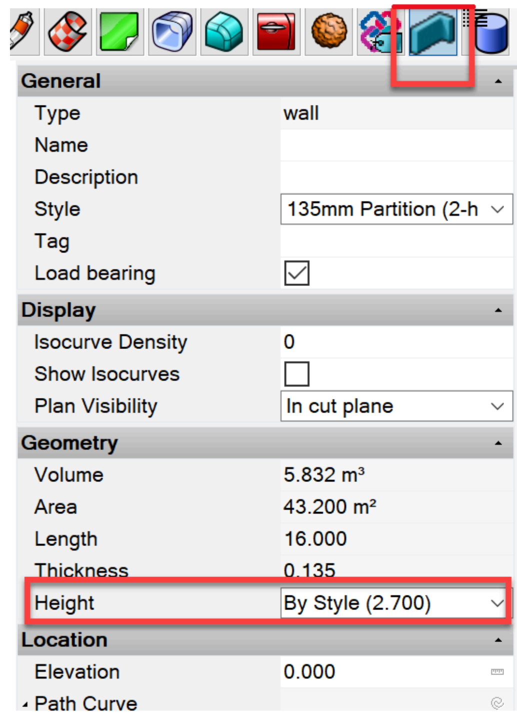
<table>
<tr>
<td>Now we need to make an opening in the slab.
Draw a planar curve with the shape of the opening to make the cut.

From the slab menu select (Subtract Boundary), choose the slab then the planar curves to make the cut.

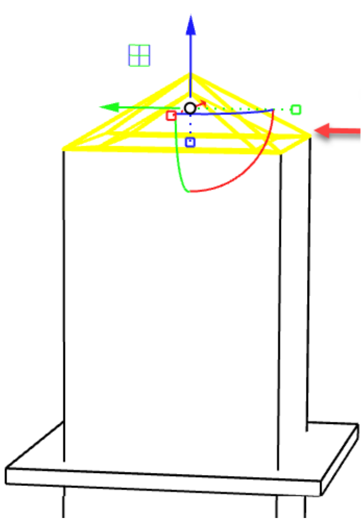
Repeat the same steps for all slabs:
</td>
<td>
</td>
</tr>
<tr>
<td>Select the plan curve from Level 0.
Click on (vaWall) Wall from curves.</td>
<td>
</td>
</tr>
</table>

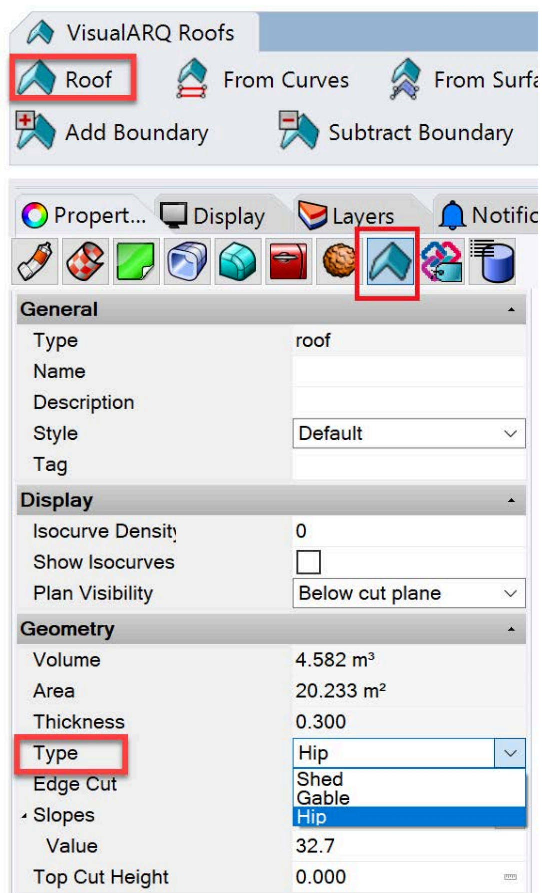
Now highlight the walls and change their height from the wall properties to 15m.



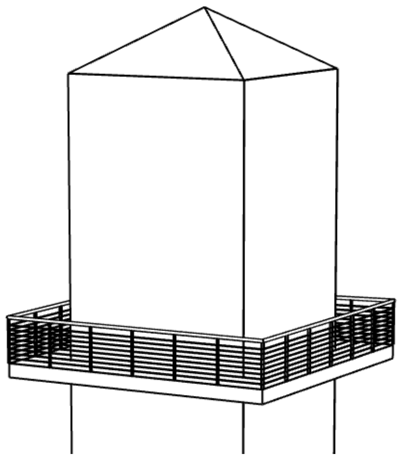


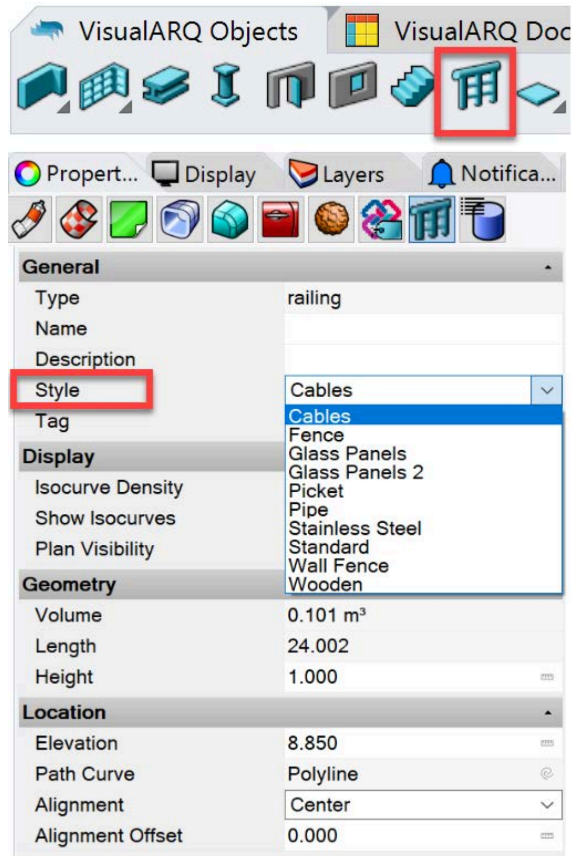
Now create the roof.
From the roof menu, select (Roof) then start selecting the corners of the walls.





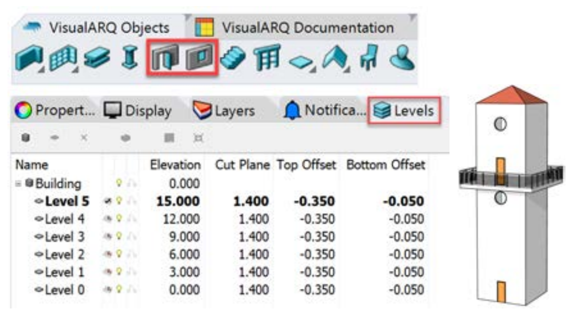
Add the railings.
Select the railing option then start selecting the corners of the slab.
From the properties panel, change the railing style.

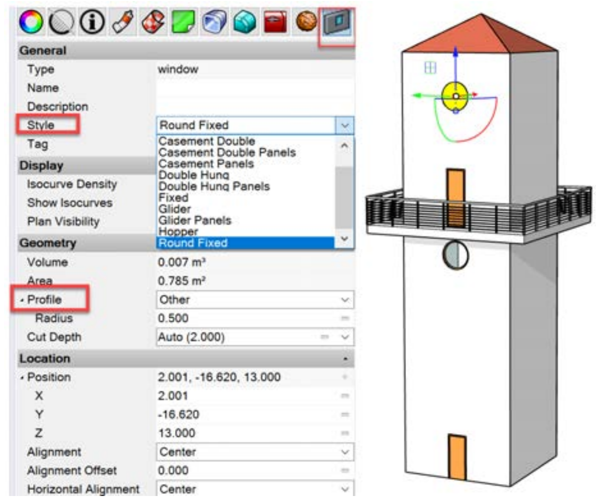




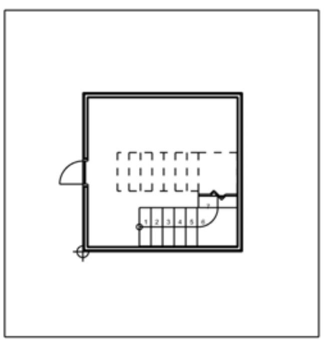
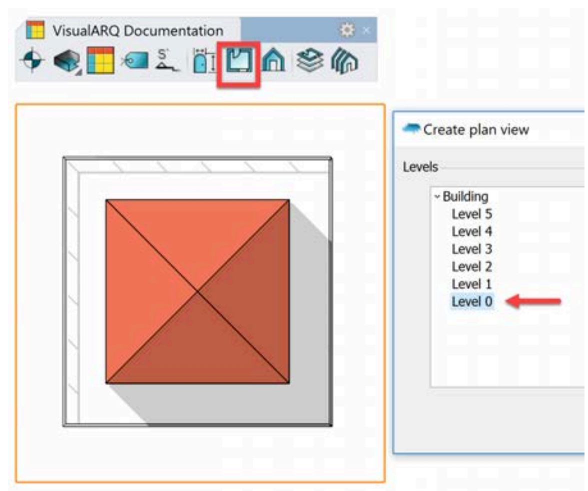
Select the door option.
Make sure you are on Level 0 then add the door in the center of the wall.
Next, select the direction you want the door to open.

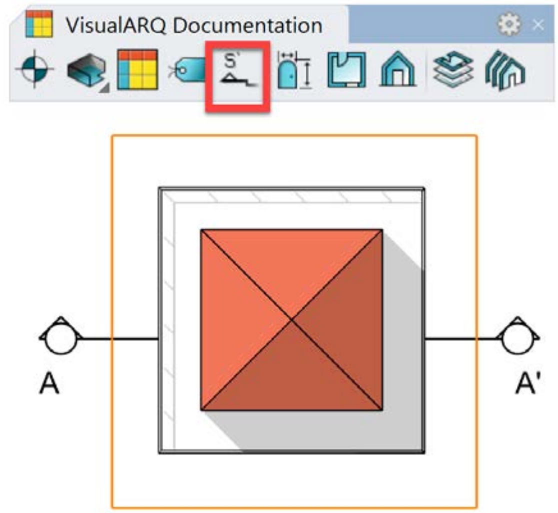
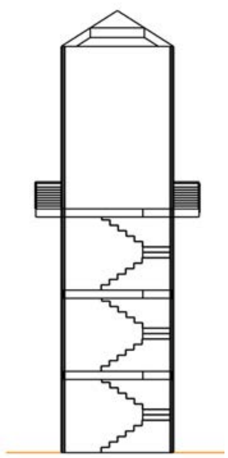
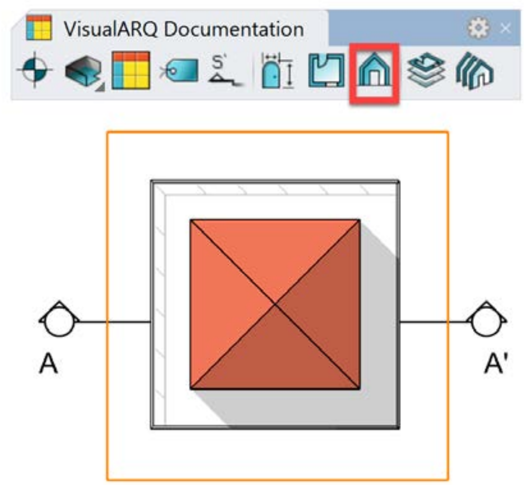
Now do the same for the windows.



<table>
<tr>
<td>Click on the door or windows and change their styles and profiles from the default drop-down menu.</td>
<td>
</td>
</tr>
<tr>
<td>Extract 2D documentation.

Plan:

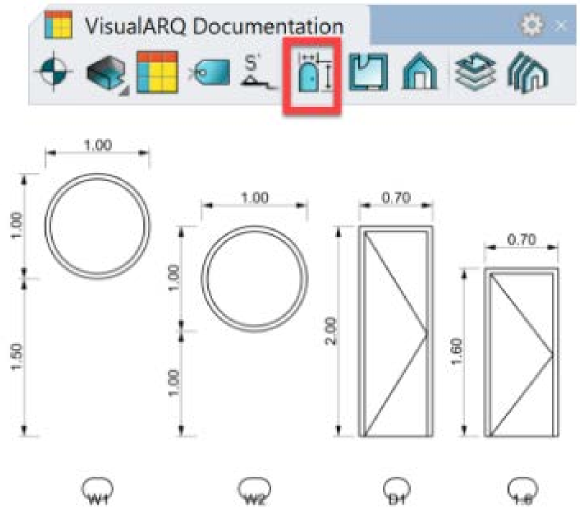
From VA Documentation tab, Select Plan view, specify style and level.
Drag a rectangular around the building, then click for an insertion point.
Do the same for all Levels.

Level 0</td>
<td>
</td>
</tr>
</table>

Section: Under the same tab select Section, then pick a style (Arrow) in this case. Then specify two points for the section cutting line. Type in a name for the section on the command line then hit enter.	
Now for VA Documentation Tabs click on Section View. Pick the Section arrow for the previous step and select an insertion point.  A - A'	

Openings Elevation:

Select (Opening Elevation) from the VA Documentation tab.
Select all openings, then specify and start and end point.

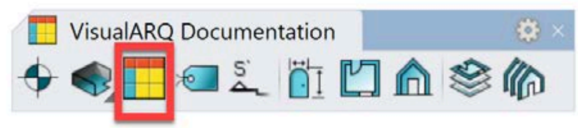
*Note: you can drag a window around the model to select all openings.



Tables:

Select (Table) under the VA Documentation tab.
Choose a table style, I choose openings for this exercise.

Now select all the openings in the model.
Specify where you want to insert the table.



	Openings					
Quantity	Type	Reference	Width	Height	Elevation	Opening Side
1	Window	W2	1.000	1.000	13.000	Left
1	Window	W1	1.000	1.000	7.500	Left
1	Door	D1	0.700	2.000	9.000	Left
1	Door	1.6	0.700	1.600	0.000	Left

Resources

Support

- Rhino Support Forum: https://discourse.mcneel.com/ - the best place to ask questions and meet with other Rhino users.
- Rhino email support: tech@mcneel.com
- Rhino support page: https://www.rhino3d.com/support
- Plugins for Rhino: http://www.food4rhino.com/

Workflows

- Digital Fabrication tutorials for Rhino: https://www.rhino3d.com/tutorials#digital_fabrication
- Architecture Digital Fabrication: https://wiki.mcneel.com/rhino/architecture/home#digital_fabrication
- Rhino Fab Studio: http://www.rhinofablab.com/
- Using Rhino with Revit: https://wiki.mcneel.com/rhino/architecture/bim/rhino-to-revit
- Nick Senske Computational Design Courses YouTube : https://www.youtube.com/user/nsenske
- Jose Sanchez would like to share a series of online videos on Rhino modeling in architecture: Modeling the LF-ONE by Zaha Hadid : https://www.plethora-project.com/education/

Rhino in Architecture

- Many Articles, Books, Workflows: https://wiki.mcneel.com/rhino/architecture/home
- ArchDaily blog covering Rhino: https://www.archdaily.com/tag/rhino
- The latest news about Rhino in AEC: http://blog.rhino3d.com/search/label/AEC
- Many videos on Rhino in Architecture: https://www.youtube.com/results?search_query=rhino+architecture
- Gallery of a few buildings done with Rhino and Grasshopper: http://www.grasshopper3d.com/page/architecture-projects

Appendix A Site modeling

Elk for Grasshopper

Elk is a free plugin for ***Grasshopper*** to process GIS data from two sources: the open street maps (OSM) and shuttle radar topography mission (SRTM). ***Elk*** was developed by ***Timothy Logan***.

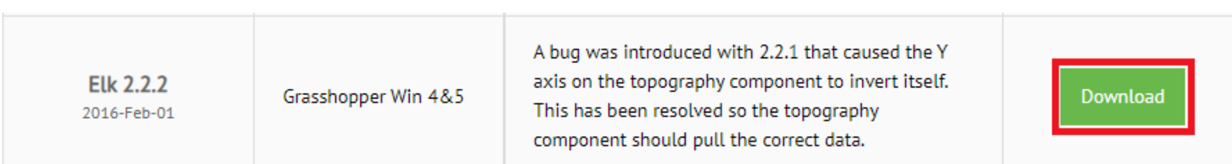
"OpenStretMap.org is an open/crowd sourced website of mapping data. It allows you to export XML formatted data of a selected area and then ***Elk*** will organize and construct collections of point and tag data so that you can begin creating curves and other ***Rhino/Grasshopper*** geometry.

USGS is a science organization that provides access to a large range of scientific data pertaining to the earth. ***Elk*** uses data that originates from the ***Shuttle Radar Topography Mission*** (SRTM) of 2000. This was a shuttle mission where most of the earth was scanned for elevation and packaged in 1°x1° tiles." [3]

Elk is not currently under active development, but it is an open source project. You can find it here: [Link to elk open source project]

Download and setup

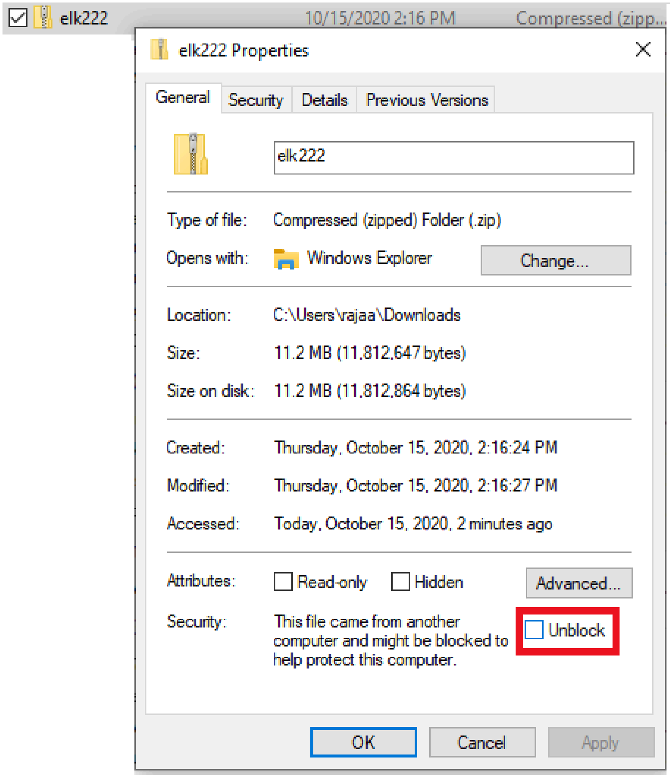
Download the latest ***Elk*** zip file from ***Food4Rhino***. Note that although the last version states that it was built for Rhino 4 and 5, ***Elk*** still runs in Rhino 6.
https://www.food4rhino.com/app/elk



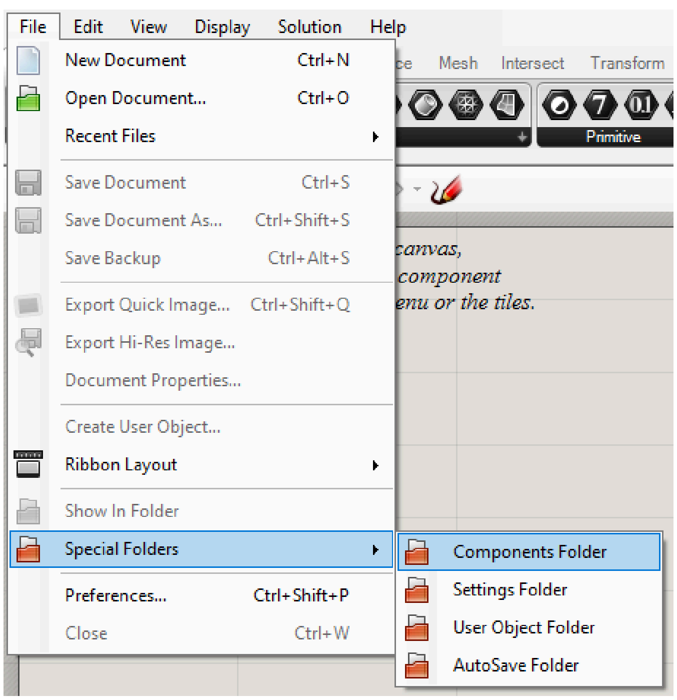
Elk 2.2.2 2016-Feb-01	Grasshopper Win 4&5	A bug was introduced with 2.2.1 that caused the Y axis on the topography component to invert itself. This has been resolved so the topography component should pull the correct data.	Download

Make sure to ***Unlock*** the downloaded zip file before extracting the content by right-mouse-click, then go to ***Properties***. In the dialog, check the ***Unlock*** checkbox, then click ***Apply***:

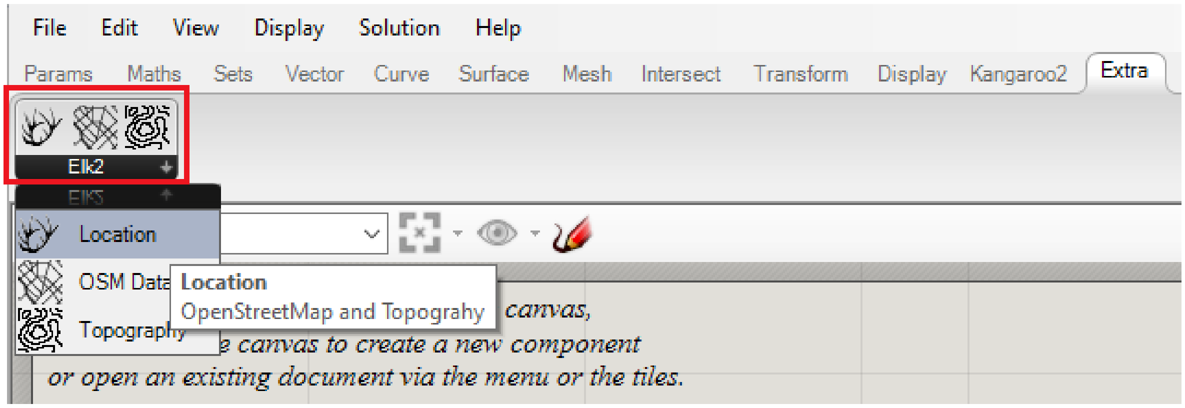
[3] From Elk download site: https://www.food4rhino.com/app/elk



Move the downloaded and unlocked ***Elk*** zip file file to the ***Grasshopper*** component libraries folder and extract all files (you can open it through an open session in ***Grasshopper*** under ***File>Special Folders> Components Folder***.

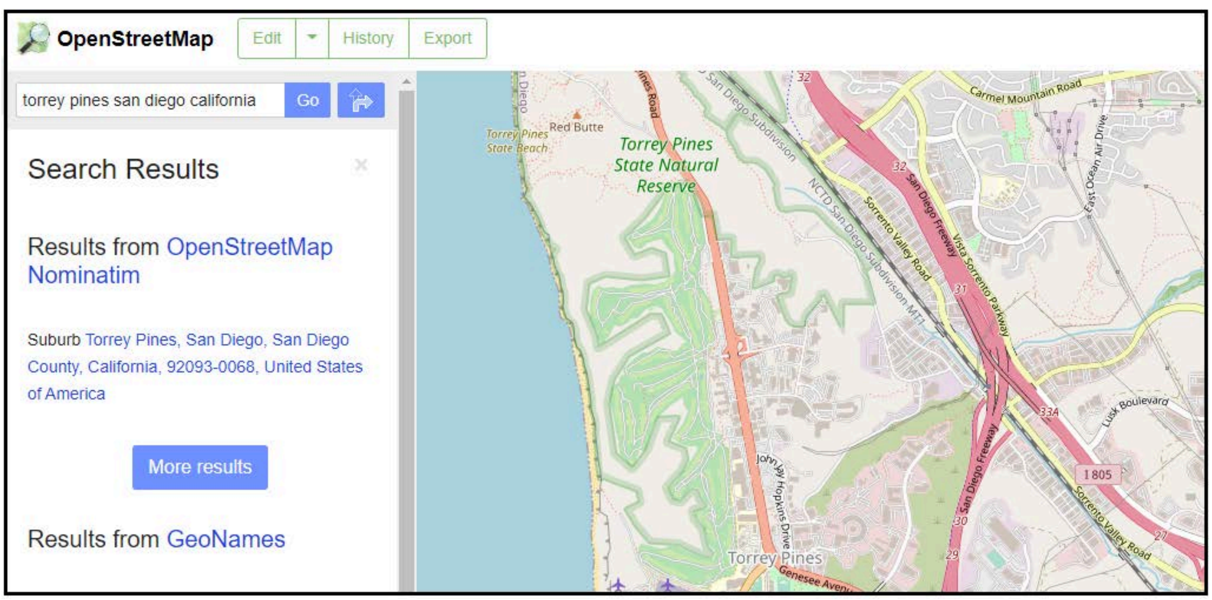


Open Grasshopper, and you will see a new ***Extra*** tab with ***Elk*** components inside:

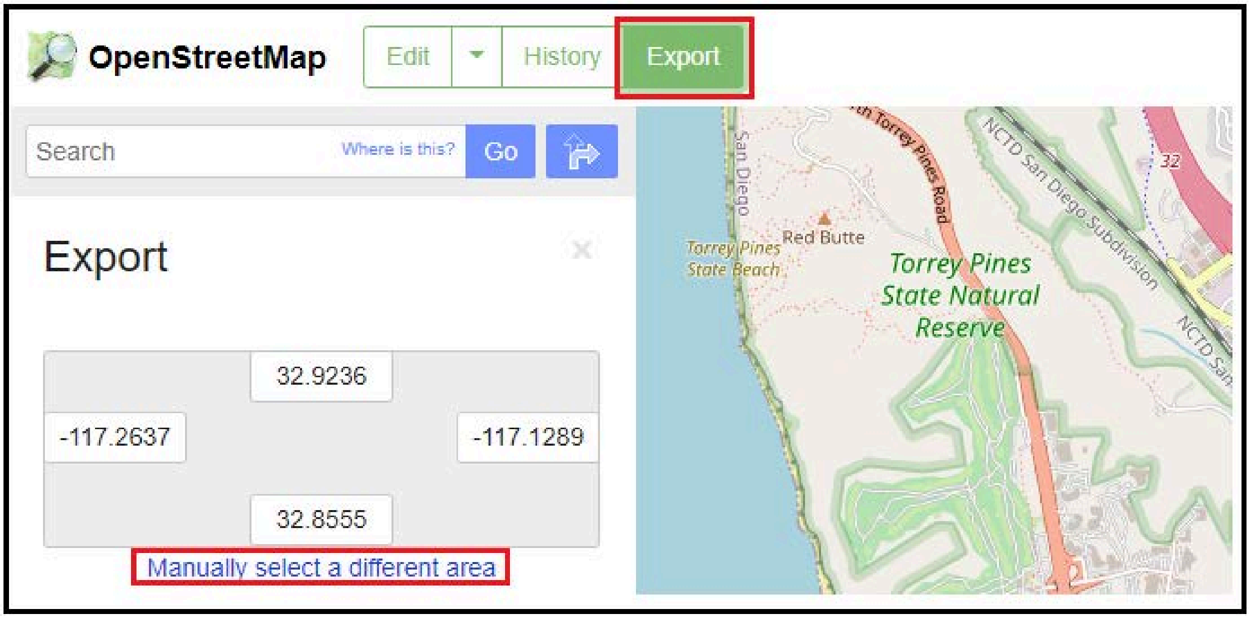


Elk Workflow

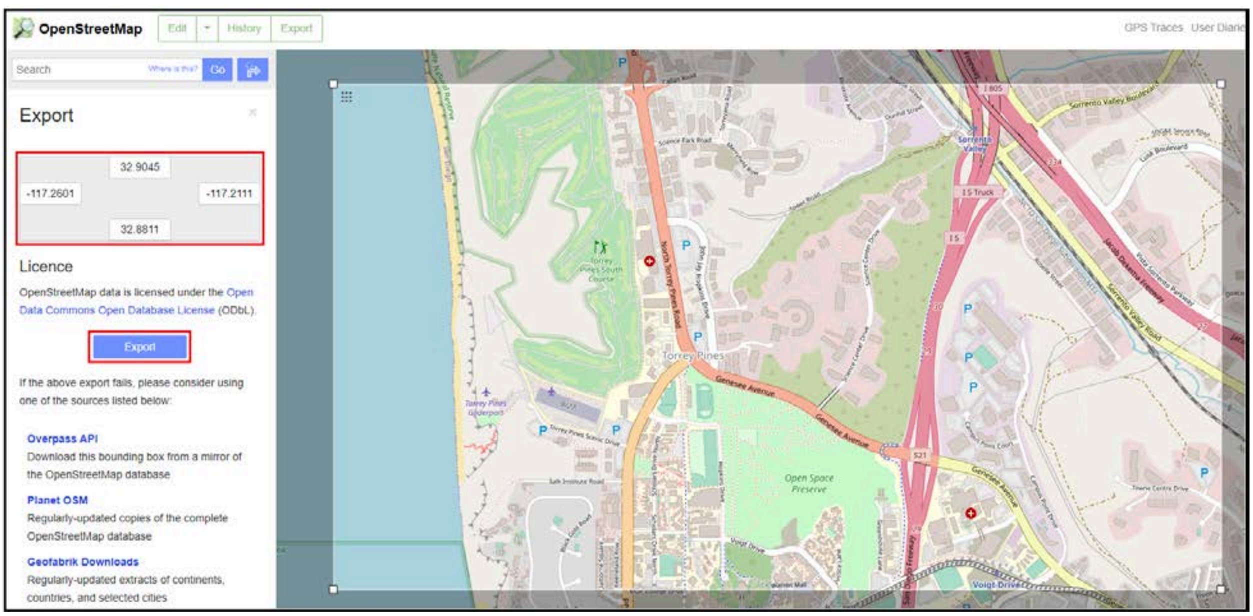
The first step is to download the open street map (***osm***) file for the area you are interested in. All maps are located in https://www.openstreetmap.org/ which is an open source website and anyone can edit and download at any location around the world. Here are the general steps to get the ***osm*** file for the ***Torrey Pines*** location in ***San Diego***.
First go to https://www.openstreetmap.org and search for "***Torrey Pines San Diego California***". You will get the following map:



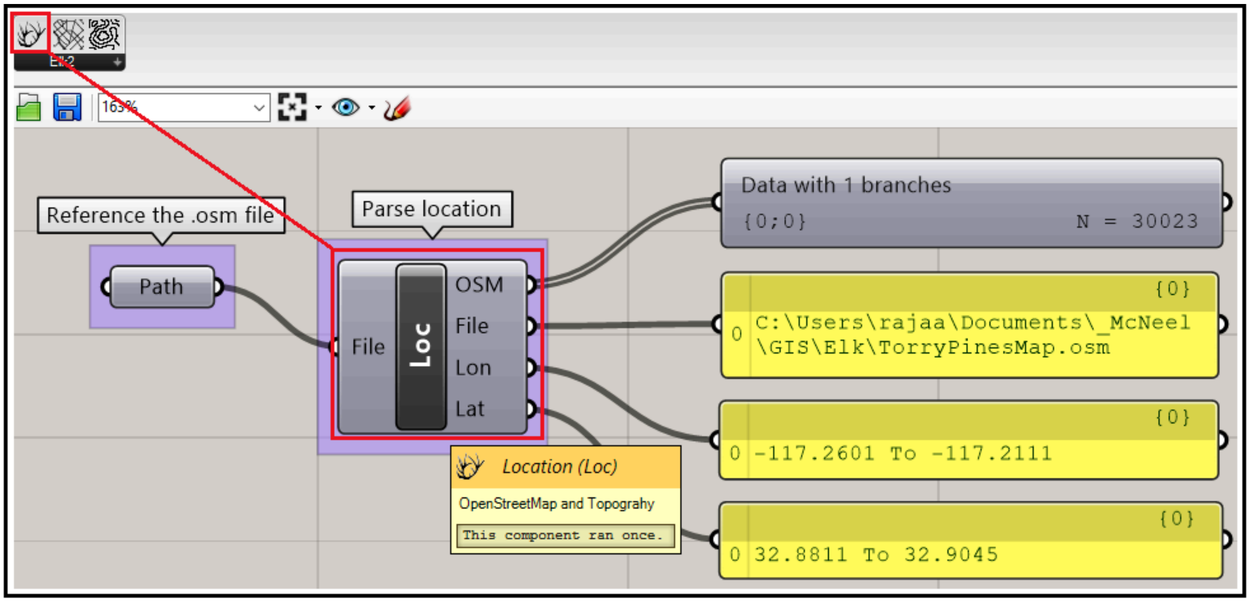
Click ***Export*** to view the exact latitude and longitude of the area visible on the screen. You can specify a region within your map by clicking on and directly editing the lat/long numbers or by clicking on ***Manually select a different area*** to window select the area of interest.



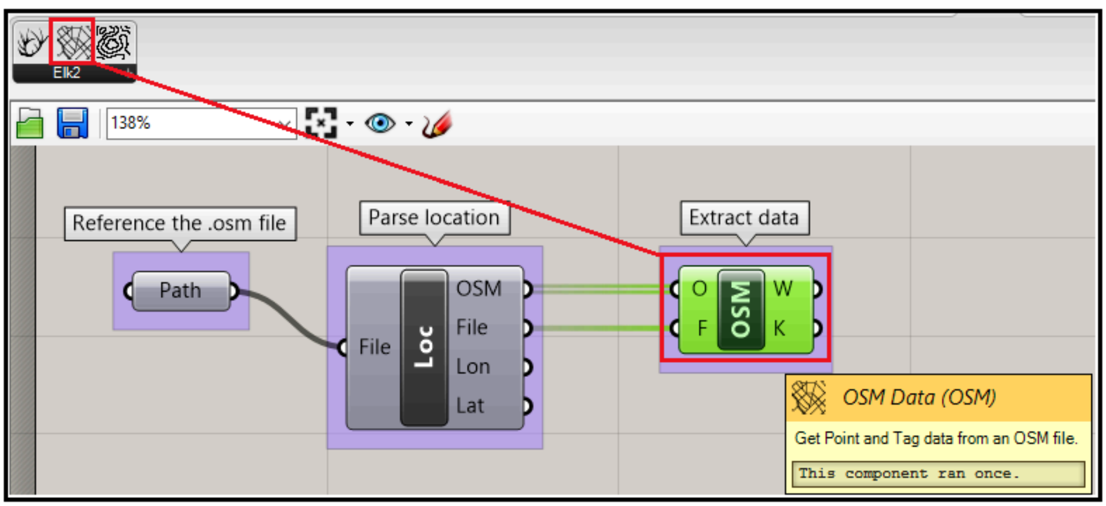
Once you select a region or exact coordinates, you can click ***Export*** to download the ***maps.osm*** file which is the file that ***Elk*** uses.



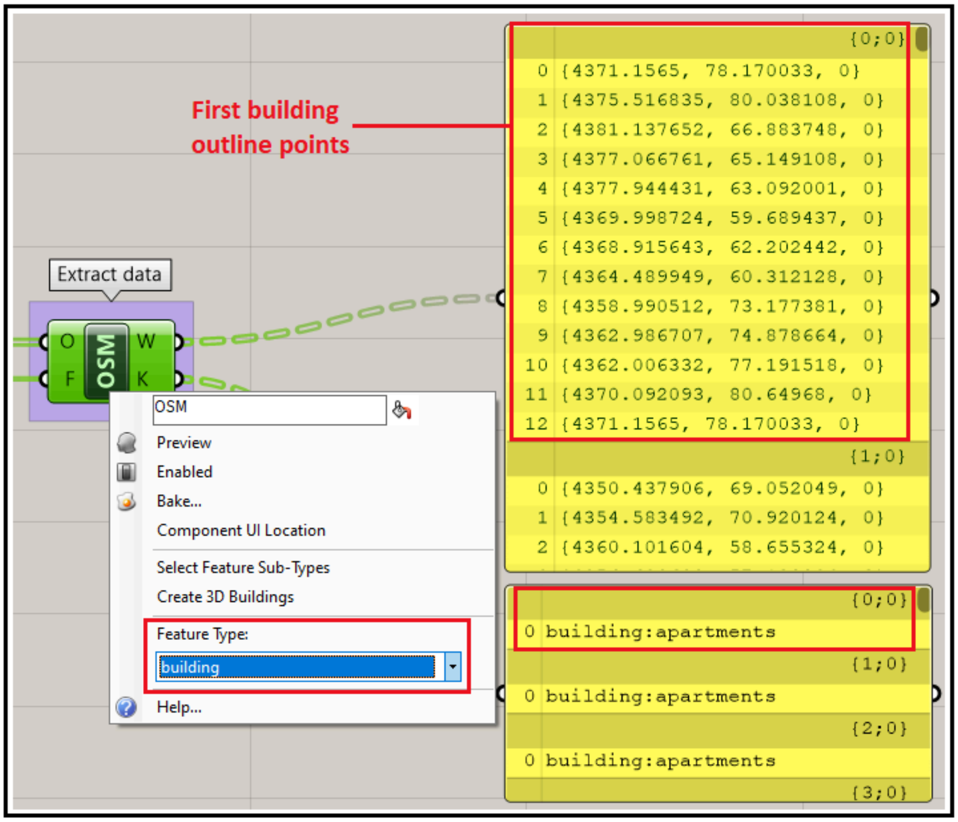
Using Elk in Grasshopper, the ***Location*** component helps import the ***osm*** data. The input in the osm file path and the output include the latitude and longitude and the osm-points which are 3D points with an osm defined IDs.



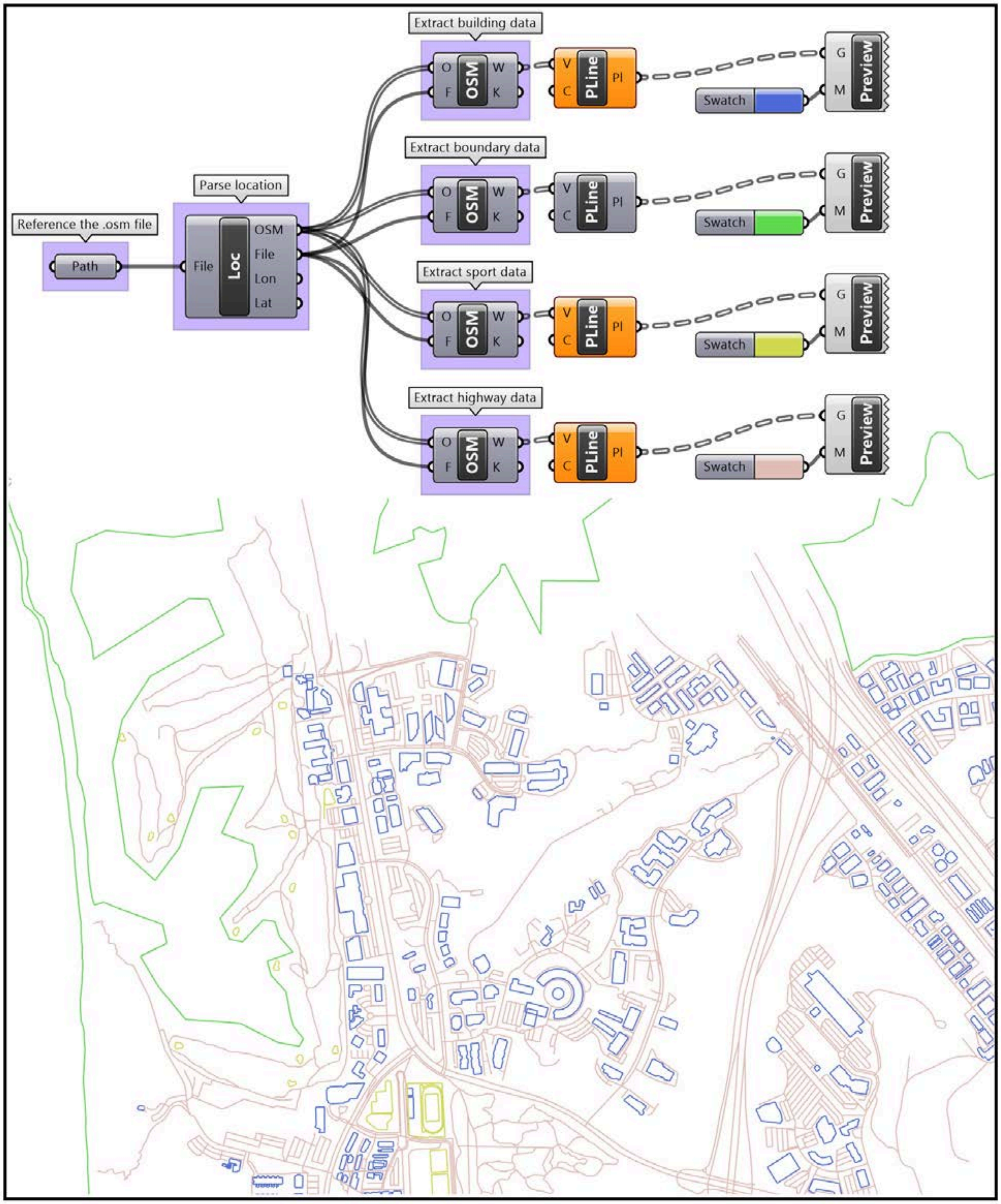
Next, you need The ***OSMData*** component to start organizing and collecting the data from the ***OSM*** file. It takes the ***OSM*** and ***File*** as inputs from the ***Location*** component and outputs one specified feature in the component itself.



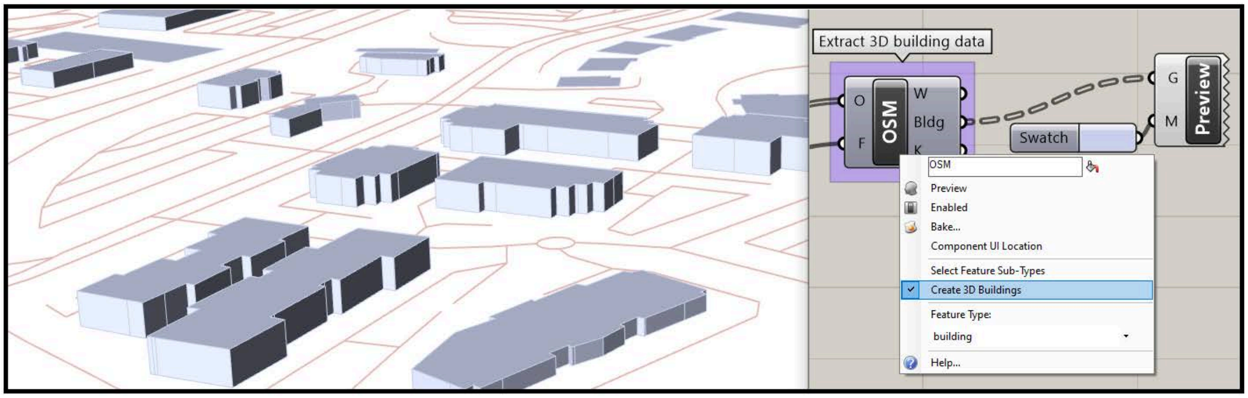
The feature type is set Building by default, and it outputs all building types. The first output includes the post set that represents the outline of the building and the second is a list of information about each building. It can include only a description, but sometimes it also includes other information such as the building type, source of information and height, but that varies among different buildings in the list.



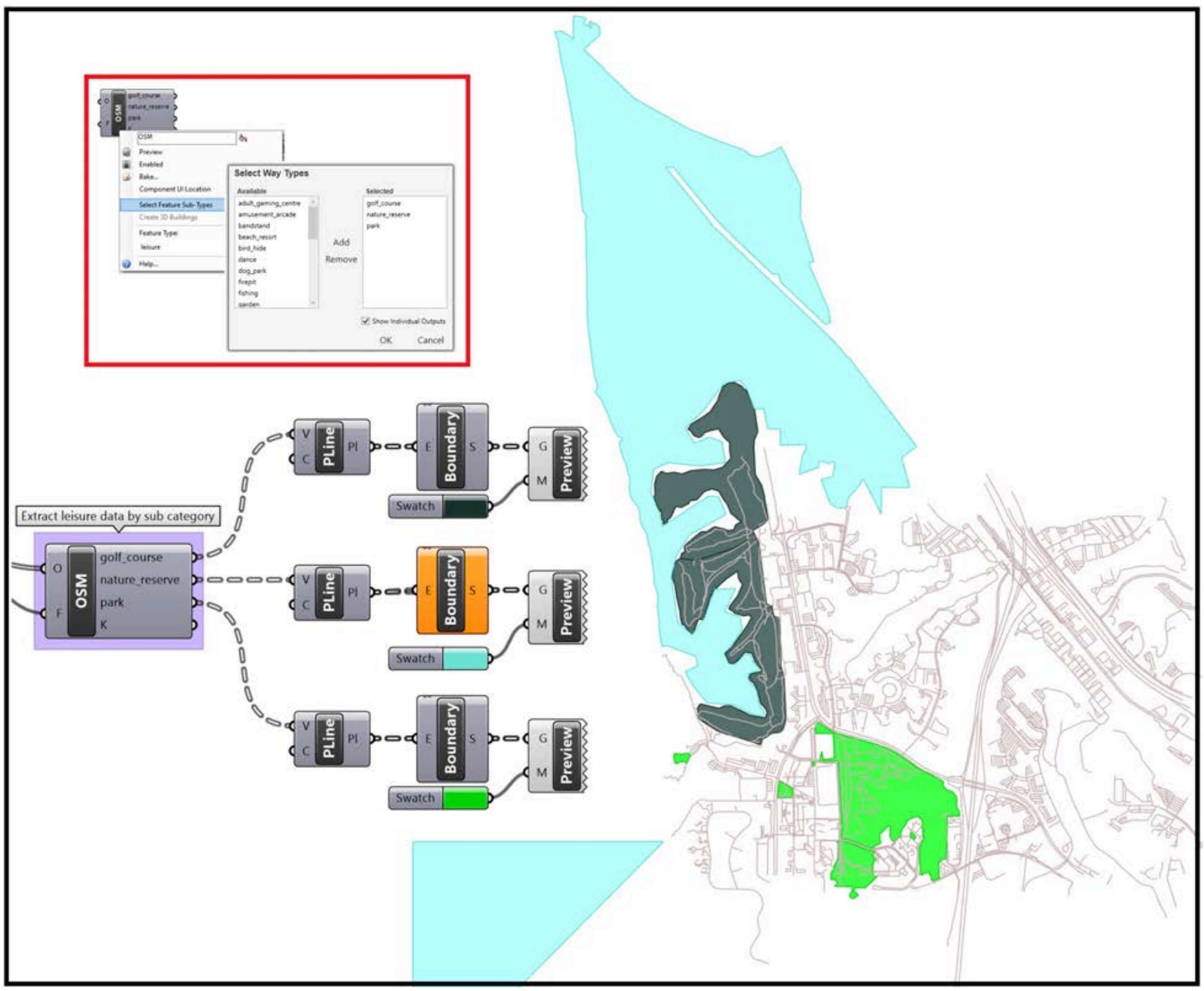
You can use multiple ***Elk OSM*** components and set to different features. You can also use the ***GH PLine*** component to connect the points of each output. Sometimes the output consists of one point and the ***PLine*** component might give a warning in that case, but that does not affect the output for the rest of the list.



The OSM component is highly customizable and offers different output options based on what ***Feature Type*** it is set to through the component menu. For example, if the ***Feature Type*** is set to ***buildings***, then you can add an output for the 3D building geometry. Notice that not all buildings come with height information and hence some remain flat.

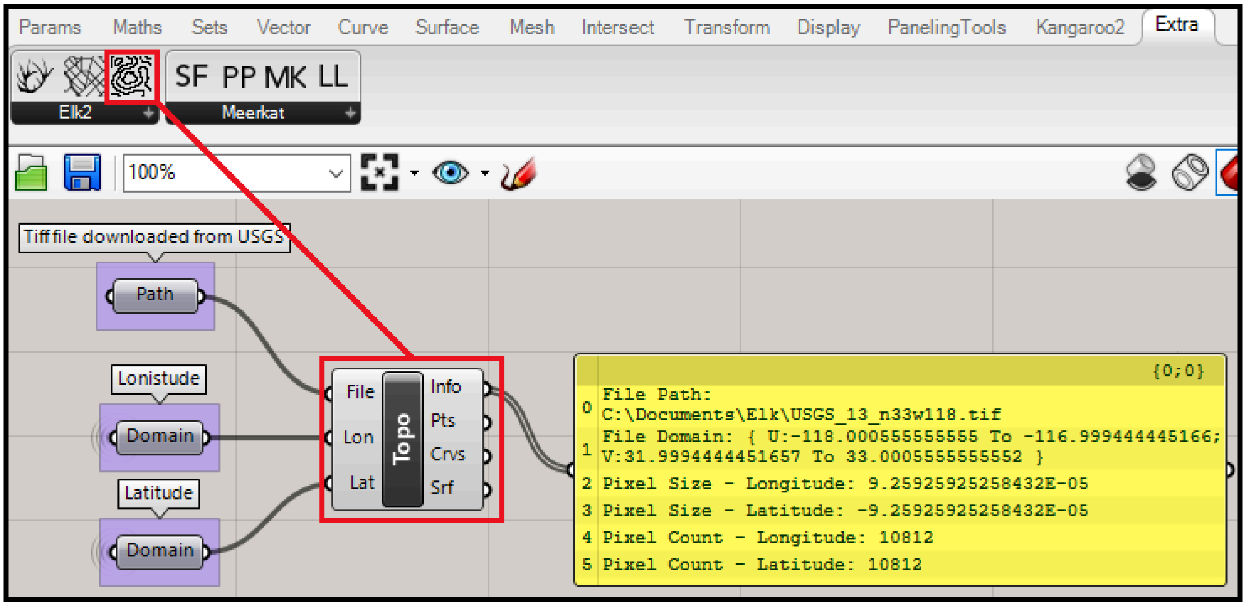


The OSM component allows setting a ***Sub-Types*** for any given ***Feature Type***. For example, the feature type ***building*** includes ***sub-types*** such as ***commercial*** and ***residential***. In the following example, we extract golf courses, nature reserves and parks from the ***leisure*** feature type then color-code them.



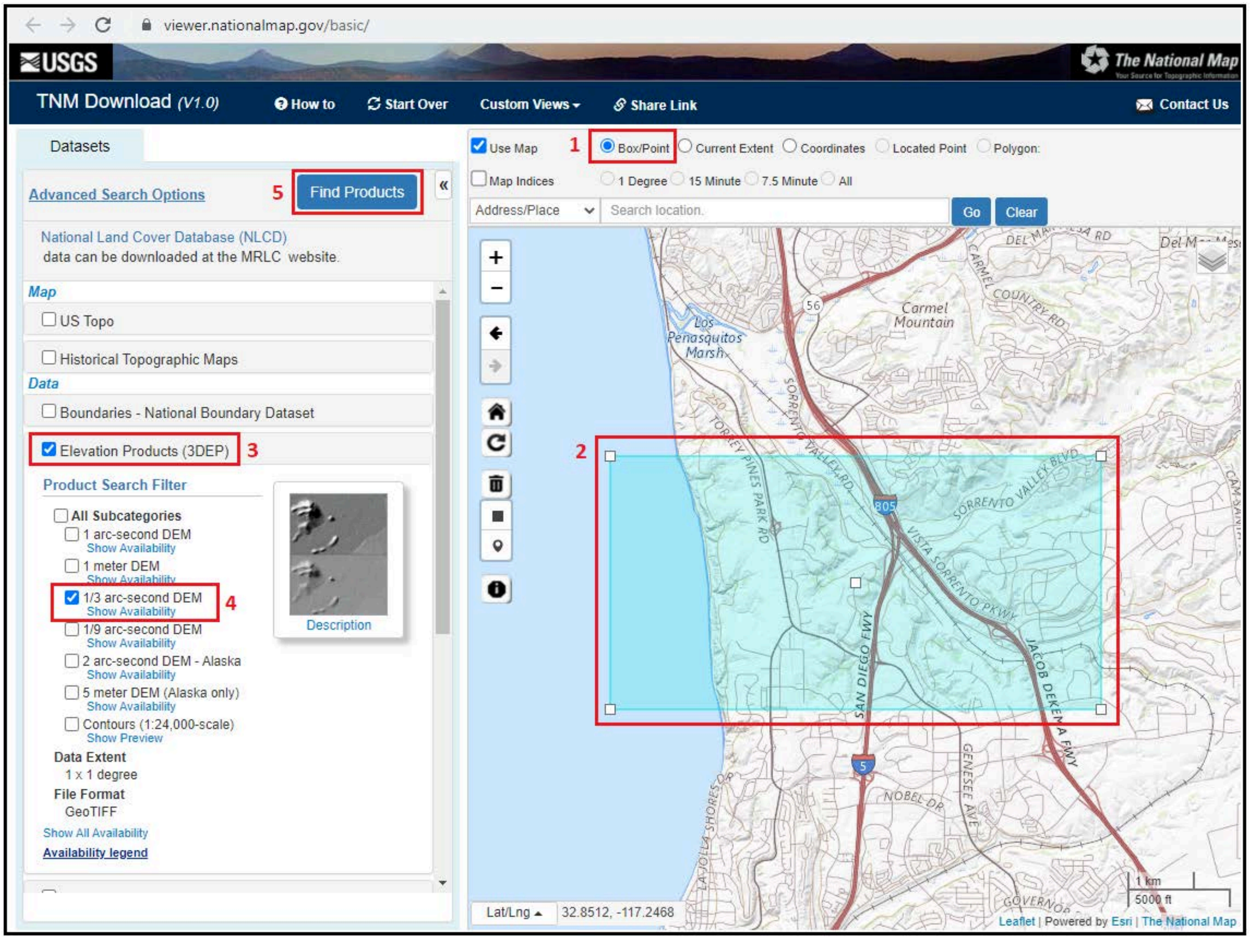
The third major component in ***Elk*** is the topography component (***Topo***) that uses data from multiple sources using three file formats. The first is from the ***United States Geological Survey*** (USGS) which uses ***.img*** file format. The second is related to image files and is called ***GeoTiff*** with data accessed from ***USGS Earth Explorer***. Last, you can use digital elevation models from the ***Shuttle Radar Topography Mission*** (SRTM) where most of the earth was scanned for elevation and packaged in 1°x1° tiles that you can get from and the file format is ***.hgt***. This is where the topology files is found:

1. ***IMG*** and ***GroTiff*** Inside the ***USA*** Only: https://viewer.nationalmap.gov/basic/
2. ***Global IMG*** and ***GroTiff*** data: https://earthexplorer.usgs.gov/
3. **HGT** files: https://www2.jpl.nasa.gov/srtm/



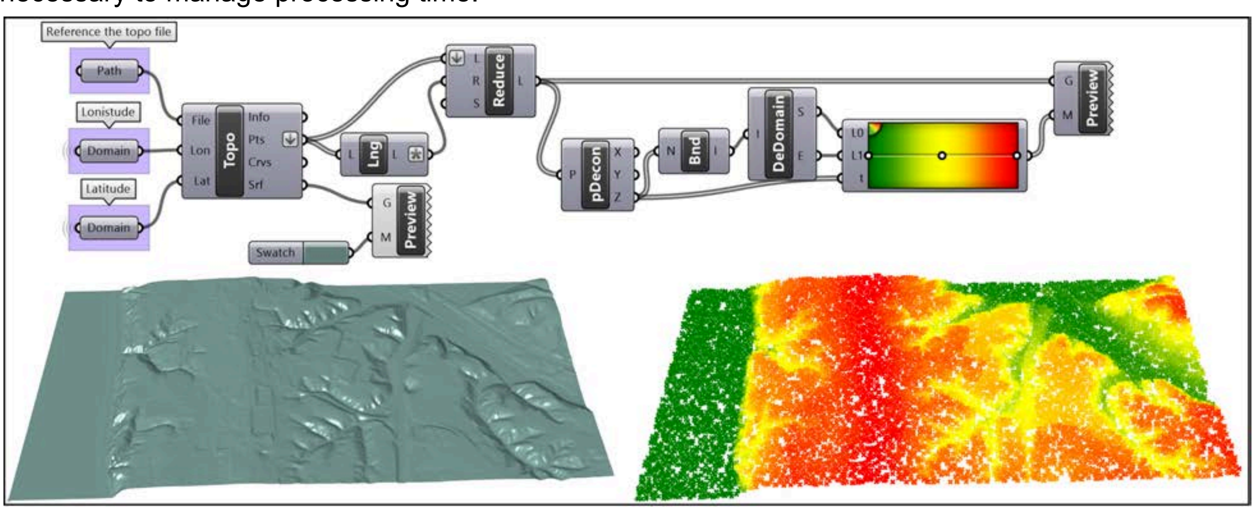
The first step is to get the ***Tiff*** or ***IMG*** file for the ***Torrey Pines*** region in ***San Diego*** from the USGS website. This offers the best resolution images. Zoom to the desired area and follow the steps in the image below. Note that you might need to create an account to login and get the desired maps.

1. Set to Box/Point located in the area above the map
2. Window select the area of interest on the map
3. Select "Elevation Products
4. Select ⅓ arc-second DEM
5. Click ***Find Products***

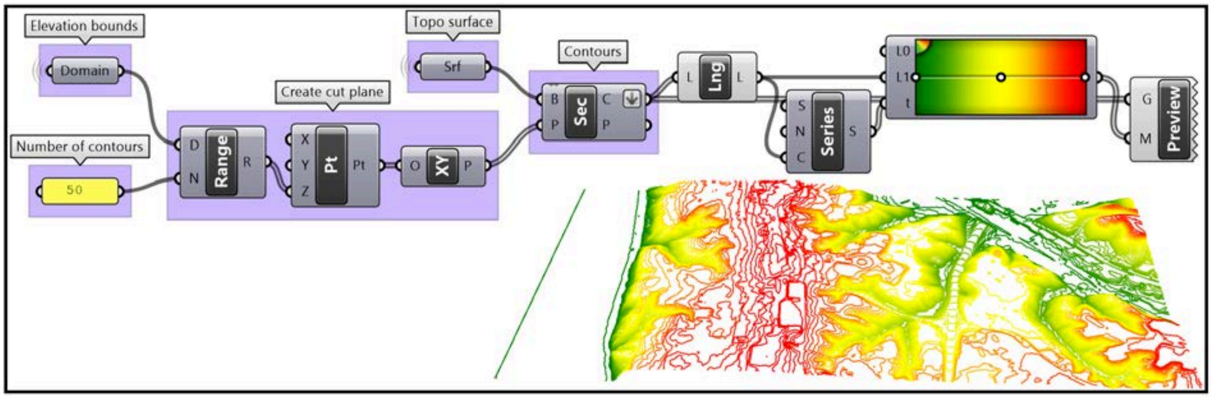


The ***Elk*** topology component outputs 3 data types: **Points, Curves** and a **surface**.

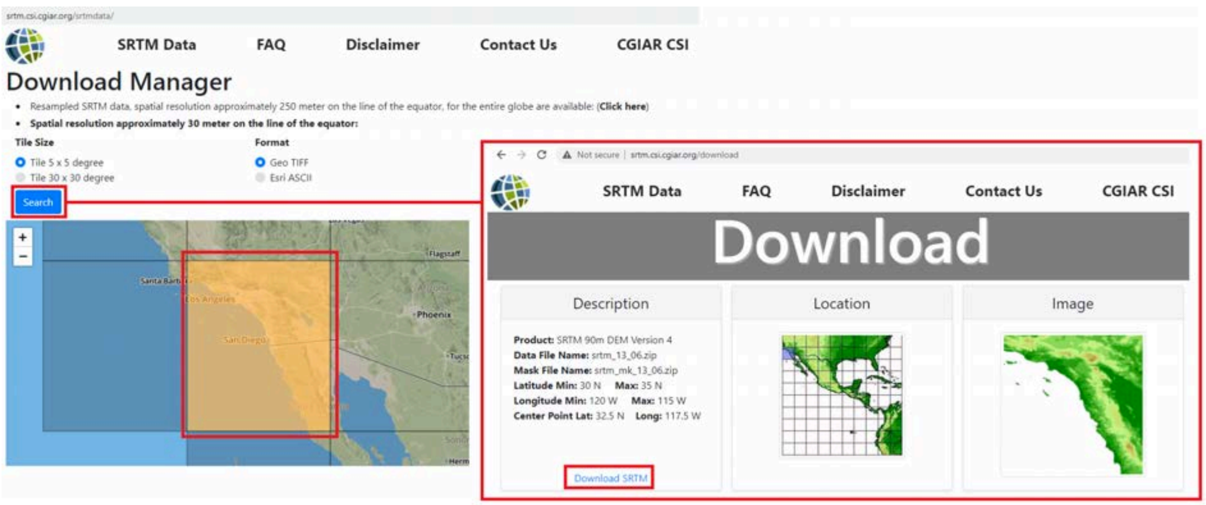
The surface output is the one that is useful to use.You can also color points by z elevation of points to get a quick overview of the topology but make sure to check the size of pointset and reduce if necessary to manage processing time.



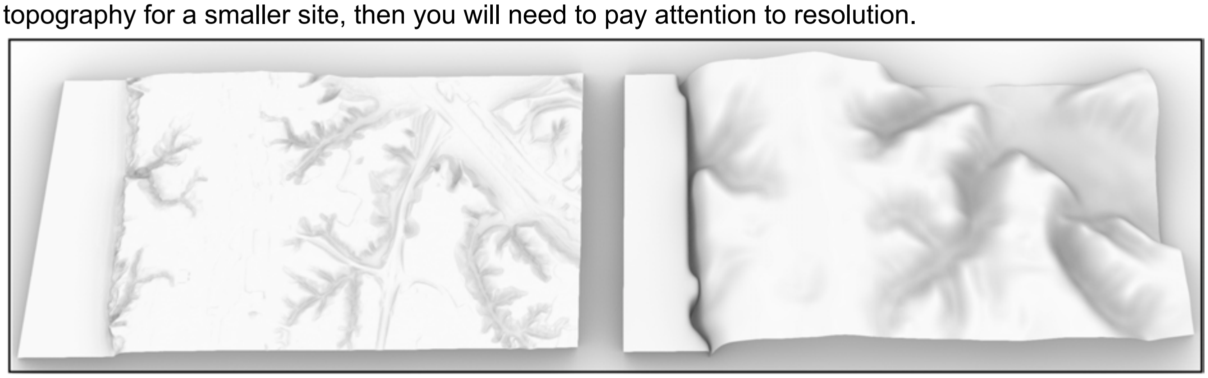
The curves do not represent contours and are generally not useful to use. You can however extract the topography from the surface and the minimum and maximum z elevation from points as in the following.



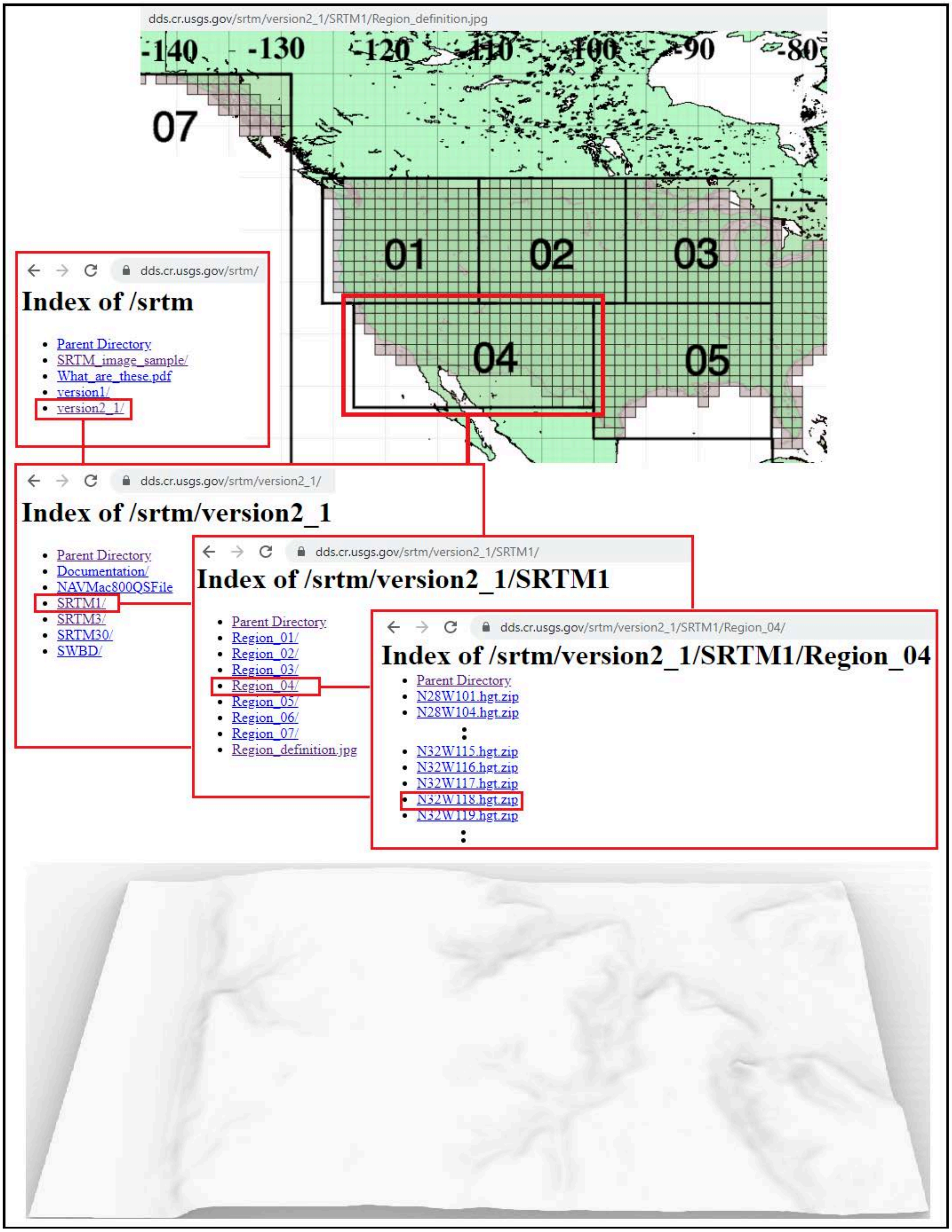
You can get the ***TIFF*** file from the ***SRTM*** site:



The images acquired from the ***NASA Shuttle Radar Topography Mission*** digital topographic have lower resolution than the one available from ***USGS***. This shows a comparison between the two. The left is the topo surface from the tiff file downloaded from ***USGS*** and the image on the right shows the result from the file downloaded from ***SRTM***. For some applications this might be sufficient, but if you need the topography for a smaller site, then you will need to pay attention to resolution.



Elk topography accepts ***HGT*** files.

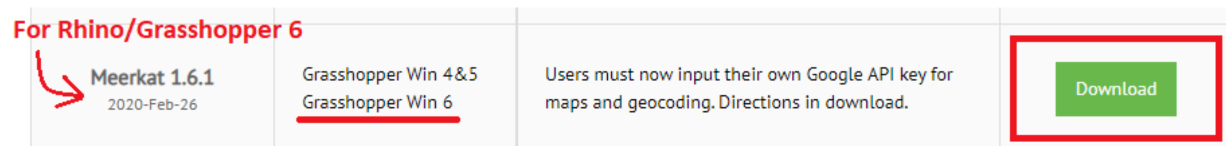


Meerkat for Grasshopper

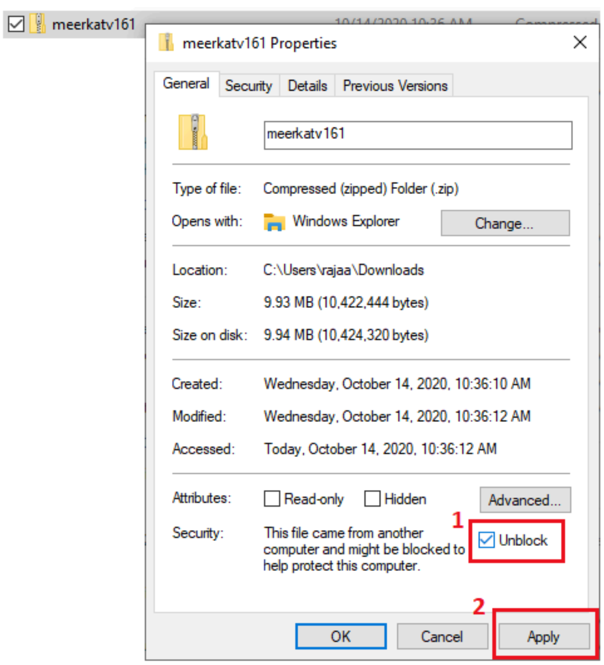
Meerkat is a free plugin for ***Grasshopper*** to process GIS data from shape files.
Meerkat is developed by ***Nathan Lowe*** and is currently under active development.

Download and setup

Download ***Meerkat*** zip file from ***Food4Rhino***:
https://www.food4rhino.com/app/meerkat-gis



Make sure to ***Unlock*** the downloaded zip file before extracting the content by right-mouse-click, then go to ***Properties***. In the dialog, check the ***Unlock*** checkbox, then click ***Apply***:



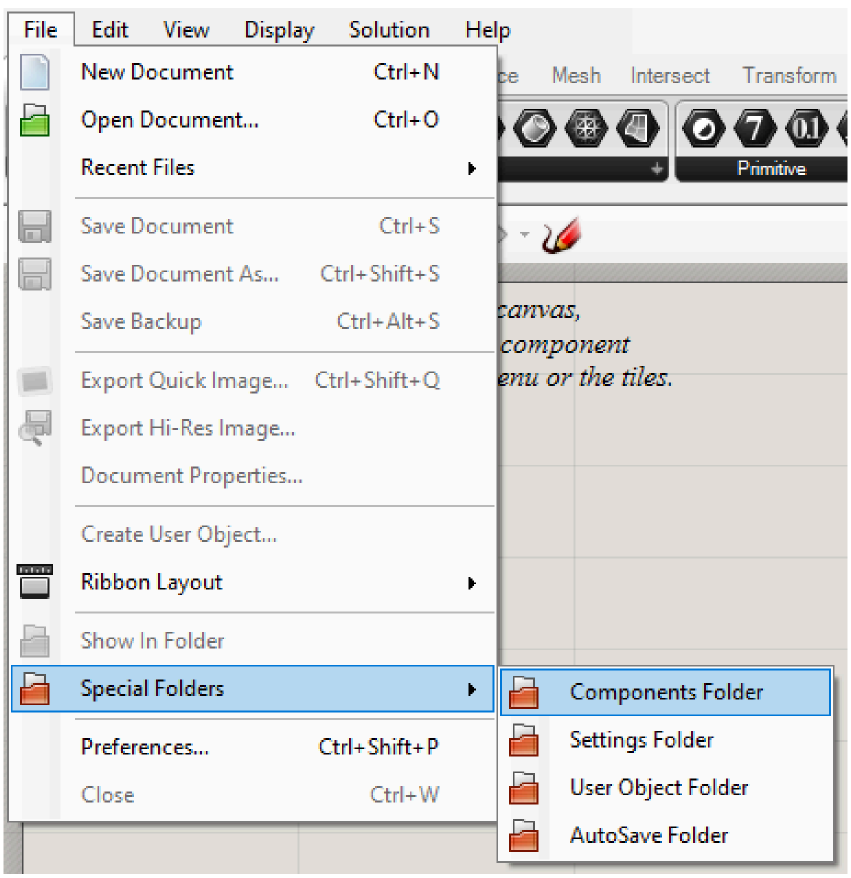
If you do not have one, create an account with ***Google Cloud Platform***. You can set up a trial (90 days) and you will be given $300 credit to start with. You will need to provide an address and a credit card information to set up the account. You will not be charged automatically after using your credit and you will need to manually activate the automatic payment of $200 per month for access.

Google Cloud Platform

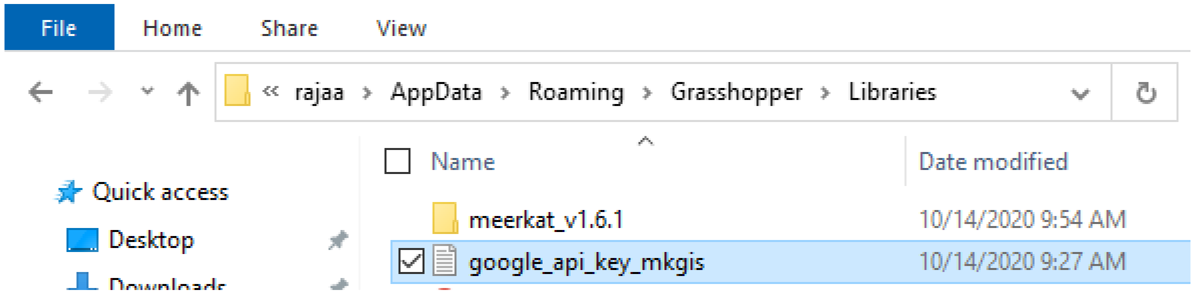
Locate your ***API key*** in your ***Google Cloud Platform*** . Make sure it is either not restricted to specific apps, or create an API key that has access to:

- ***Maps Javascript API***
  https://developers.google.com/maps/documentation/javascript/get-api-key
- ***Geocoding API***
  https://developers.google.com/maps/documentation/geocoding/get-api-key

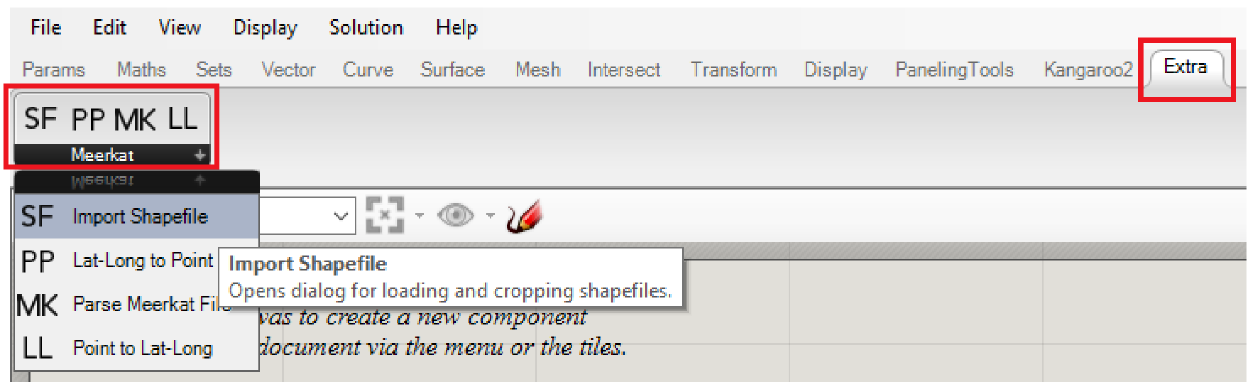
Copy your API key to the ***google_api_key_mkgis.txt*** file in the ***Meerkat*** folder (delete everything else in the file. You should only have the key there).
Move the file to the ***Grasshopper*** component libraries folder (you can open it through an open session in ***Grasshopper*** under ***File>Special Folders> Components Folder***.



Make sure you place ***google_api_key_mkgis.txt*** directly in the ***Libraries*** folder and not in a sub-folder, then move the extracted Meerkat folder and place it in the same Libraries folder.

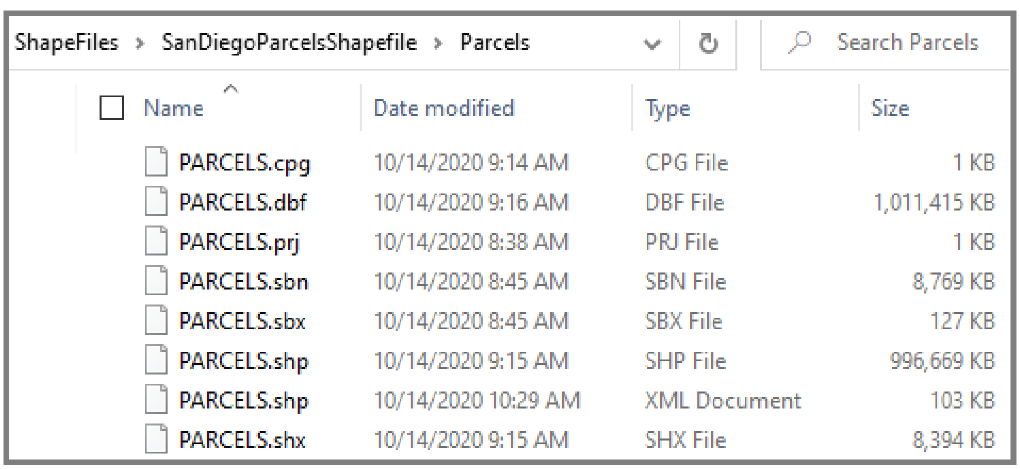


Open Grasshopper, and you will see a new ***Extra*** tab with ***Meerkat*** components inside:

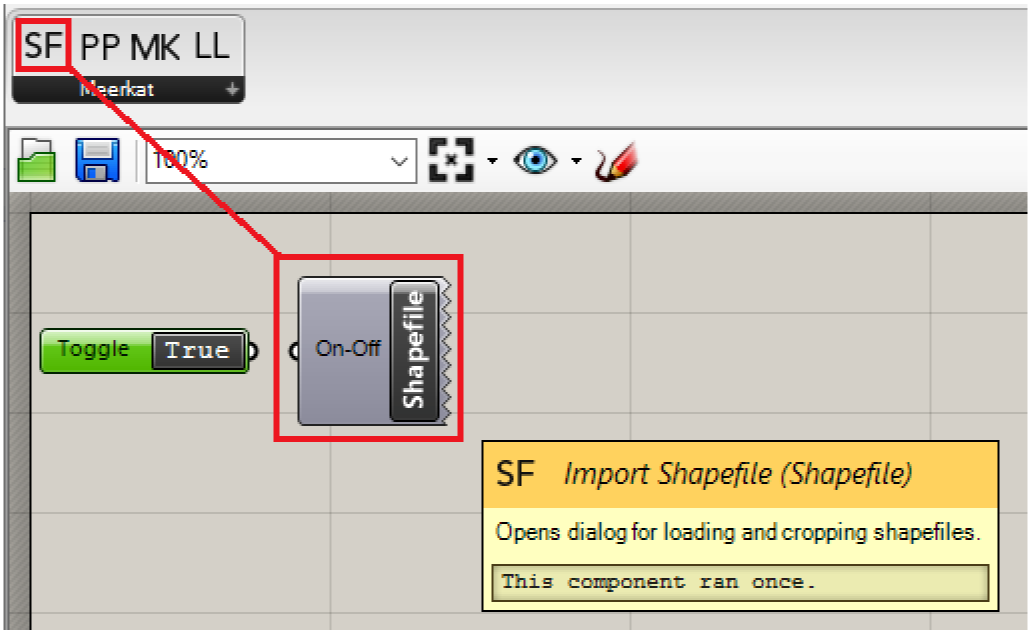


Meerkat Workflow

The first step is to download the shapefiles of the areas you are interested in. Different cities usually manage their shape files and each has a different location to access. For this example,we will get the ***Torrey Pines*** area in ***San Diego*** through ***sandag.org > Resources > Maps and GIS*** to get the parcels shapefile. Make sure to unlock the ***parcels.zip*** file before extracting data as explained above. The folder of the parcels typically include many files but only one ***.shp***.

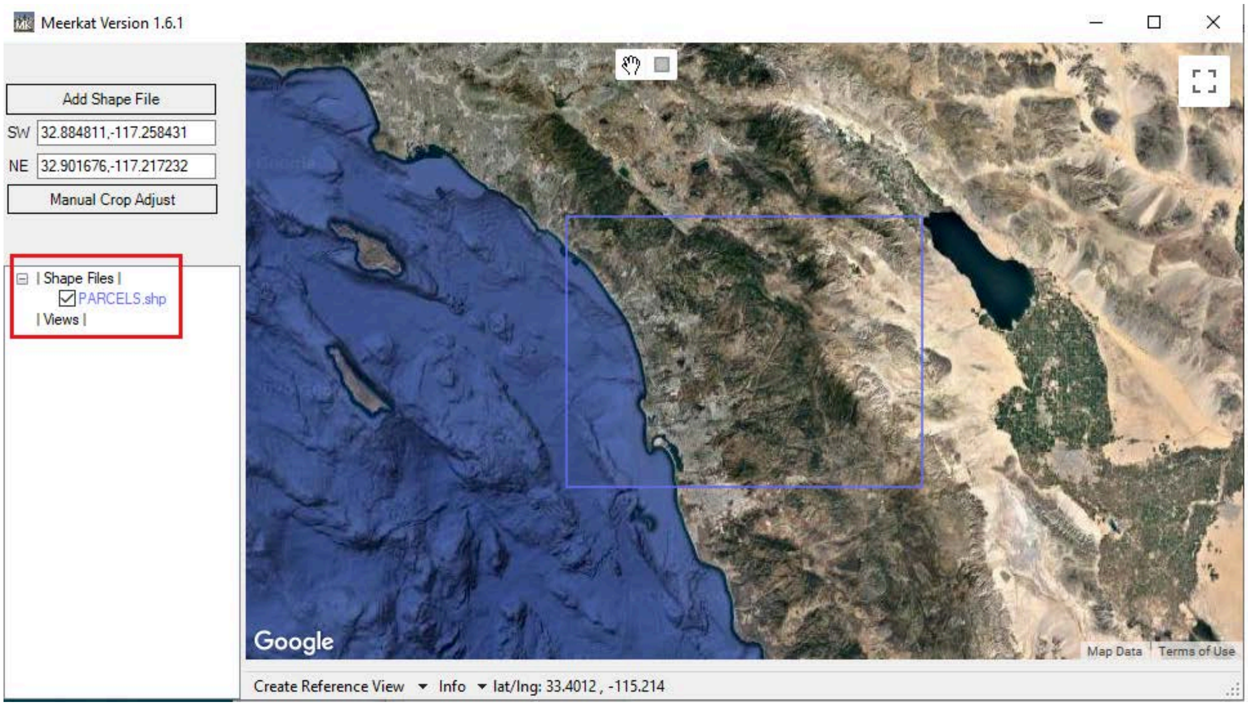


Using ***Meerkat*** plugin for ***Grasshopper***, you can use the import shapefile component to load your shapefile. Connect a GH ***Toggle*** to the ***Shapefile*** component



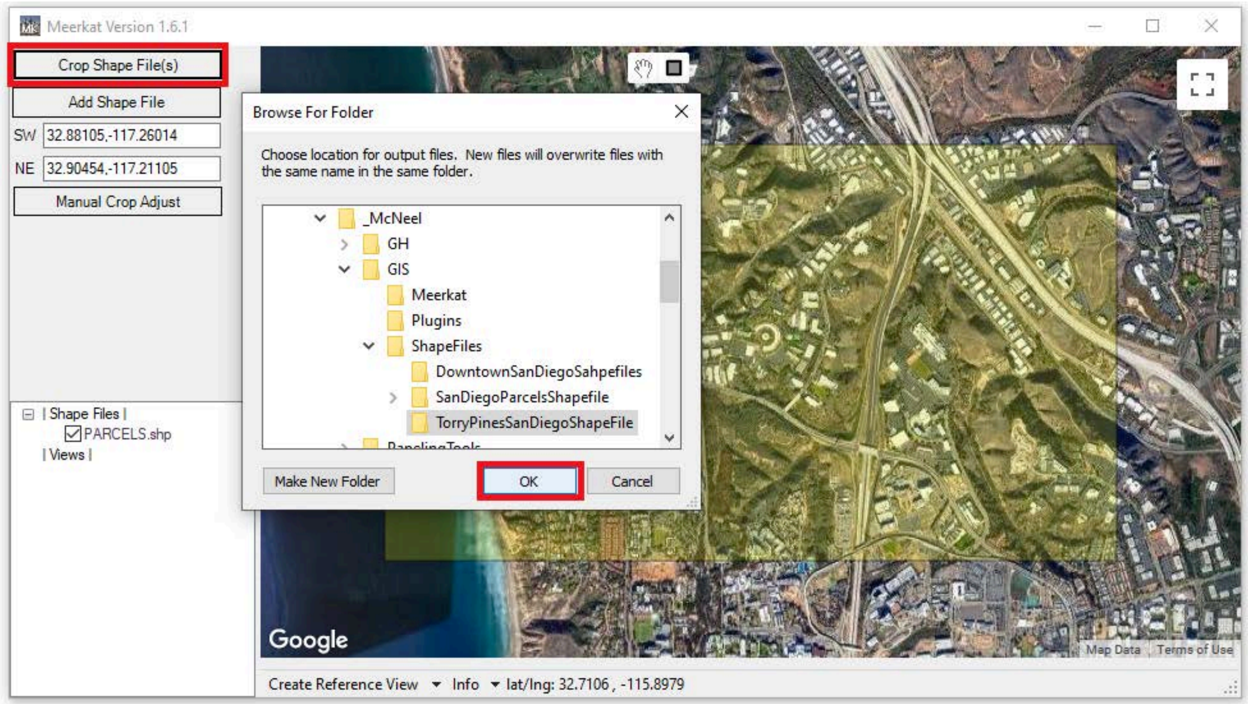
Once you connect the ***True*** toggle to ***Shapefile*** component, the following window opens. Clock on ***Add Shape File*** and navigate to your shapefile.

Once you add the shapefile downloaded earlier, you will see it added to the left menu, and when you check the box next to it, the map updates:

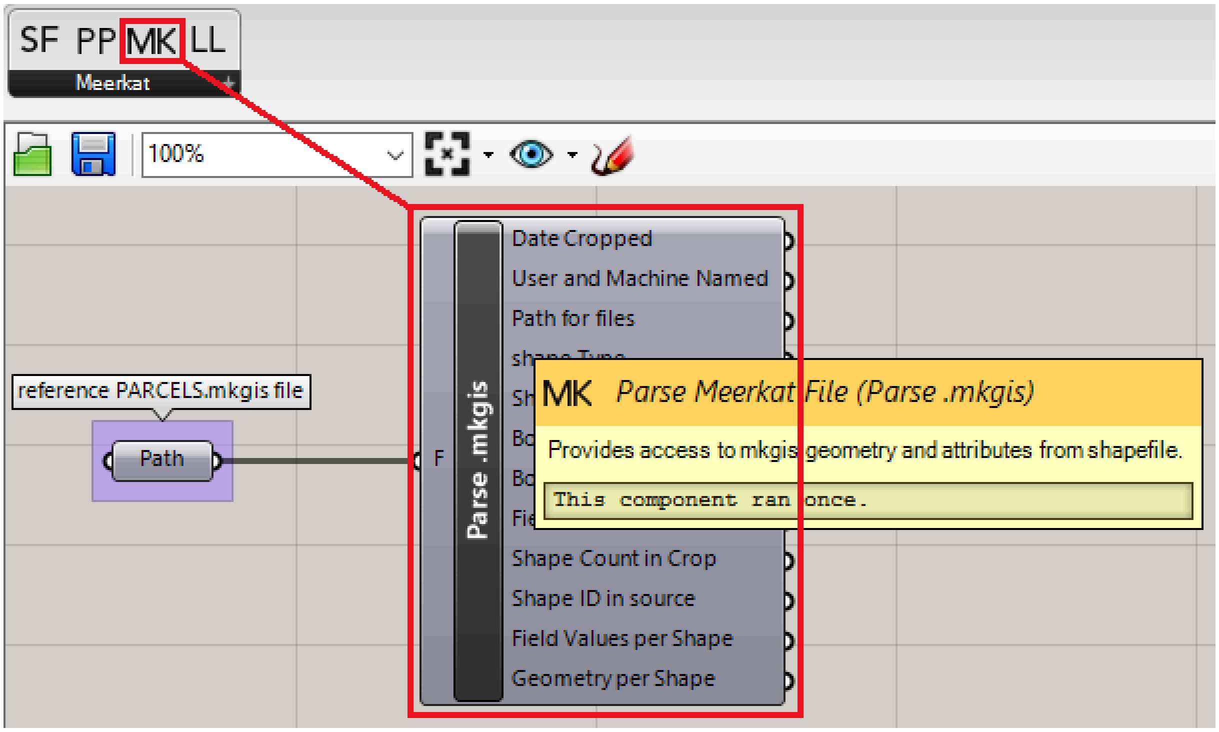


Zoom to the downtown area, then draw a boundary using the little square icon at the top of the map area. Notice that the left menu updates with the corresponding bounding coordinates. Click on the ***Crop Shape File(s)*** to select the marked area.

Create a folder for the cropped area to place your cropped shapefile. This will create a ***PARCELS.mkgis*** file in the selected location that you can then reference and use to extract data in ***Meerkat***.



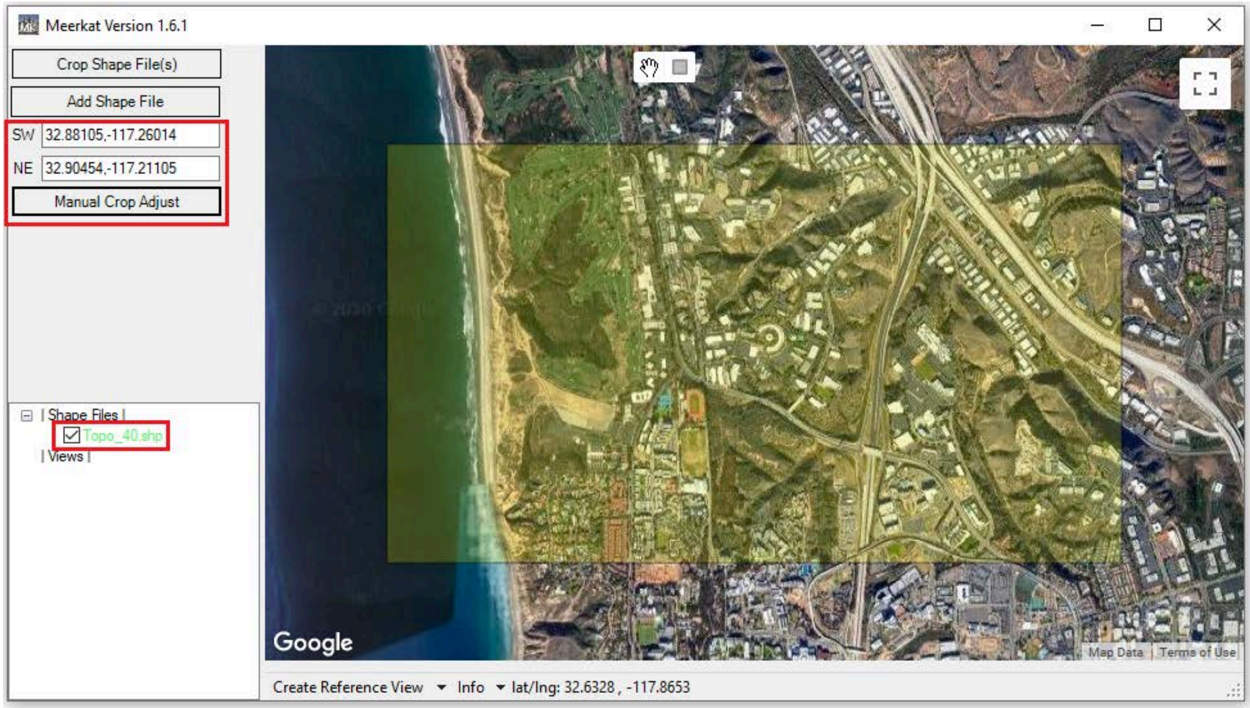
Now that the Meerkat shapefile is created, you can reference it to extract data. Use the ***Path*** component in ***Grasshopper*** to set the path and connect to the ***Parse .mkgis*** component



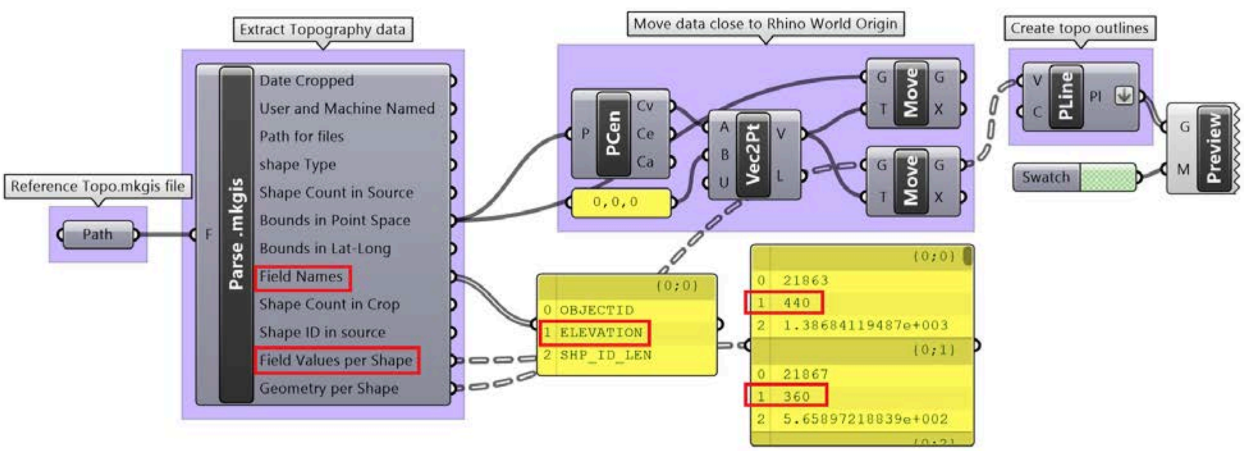
The information of the shapefile includes the boundaries of the parcels as an output from the ***Geometry per Shape*** in the ***Parse .mkgis*** component. In the Rhino viewport you can locate the points if you zoom to the right area. Most of the time, the data is placed away from the origin and this makes it hard to find and process in most cases. It is recommended to move all the points close to the origin before processing further.

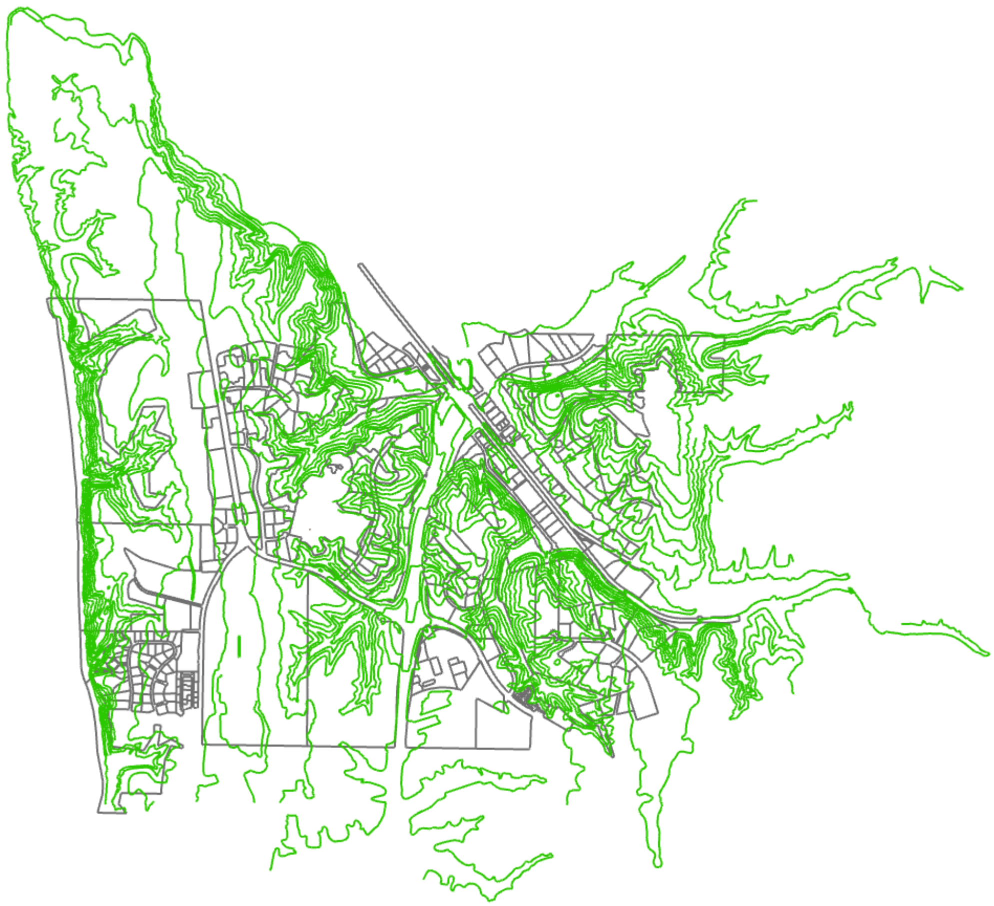
The outlines of the parcels or sites are defined as point sets in the shapefile, therefore you can use the ***Polyline*** component to create the outlines of the buildings and terrain.

When adding and crop additional shape files that have other information such as topography, you need to make sure to apply the same boundary. You can create an approximate boundary then manually enter the exact location used earlier then use ***Manual Crop Adjust*** to create identical boundary, then click ***Crop Shape File(s)***. Alternatively add all shape files for the project and crop together.

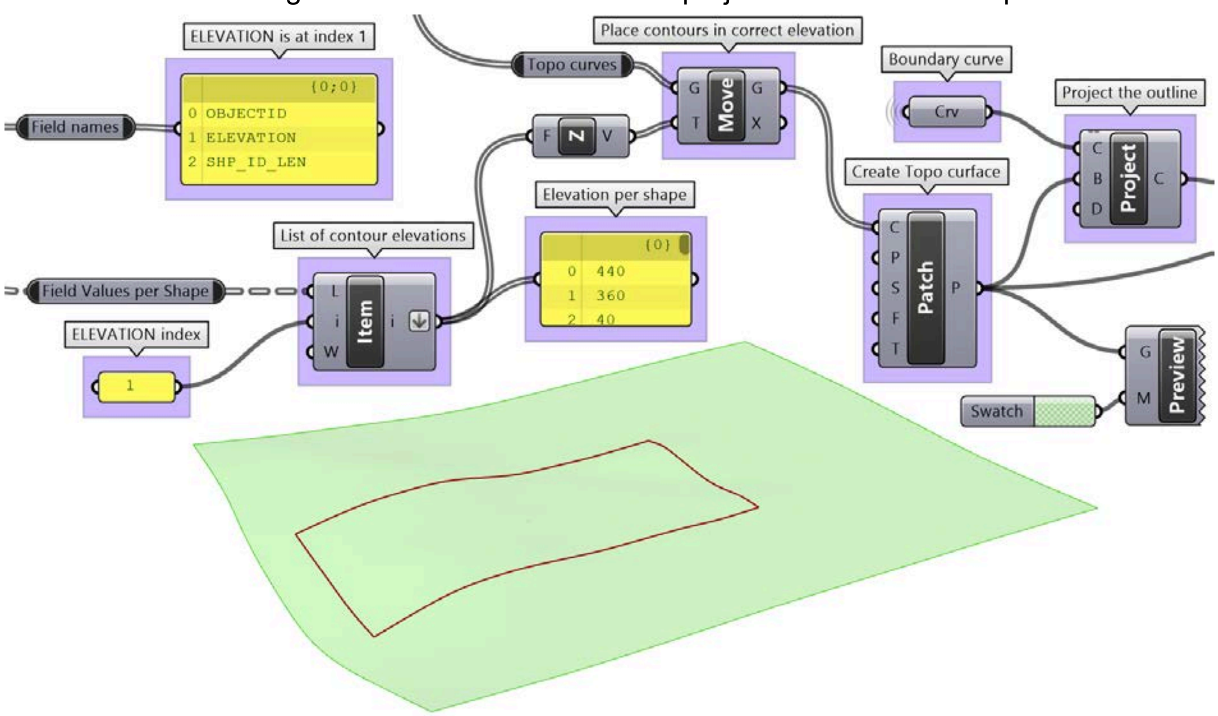


Once the topography is added to the Grasshopper file, you can move to origin to overlap properly with the parcels. Also notice that topography contours have elevation data. You can find the index of the elevation by examining the ***Field Names*** then extract the actual values from the ***Field Values per Shape***.

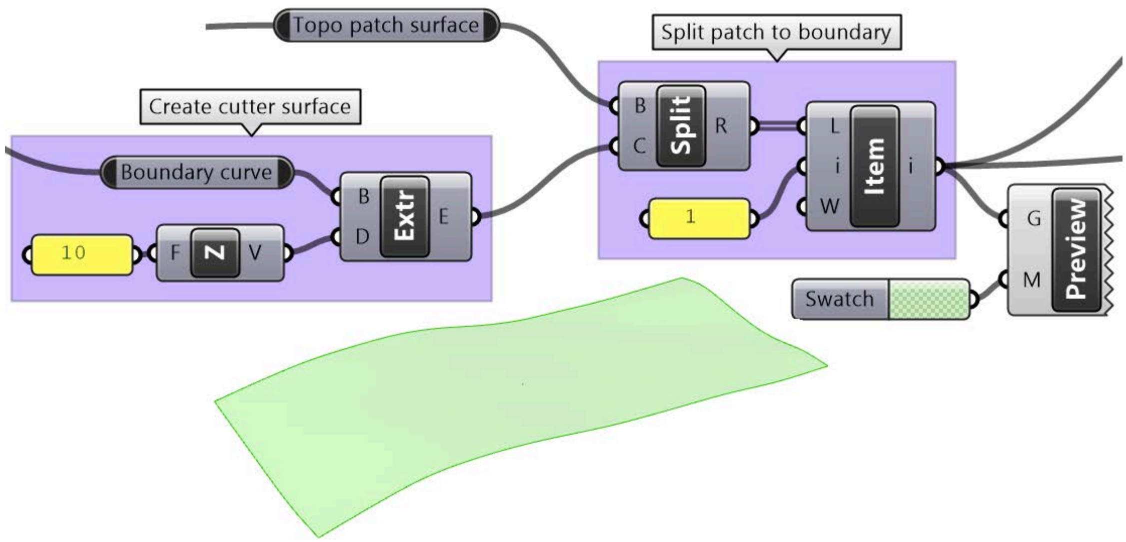




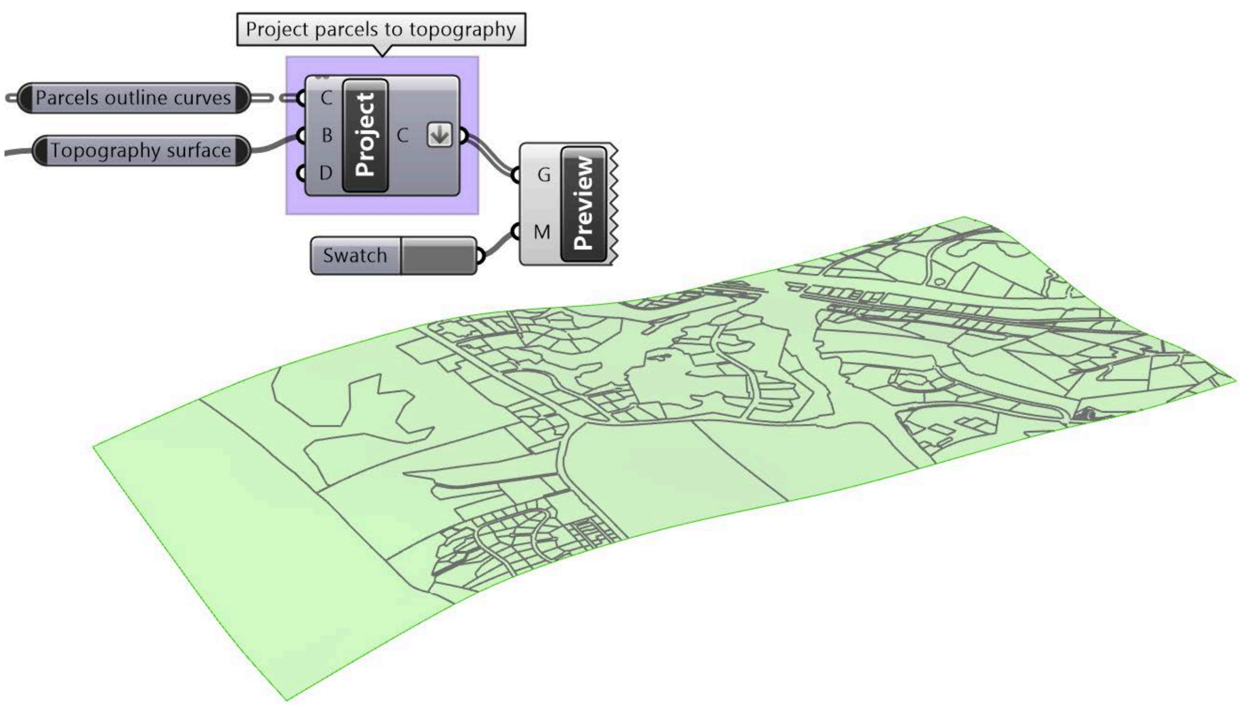
Use the elevation data per contour to place in the proper vertical location, then use ***Patch*** to create a surface through the contours. You can then project the outline to the patch.



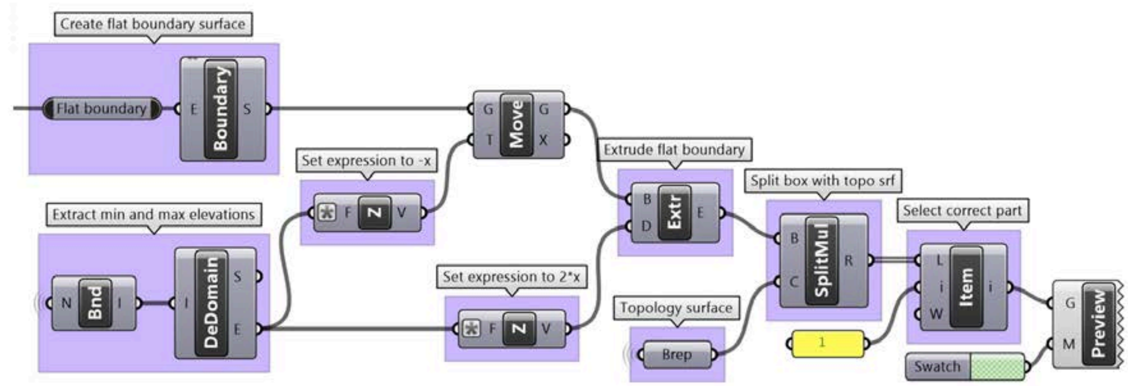
Extrude the outline and then crop the patch.

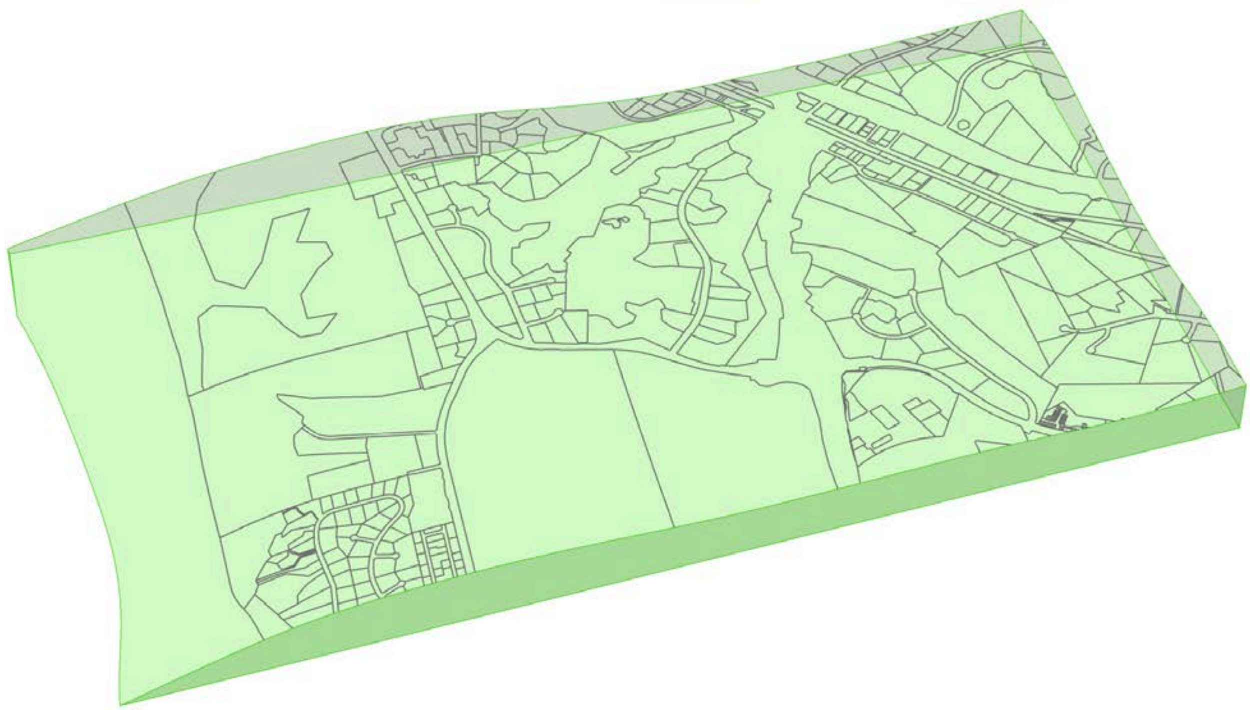


Next, project the parcels outlines onto the topography surface.



The next step is to generate the solid for the topography.





If you acquire the shapefile for the surrounding buildings, then you can also create the .mkgis and add to the model the same way as the parcels and topography, You can then create the 3D geometry of the surrounding buildings by extruding their footprint using the elevation value.