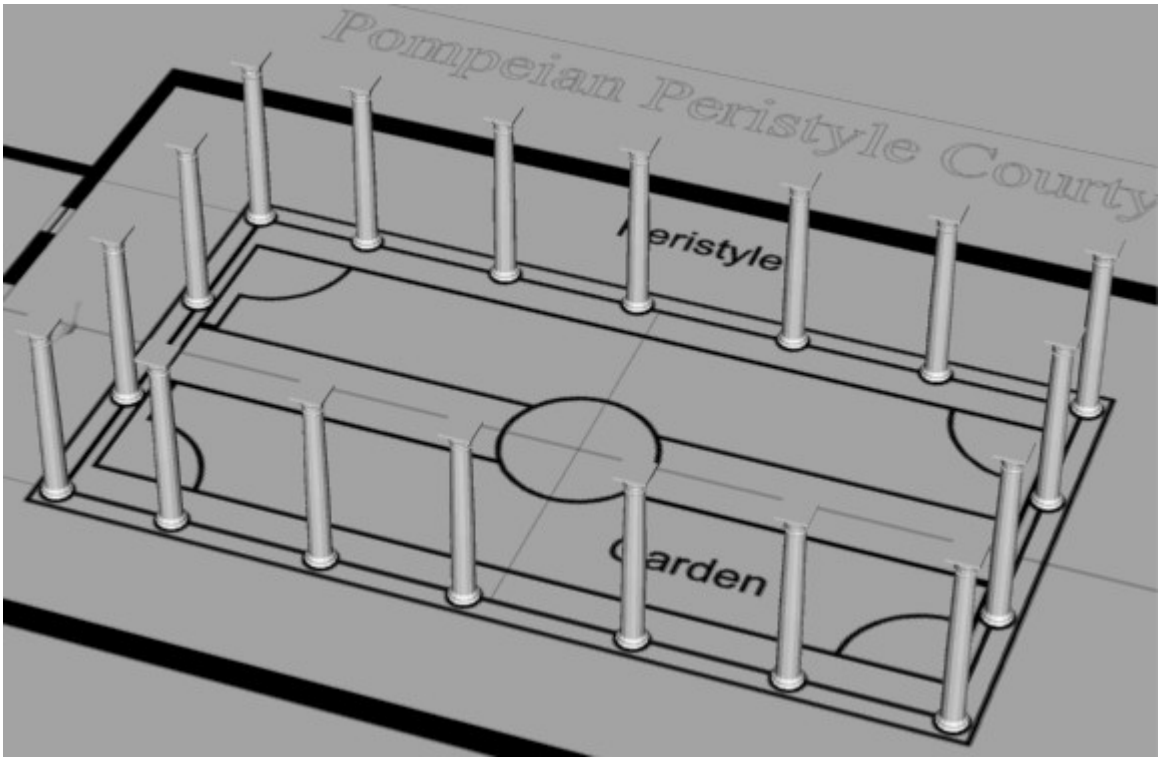
Command Scripting In Rhinoceros
By Roland Montijo

By the end of this section, you will learn how to automate Rhinoceros, to create complex models and scenes (including the one shown above) via your own Rhinoceros command scripts. Command scripting in Rhinoceros takes some patience, time, and understanding.

Introduction to Commandscripting in Rhinoceros



One of the most powerful features in Rhinoceros is command scripting. Rhinoceros features command scripting capabilities that let you automate modeling tasks and create new commands and icons to perform customized functions. Commandscripts can even build a whole model for you, based on different kinds of imported data or scripted procedures. You don't have to be a programmer or expert user to write command scripts in Rhinoceros.

A Commandscript or Macro is a series of Rhinoceros commands listed together in a text file so that they can be executed automatically in succession. This chapter will show you the different ways to run a Commandscript, the proper way to format Commandscripts, and get you started creating your own scripts.

If you never created commandscripts before or are new to Rhinoceros please go through Chapter1 in a linear fashion. I suggest this because each mini tutorial builds upon the one precedeing it. The tutorials are arranged in a step by step learning process that you should not skip if you want to gain a thorough understanding of the basics.

There are three different ways to run a commandscript in Rhinoceros. By working through the three examples in this chapter, you will gain a good understanding of their differences.

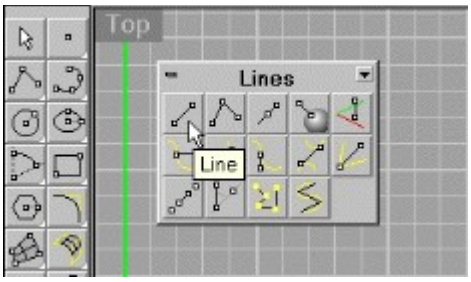
- The first way tp run a commandscript is by assigning correctly formatted text to a mouse button or hotkey. For commands that you use all the time, running Commandscripts from Icons or hotkeys can be ideal. However, you might not want a huge, rarely used macro assigned to a mouse button of an icon, or taking away precious hotkey space.
- The second way is by typing **commandpaste** or clicking the commandpaste Icon in the Tools toolbox. For Commandpaste to work you need to have correctly formatted text copied to the Windows clipboard first. Command paste is mainly used to test commandscripts. It can also be used to execute quick one-time macros you create on the fly or you don't want to save.

- The third is the Rhinoceros command **Runcommandscript**. Runcommandscript is used to execute commandscripts from a saved text file. You first have to write the macro in a text editor. You then save it as a .txt file and then execute it in Rhinoceros. These commandscripts should be fully tested and free from errors. You should store your commandscripts in a separate distinct folder. You can share the fully tested ones with others and combine them to make other commandscripts.

Example 1- Running a commandscript from an icon.

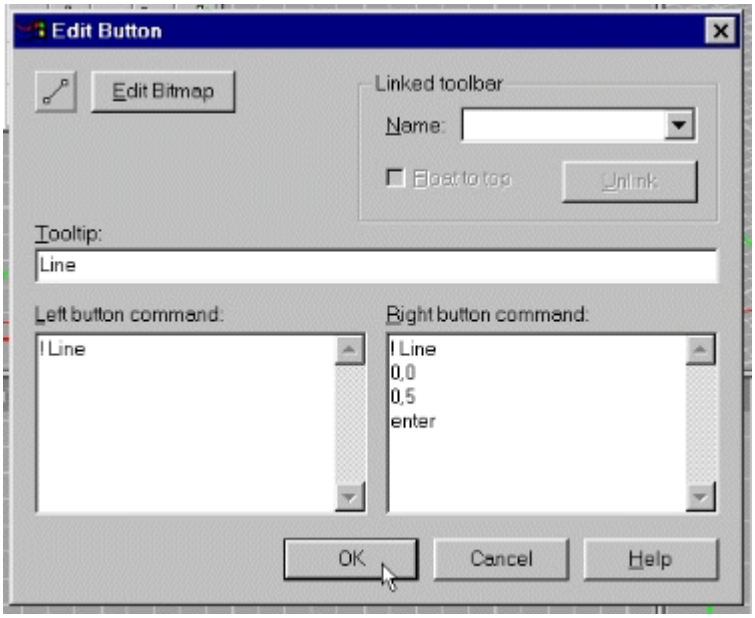
Here is a simple but effective way to create a commandscript using the line command. This simple macro will draw a line 5 units long when you right mouse click the line icon...

1. Hold the Shift key down and right mouse click over the Line icon to edit the icons' contents.

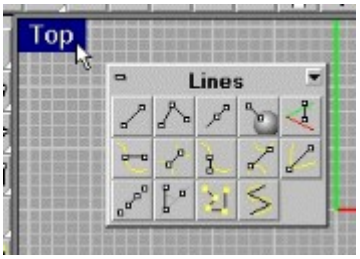


2. In the right button command box enter the text exactly as shown in Figure 2, Click OK. You can copy and paste the text below.

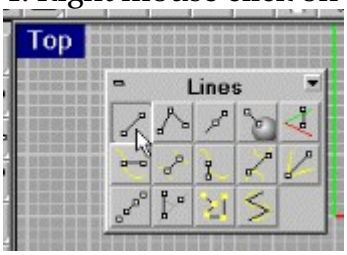
```
! Line
0,0
0,5
enter
```



3. Click on the Top viewport title to make it active. Commandscripts are viewport sensitive. This means that Commandscripts might work differently depending on which viewport is active before processing the commandscript.



4. Right mouse click on the line icon in the Lines toolbox. Figure 4

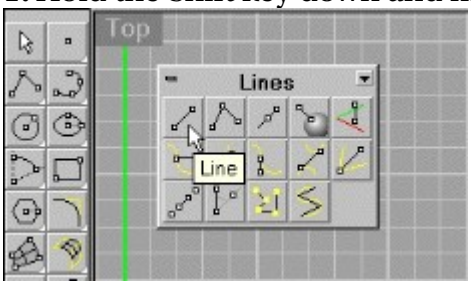


5. Select all and delete the contents leaving the drawing free of objects for Example 2.

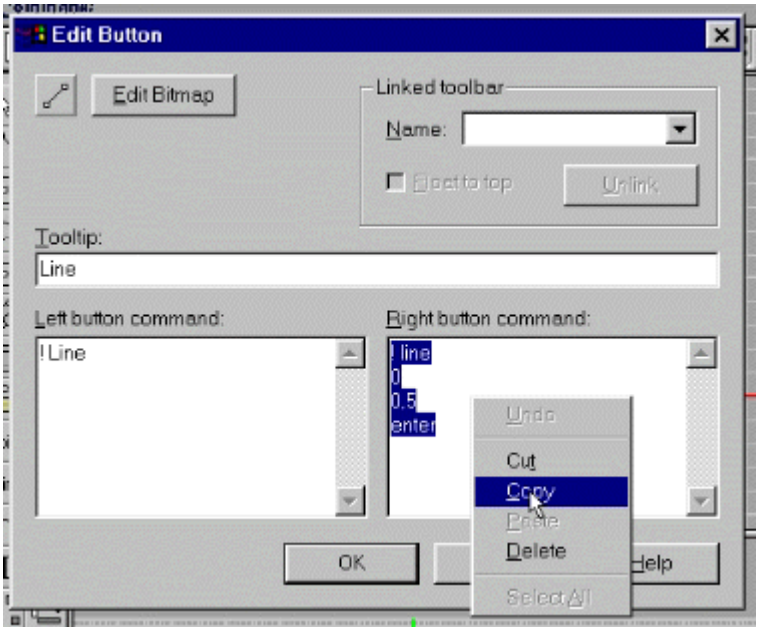
Example 2- Running a commandscript using Commandpaste

Command paste runs a commandscript by pasting text contents from the Windows clipboard. As an example...

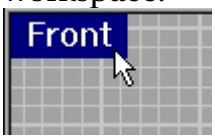
1. Hold the Shift key down and mouse click over the Line icon.



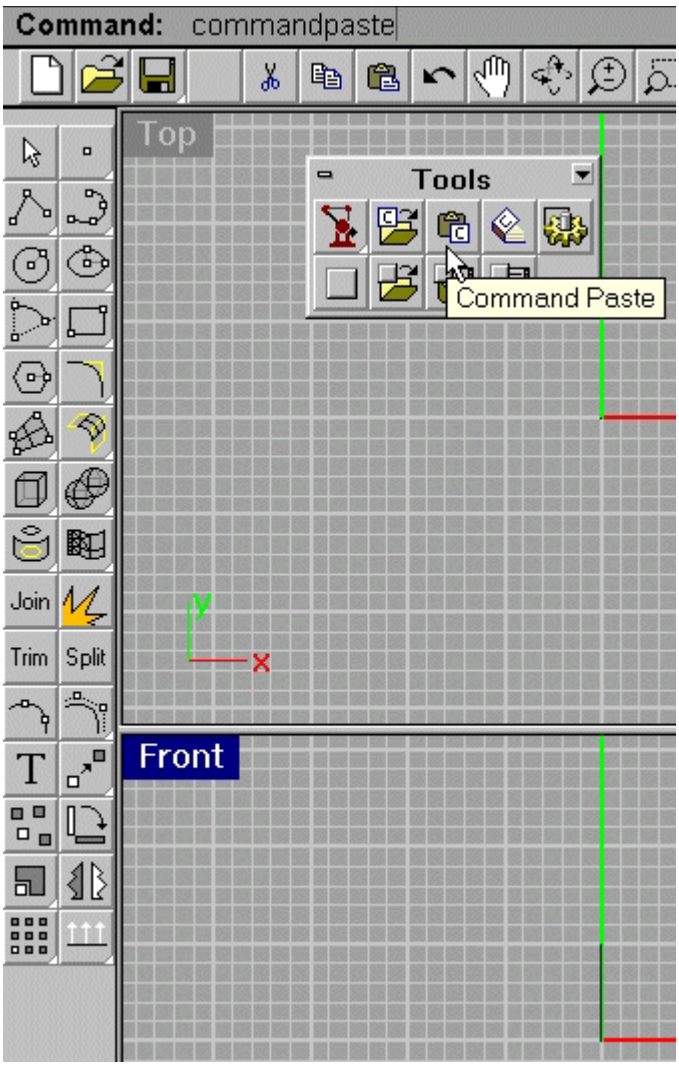
2. Highlight and then Copy the contents of the right button command box to the Windows clipboard.



3. Make sure the front viewport is active by mouse clicking on its' viewport title. The viewport title should then be highlighted in blue if you are using the Rhino default workspace.



4. At the Rhinocereos command prompt type "commandpaste" and hit the enter key or from the Tools toolbar click the commandpaste Icon. Notice this command also performs the macro but in the front viewport which gives it a different orientation.
Important: For command paste to work you need to have some kind of script or text copied to the Windows clipboard first. Figure 8

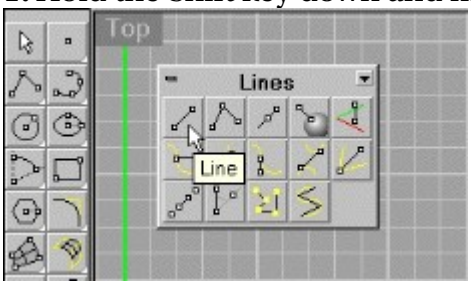


5. Select all and delete the contents leaving the drawing free of objects for Example 3.

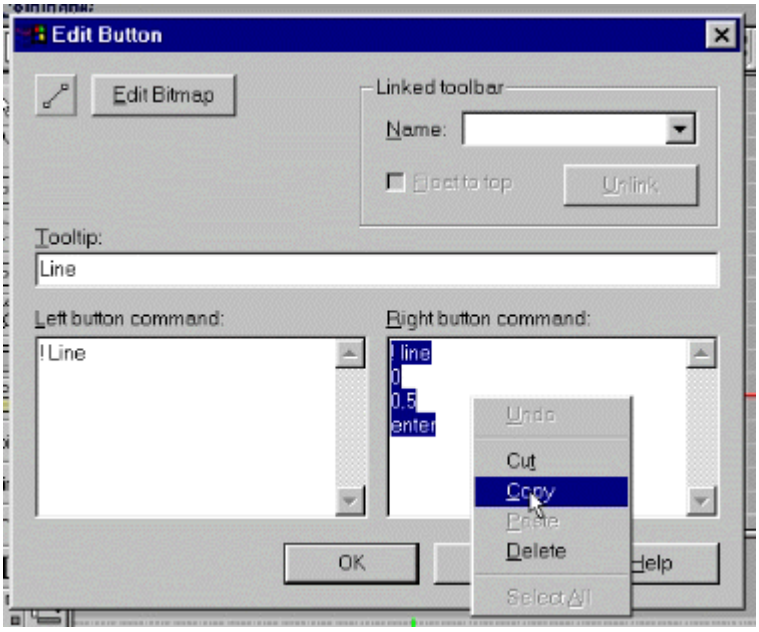
Example 3- Running a commandscript using RuncommandScript

Rhinoceros does not include any command script files, so you have to provide your own.As an example to create one quickly.

1. Hold the Shift key down and mouse click over the Line icon again.

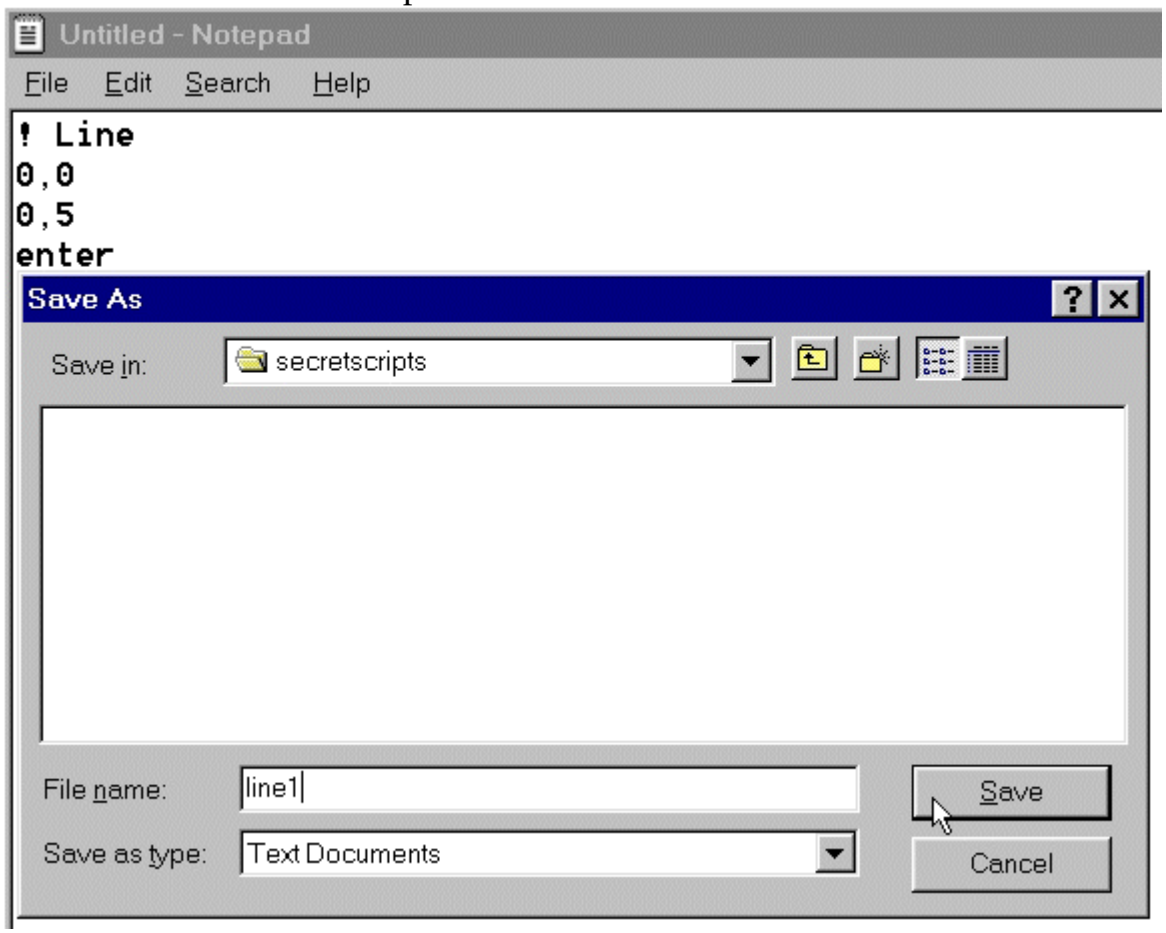


2. Copy the contents of the right mouse button command box to the Windows clipboard, by right mouse clicking, selecting all, then Copy.



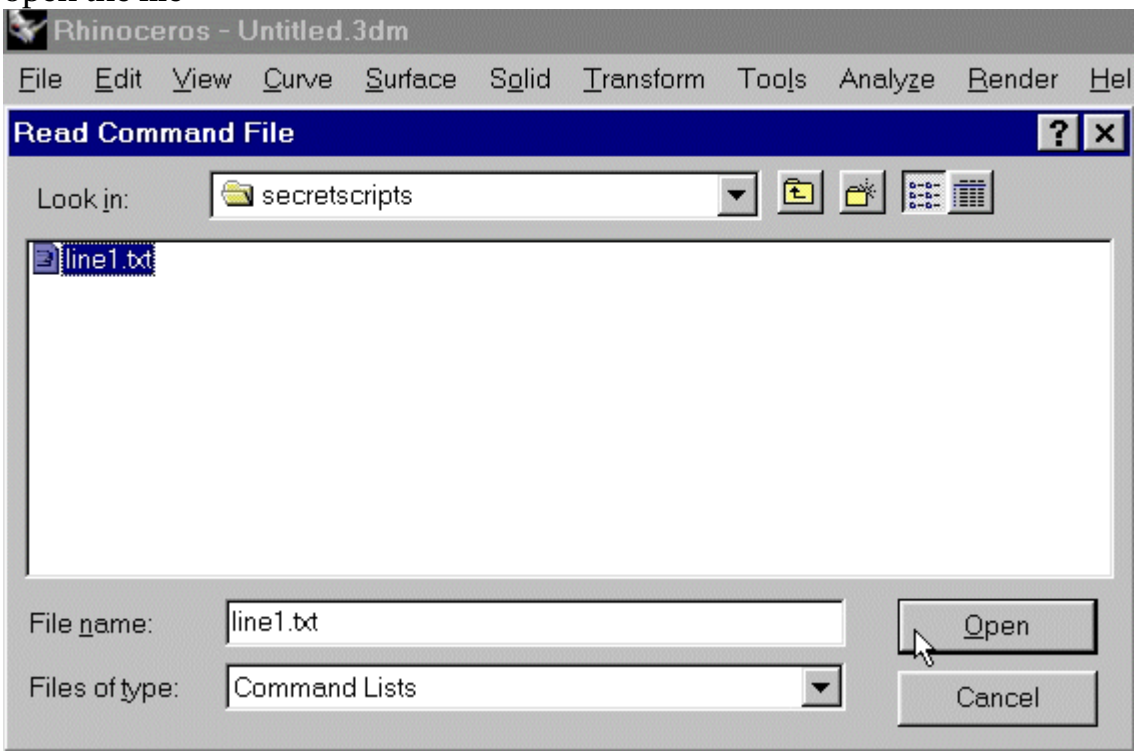
3. Open the Windows Notepad application from Programs Accessories.

4. Paste the contents of the Windows clipboard into a file.
Save the file as "c:secrtedcripts\line1.txt"

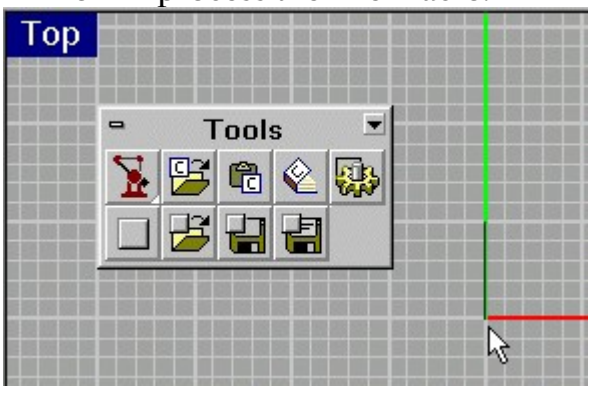


6. In Rhino at the Command prompt type " ReadCommandFile" or click the read command icon.

7. The Rhino dialog box will prompt you to find your file.
Go to "c:\test1.txt"
open the file



Rhino will process the line macro.



Commandscripting Syntax

Think of each command in Rhinoceros as a small commandscript. Every command in Rhinoceros has an official "command name". The command name does not necessarily match the command's label that you see in a menu or an icon's tooltip. Only official command names can be used within a commandscript or when typing commands from the keyboard into Rhinoceros. For example, the command labeled "Change layer" in a menu must be typed Changelayer, without a space in a commandscript. Changelayer is the official command name for this function.

Avoid errors by always using the official command name when you type a command. To guarantee accuracy, you can copy the official command names from the help files, and paste them directly into your scripts. You can also type the word "commands" at the command prompt and Rhinoceros will send all its command names to the command history or to the Windows clipboard. You can then view the entire list of Rhinoceros command names in the command history by pressing the F2 hotkey on your keyboard or you can paste it into a text file.

1. Type the commands in your script just as you would at the command prompt in Rhinoceros.
2. A space between characters or a new line in your script mimics the user pressing the Enter key at the command prompt. The spaces between the entries are the same places you would press Enter when typing the command by hand.
3. The word enter, ! enter, or ! can end a script or a script sequence.
4. An exclamation point ! and a space after it (remember to put a space after the ! symbol) are usually used to begin a script . The exclamation point ! symbol cancels any previous command. The exclamation point can also be used at the end of the script it is similar to ending a script with the enter key.
5. Three-dimensional spatial coordinates that are used to locate a point in space in Rhinoceros must not have spaces in between the numbers and commas.
   A point located at cplane 12units X axis,10 units Y axis, and 8 units Z axis in Rhinoceros would be written in a script:
   12,10,8 not 12, 10, 8
6. Commandscripts are viewport sensitive this means that scripts work differently depending on which viewport is active before you process the commandscript.
7. Commandscripts are generally not written as one line of text with spaces only, although they can be.
   Here is an example:
   ! Polyline 0 10,0,0 10<60 0 enter
   This will work but it makes it harder for the user to read. Notice that it is very hard to sort out the spatial coordinates.
   Commandscripts should generally be written with line breaks between the commands and three-dimensional spatial coordinates to make it easier to script, debug and understand at a glance. Below in the left hand column is the same script with line breaks in between commands and three-dimensional spatial coordinate points. Notice how much easier it is to read.

! Polyline	This command tells Rhinoceros to create a polyline

w0	Coordinate point to begin creating polyline
10,0,0	Coordinate point for polyline
10<60	Polar Coordinate point for polyline
w0	Ending XYZ coordinate point for polyline
enter	Tells Rhinoceros to end the script

Protocol for Running a Comandscript

1. Make sure the viewport that the script is to be performed in is active
2. Drag the Toolbars/icons that execute a commandscript to the top of the workspace off of the viewport areas.
3. After clicking an icon that executes a commandscript do not touch the mouse.
4. Do not move the mouse cursor while a commandsrcript is active
5. Always wait for a commandscript to end completely before you touch the mouse or resume modeling. It is easy to make sure the script is complete because Rhinoceros will display a blinking cursor on the command line when a script or modeling operation is done. Always make sure the blinking cursor appears in the command line before you touch or move the mouse, resume modeling or scripting.
6. Sometimes because of an improperly placed "enter" command in a script the script might loop forever. If this happens hit crt-alt-delete to exit Rhinoceros.

Customizing Rhinoceros with Commandscripts

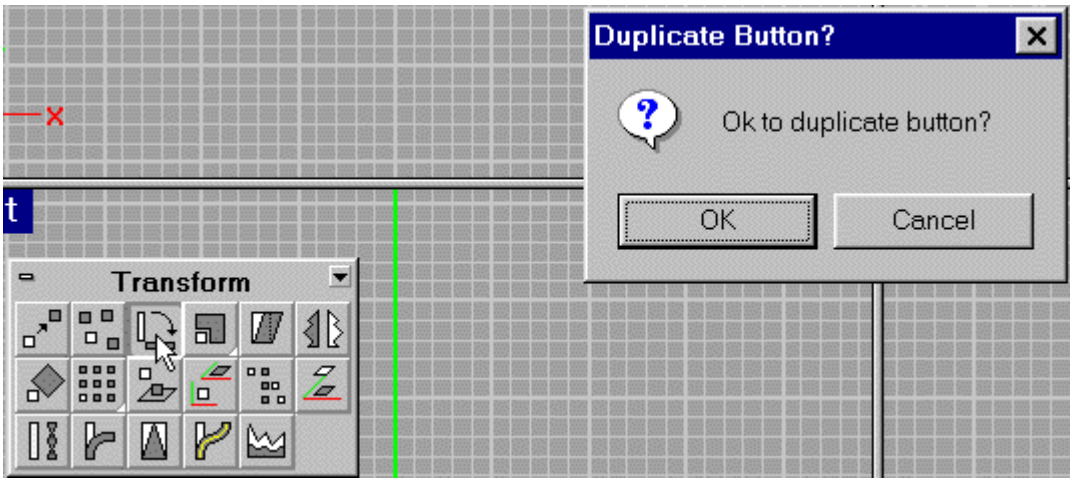
Writing simple Commandscripts assigned to icons will give you a basic understanding of how scripting works because you will become familiar with command names, what they do, and how they interact with each other. This will give you practice writing commandscripts before you go on to tackle larger and more complex projects.

Creating commandscripts on icons lets you customize Rhinoceros with your modeling, editing and display preferences. So you can change your modeling tools and add options to them in Rhinoceros to get your job done quickly and efficiently.

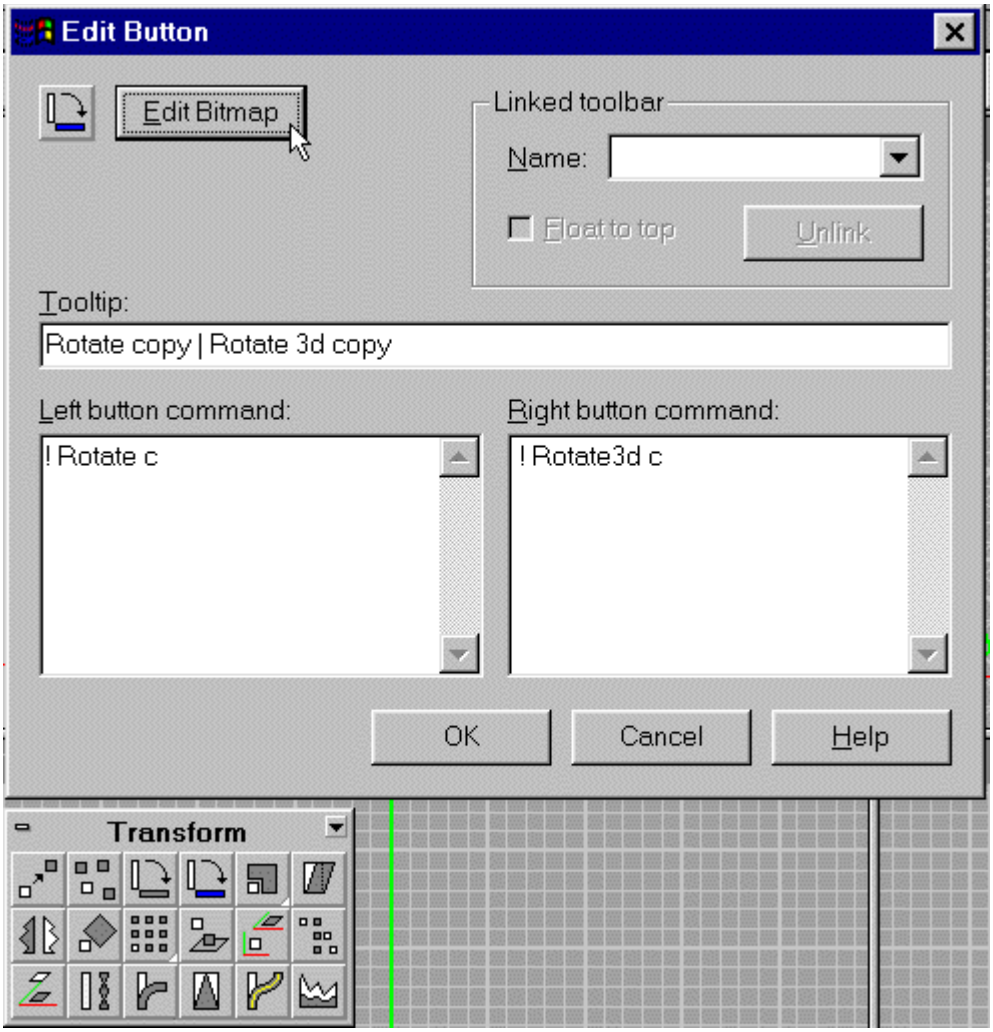
Customizing command "options" that appear at the Rhinoceros command prompt in the middle of modeling.
Quite a few commands that you use everyday in Rhinoceros have extra options to them that you have to input from the keyboard as you model. These are often one letter commands like c, y or n. For instance if you rotate an object, Rhinoceros also gives you the option to copy that object and rotate the copied object instead of the original. After you click the rotate icon you would do this by typing c at the keyboard. This can seem like it is easy enough. But if you need to rotate copies all the time and do it repeatedly this creates extra work. Usually you forget to type in the c option after many times of doing this and end up doing a regular rotate that you must cancel or undo. Since Rhinoceros has many commands with these extra options you can customize or make icons already with your pre-selected options to avoid fatigue and mistakes.
Remember *You need to select the object you wish to rotate, scale etc. first then click the icon for these commandscripts to work.*

Rotate copy Icons Tutorial

1. In Rhinoceros open up the transform toolbar and drag it into the Rhinoceros workspace.
2. Hold the control key down on your keyboard and left mouse click over the Rotate Icon.
3. Ok to duplicate box pops up click Ok.

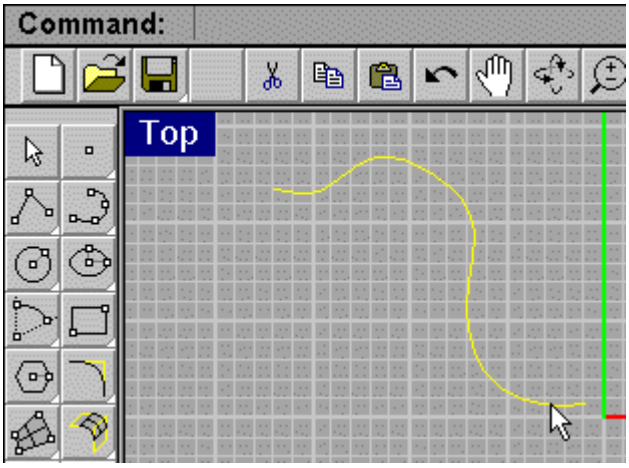


4. Shift key right mouse click over one of the duplicated Rotate icons
5. In the Left button command enter...
! Rotate c
In the Right button command enter...
! Rotate3d c
Change the Tooltip to read ...
Rotate copy | Rotate 3d copy, click Ok.
6. Change the color of the icon if you wish to make the new custom icon easier to identify.

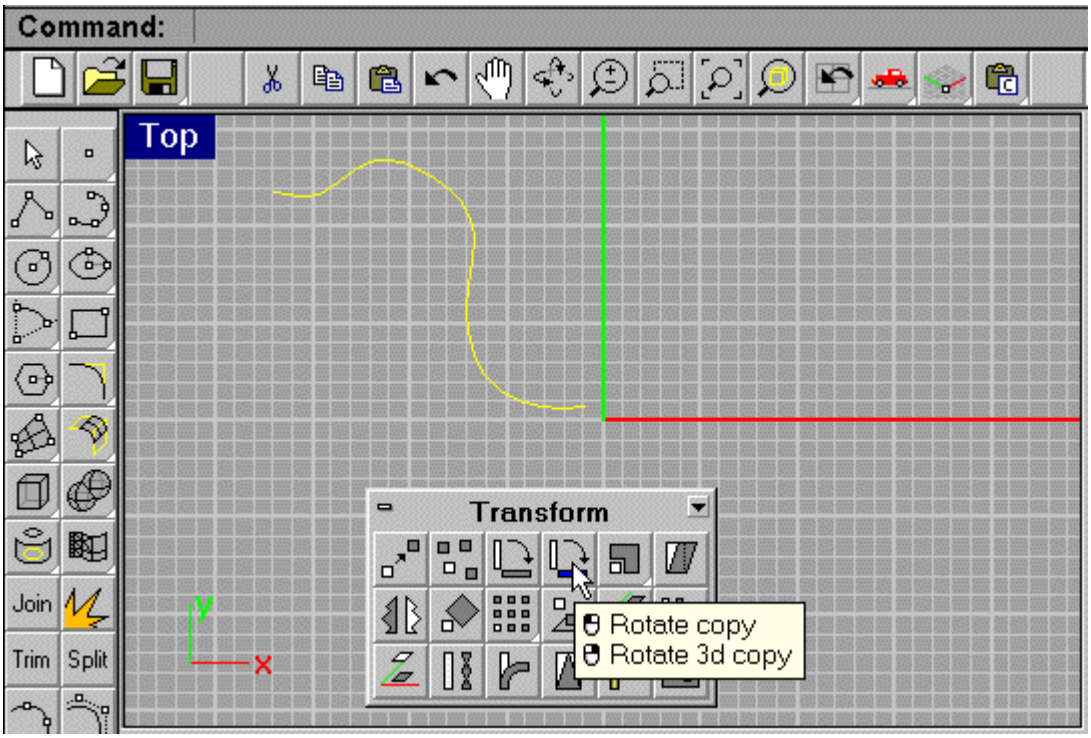


7. Click the ! InterpCrv icon

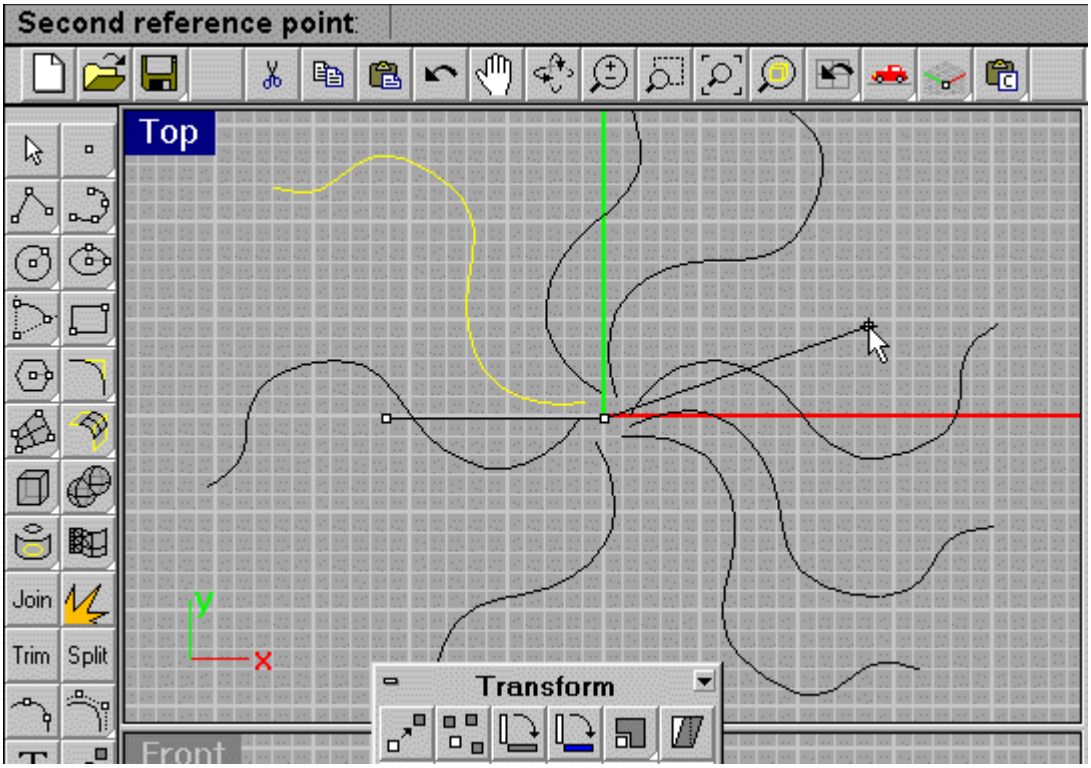
8. In the Top Viewport create a curve similar to left hand figure .
Pre-select the curve by left mouse clicking on it.
The curve should be highlighted in yellow.
Remember you need to select the curve first for these commandscripts to work.



9. Left mouse click the Rotate Copy icon you created



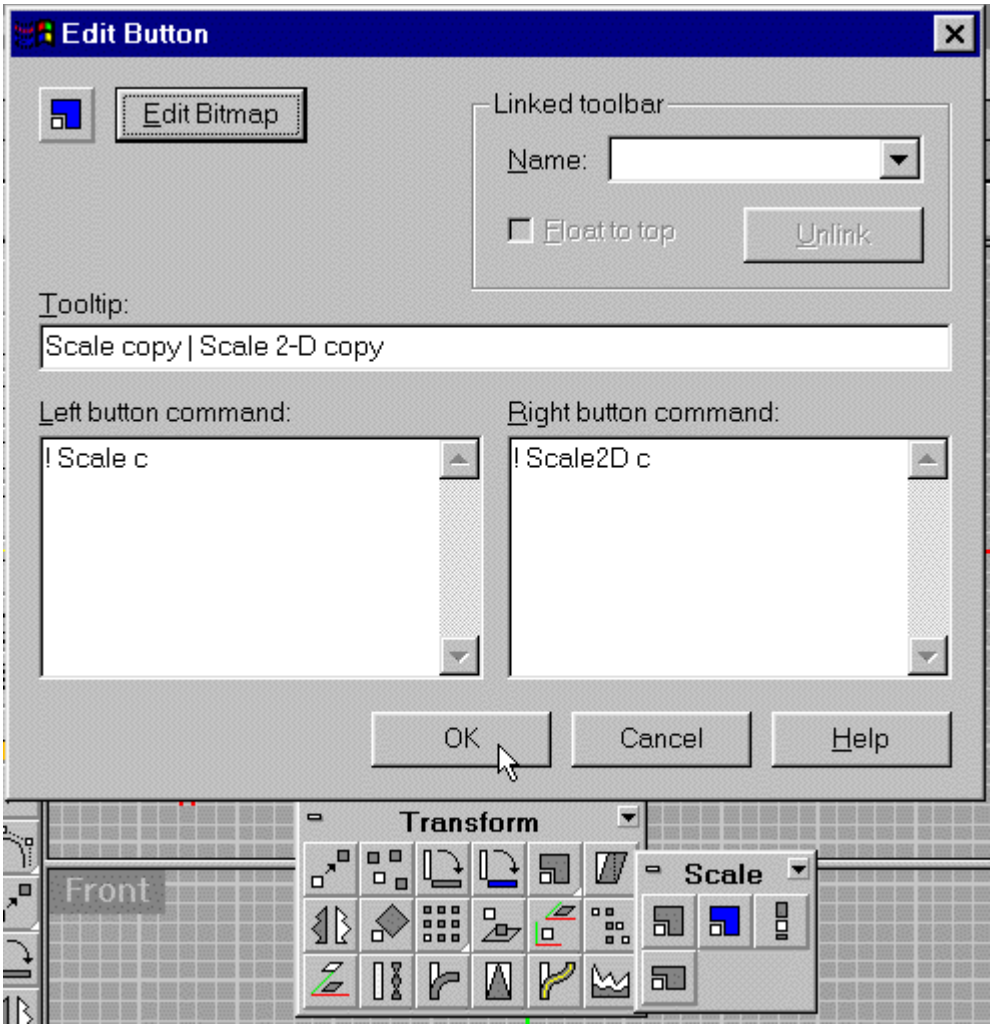
10. Click in the topViewport near the center of the Viewport.
11. Holding the shift key down to activate ortho click to the left of the last point.
Notice that now you can rotate and copy the curve at the same time. Make some copies.
12. Select all and delete the contents leaving the drawing free of objects for the next mini tutorial.



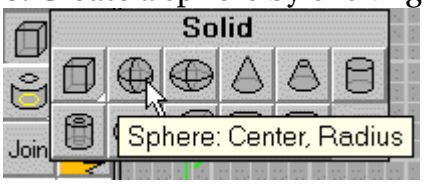
Scale copy Icons Tutorial

These scale copy icons function similar to the Rotate copy icons. The procedure is exactly the same as the Rotate copy icons so I will not illustrate the first four steps.

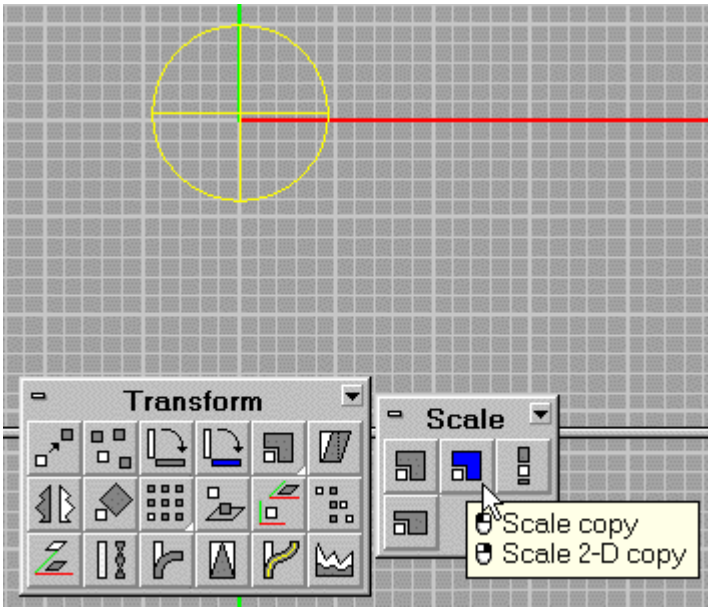
1. Drag the Scale toolbar out into the Rhinoceros workspace.
2. Hold the control key down on your keyboard and left mouse click over the Scale | Scale2d icon.
3. Ok to duplicate box pops up click yes.
4. Shift key right mouse click over the one of the duplicate Scale | Scale2d icons.
5. In the left mouse button box (see figure below) enter...
   ! Scale c
   In the right mouse button box enter...
   ! Scale2d c
6. Change the icons tooltip to read
   Scale copy | Scale 2d copy
7. Click Ok.



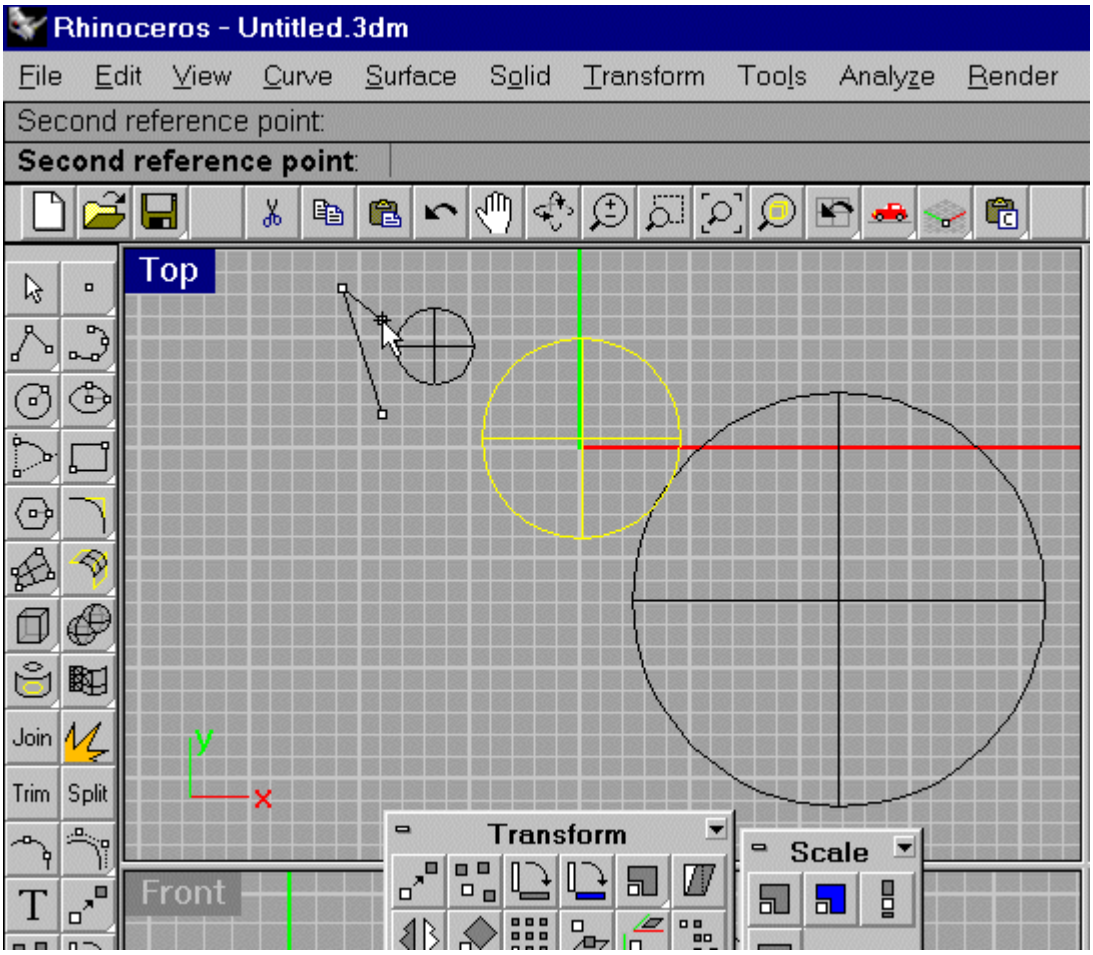
8. Create a sphere by clicking the Sphere Icon



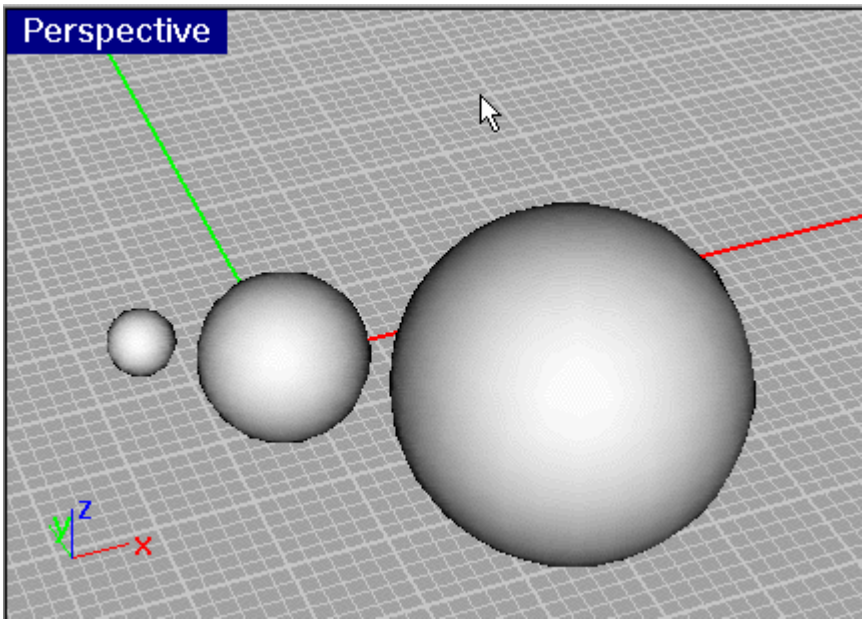
9. Select the sphere by left mouse clicking on it.The sphere should be highlighted in yellow.
Remember you need to select the sphere first for this commandscript to work.
Click the Scale copy icon you made



10. Click in the top Viewport to the left of the highlighted sphere.
Click roughly under the last point (see image to the left)
11. Create scaled copies.



Rendered image of the Scale Copy commandscript in action.
The orient, orient 3d icons all have the "c" option that you may want to customize too.



12. Select all and delete the contents leaving the drawing free of objects for the next mini tutorial.

Command scripts that control the "Appearance" of Rhinoceros

These Icons are good to have on your main toolbar since you can only access the appearance of Rhinoceros through the main menu then clicking tools/options / appearance, then your preferences. Therefore these commandscripts are great time savers for getting to that extra screen real estate quickly. This also helps when you want to see your model as large as possible on the screen either for viewing or rendering.

Toggle Menus on/off Icon

Toggle toolbars on/off with an Icon

Assigning a command script to a hotkey that toggles all your toolbars on/off

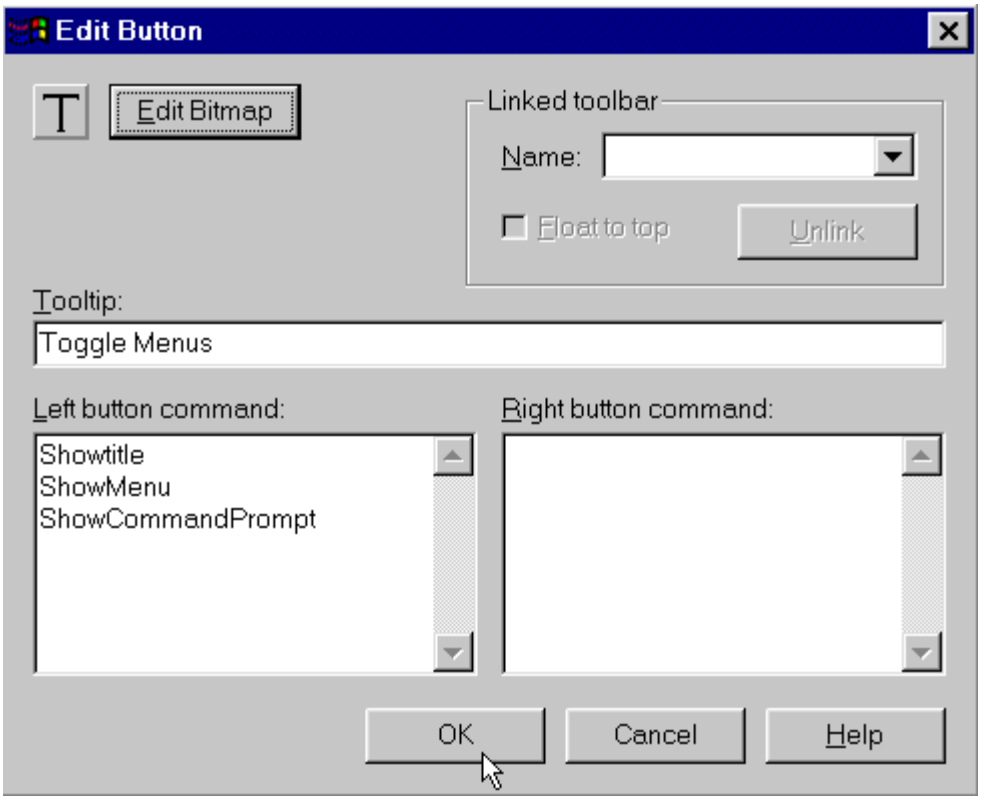
Toggle Menus on/off Icon

This commandscript is great to get rid of all menus and prompts in Rhinoceros and just use your icons. Then just click the Toggle menus Icon and you are back to normal. This will give you more room when you need it quickly.

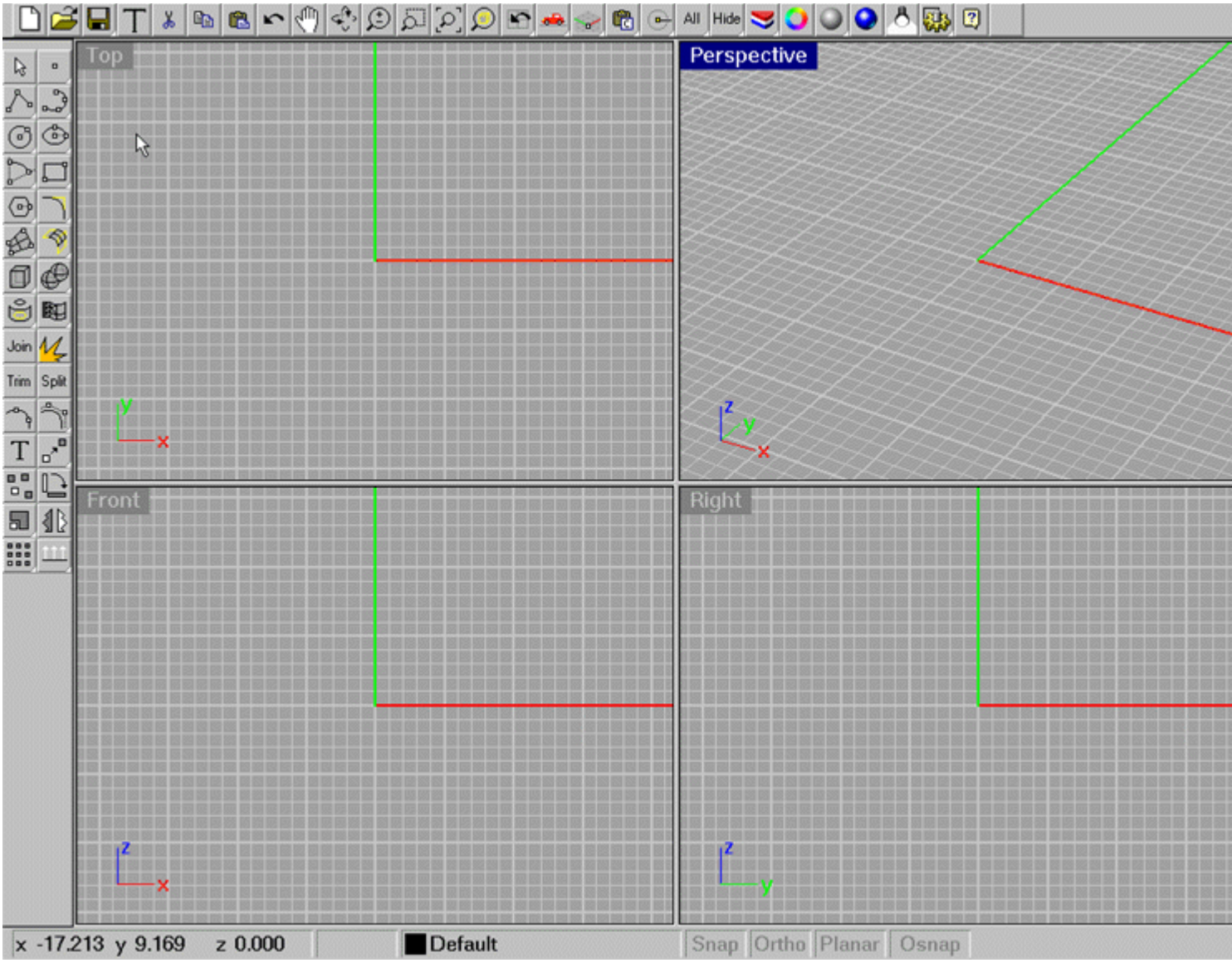
Shift key left mouse click over a blank icon from the Rhinoceros main toolbar.

1. In the Left button command box type or copy and paste the text below.
   Showtitle
   ShowMenu
   ShowCommandPrompt
2. In the Tooltips box type Toggle Menus

3. Click Edit bitmap and draw a simple T
4. Click Ok



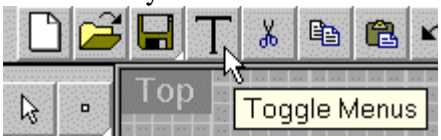
Click the Toggle Menus icon you just created the Rhinoceros workspace should look like the image below. Notice how much more screen real estate this gives you to work in. Click the Toggle Menus icon to go back to your previous settings and restore the menus.



Toggle toolbars on/off with an Icon

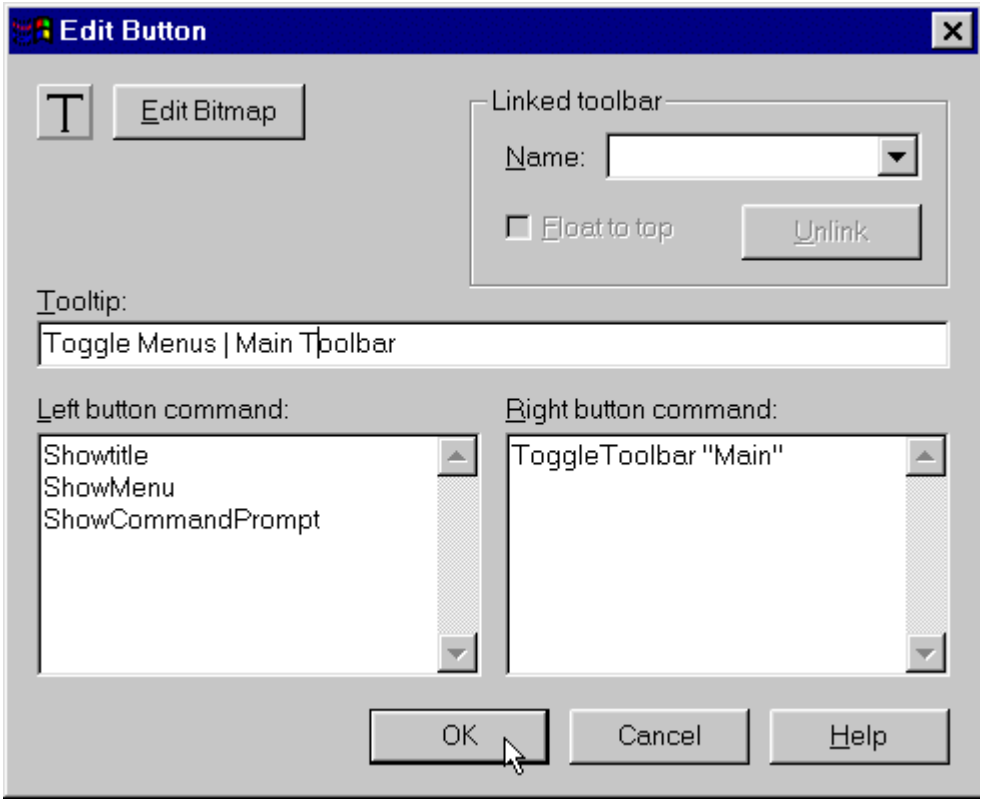
Toggling toolbars on and off can be done from icons too this lets you bring up whatever toolbar arrangement you can imagine. I have kept this one simple. This icon only toggles your left modeling toolbar off. This is why. If you click the Toggle Menus Icon you just made and then toggle off your left toolbar you will have much more screen real estate for checking your model and rendering in. This setup still gives you the main toolbar so you have rendering and other options as well as the icon to click to get back your left toolbar.

1. Shift key left mouse click over the Toggle Menus icon you just created.

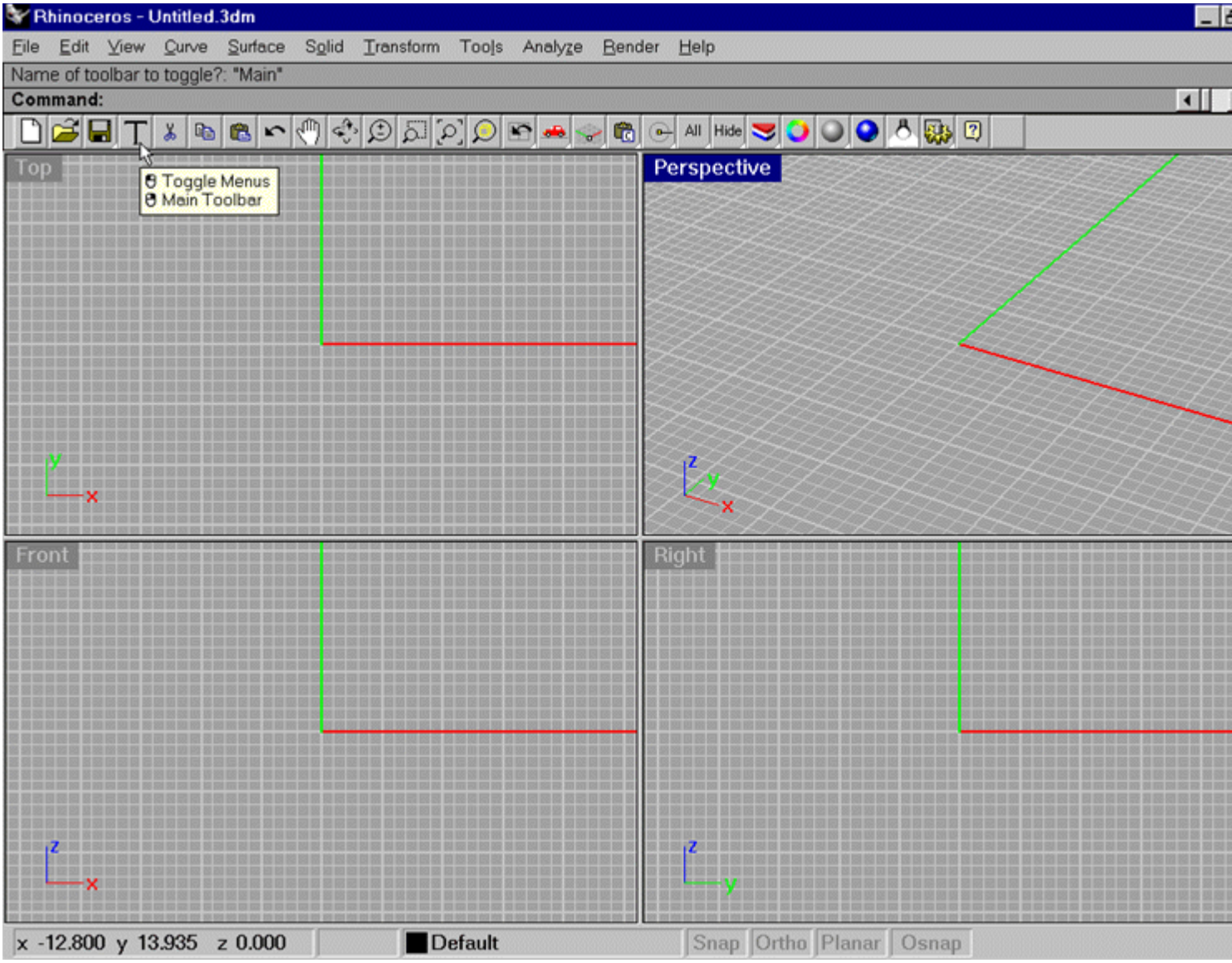


2. In the Rightbutton command box type or copy and paste the text below.
ToggleToolbar "Main"
3. In the Tooltips box type | Main Toolbar

4. Click Ok



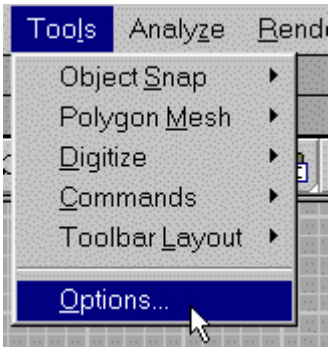
Right mouse click the Toggle Menus | Main Toolbar icon. The icon turns the left hand menu off (see image below). Right mouse click the Toggle Menus | Main Toolbar icon again to restore the Main menu.



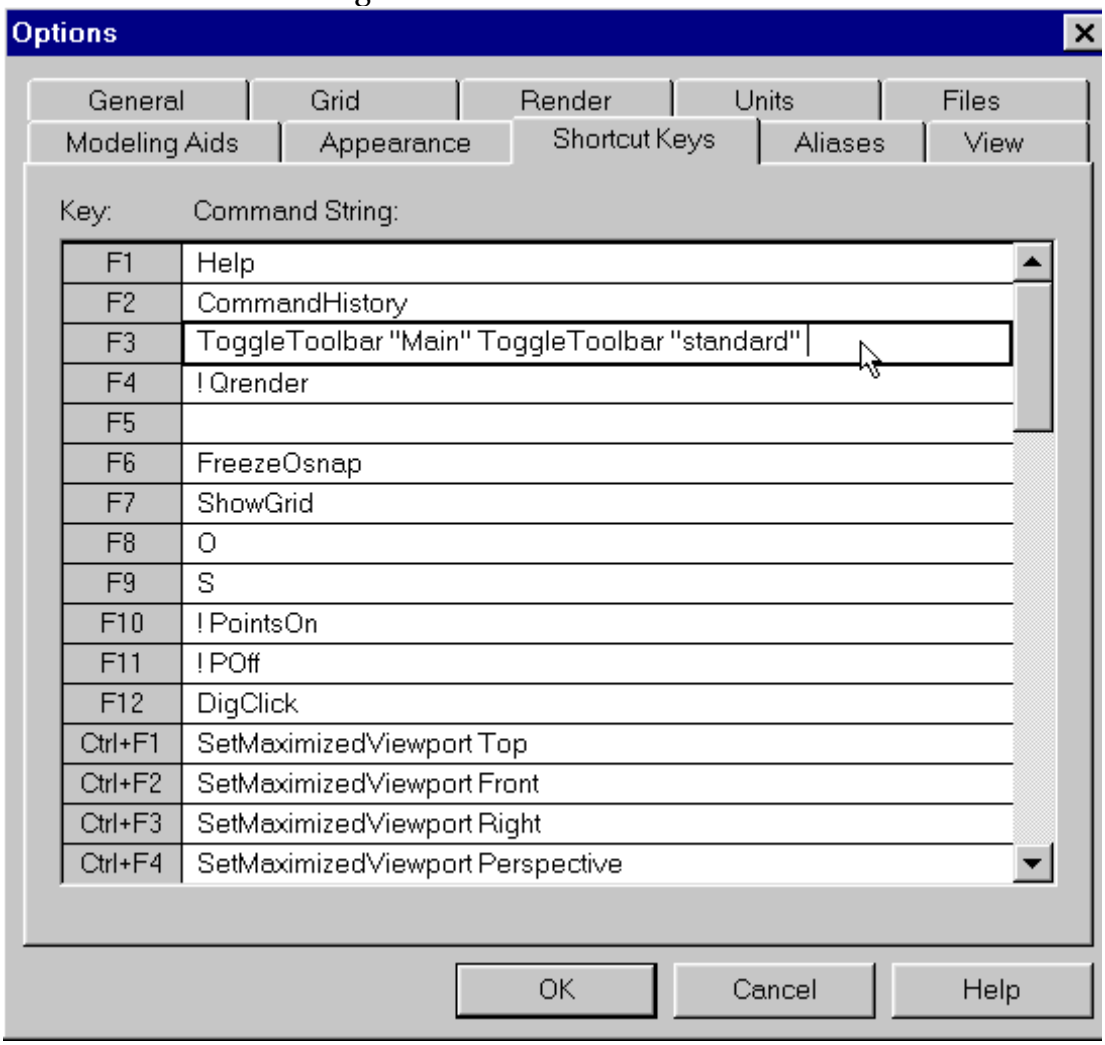
Assigning a command script to a hotkey that toggles all your toolbars on/off

Toggling toolbars on and off can be done from icons but in this case you will assign it to a Hotkey. This way you can click the Toggle menus Icon then hit the Hotkey on your keyboard with the alltoolbars off toggle and you will have one full big screen of only your beautiful model. This is great for presenting a model to someone and for moving around in the model.

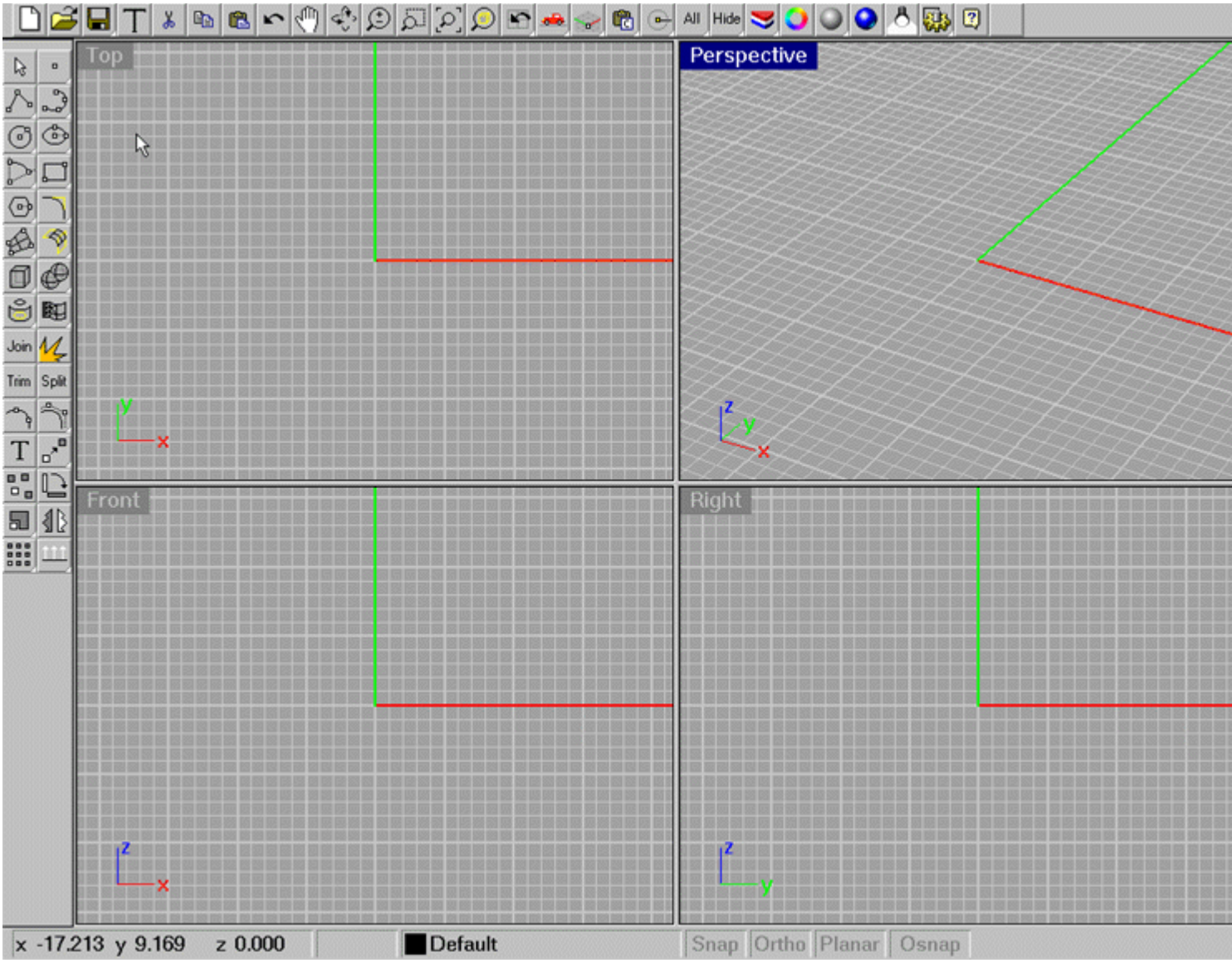
1. Go to the main Menubar and then click the Tools\options\shortcut keys.



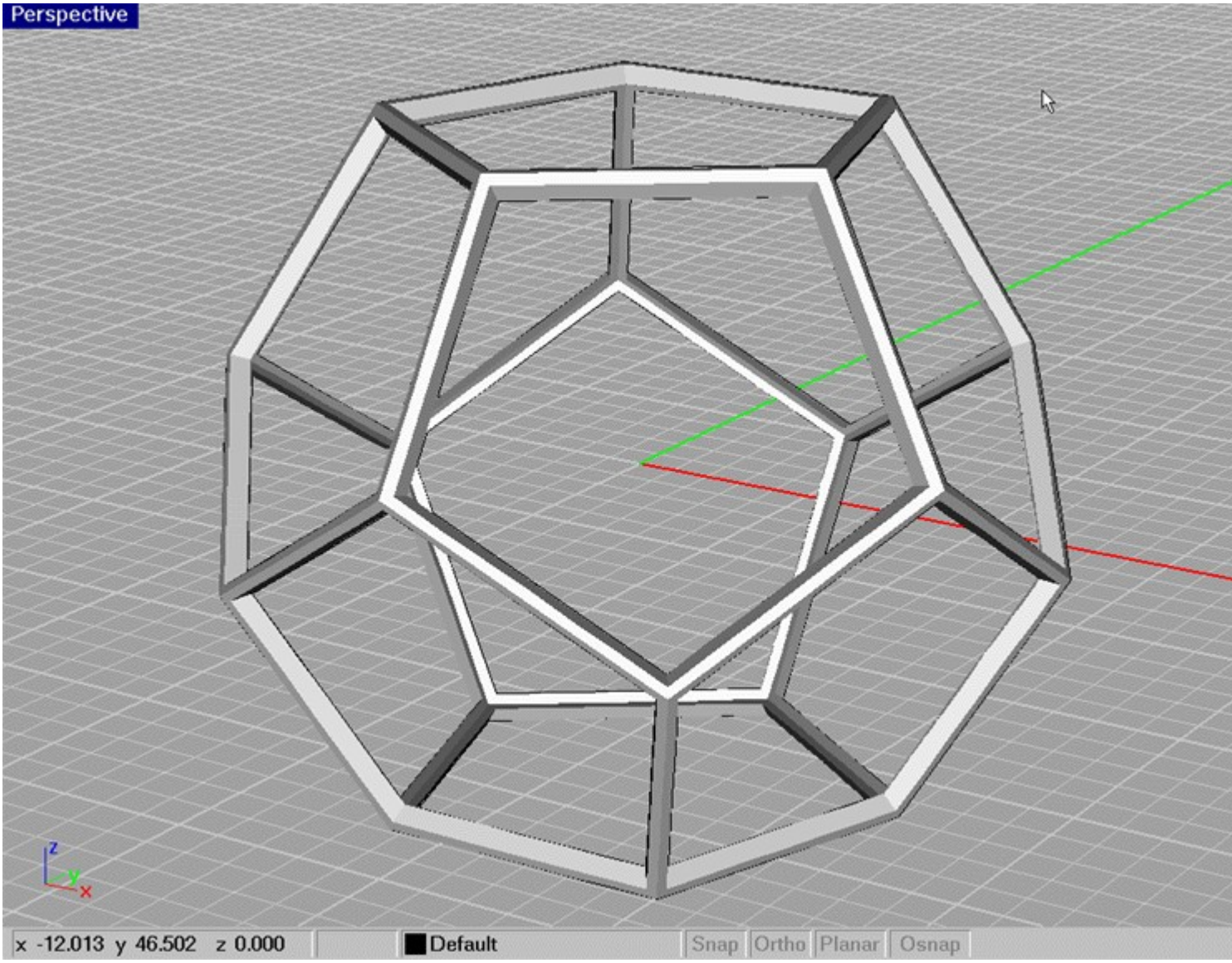
2. After the F3 hot key type in
ToggleToolbar "Main" ToggleToolbar "standard"
3. Click Ok to save settings.



4. Click the Toggle menus Icon you just created turning off all menus, see image below.



5. Now press the F3 hotkey you made. Notice how much screen real estate you have to move your model around in. The image below shows the perspective window maximized with a model present.



6. Press the F3 hotkey to bring back your toolbars
7. Click the Toggle Menus icon to bring back your toolbars.

Adding links to programs outside of Rhinoceros

These small commandscripts are very useful to know and to have assigned to icons. They streamline your work in Rhinoceros by giving you the outside applications you use most available at the click of a mouse button inside Rhinoceros. Some of these run commands will launch an external program like a bitmap editor or calculator right in the middle of modeling.

Adding the standard Windows calculator and your Image Editing program to Rhinoceros

Adding a link to Windows Notepad
Adding one important program to Rhinoceros *You must make a link to the standard windows Notepad to follow the rest of the tutorials even if you haven't created any commandscripts yet.*

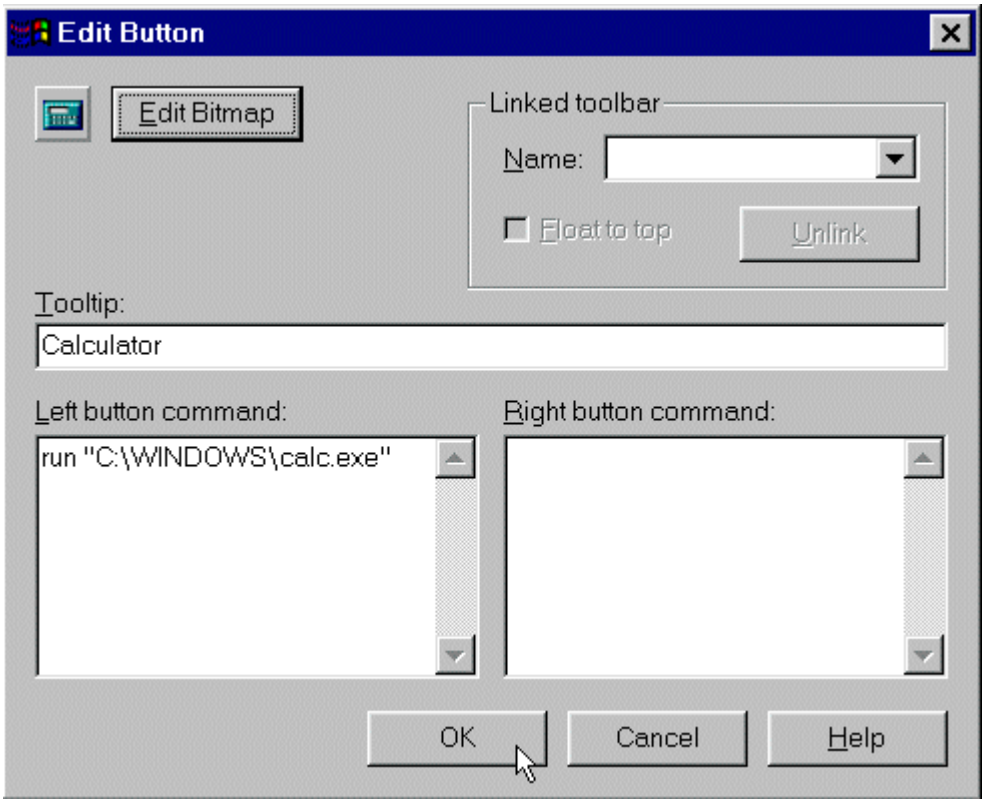
Adding the standard Windows calculator and your Image Editing program to Rhinoceros

The word "run" tells Rhinoceros to run an external .exe file. Notice the quote symbols around the path statement. This helps Rhinoceros to read the path statement because there are spaces in it and normally Rhinoceros would then read each word as command.

1. Go to the Render Toolbar and drag it out into the Rhinoceros workspace.
2. Duplicate the icon. In the left command button box add the commandscript:
run "C:\your photoeditor here"
3. In the tooltips type Image viewer and click the Icon to launch your thumbnail viewer or photoeditor .

One thing that Rhinoceros doesn't have yet is the ability to perform math calculations or the conversions of Standard Units (like Inches to meters) from one form to another. Having quick access to a calculator helps when you need to perform conversions or math calculations.

1. Shift key right mouse click over an empty icon
2. In the left command button box add the commandscript:
run "C:\WINDOWS\calc.exe""
4. In the tooltips type
Calculator
You can copy and paste the bitmap for the icon if you want to make it easier to identify.
5. Click Ok
6. Click the Calculator icon you just created to launch the Windows calculator.

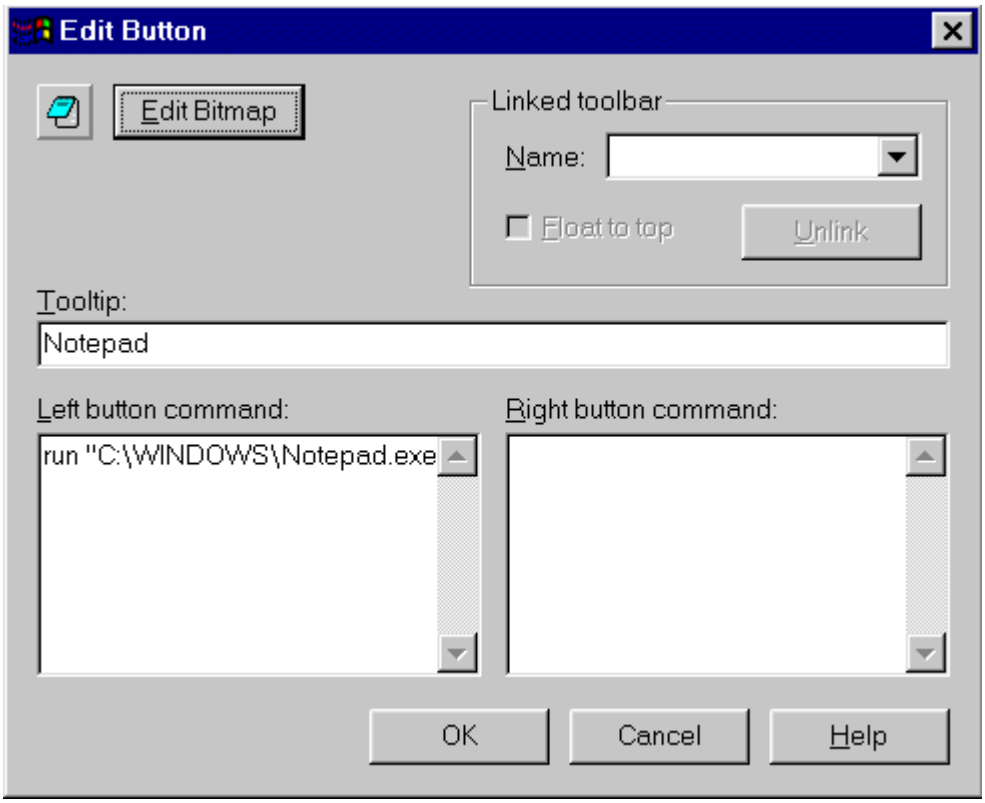


Adding a link to Windows Notepad

You will use the Windows Notepad to write your command scripts.

1. Click on a blank icon in the main toolbar

2. Type the commandscript beow in the left button command box
run "C:\WINDOWS\Notepad.exe"
3. If you wish Edit the bitmap and add the Windows Notepad icon
4. Click Ok



Assigning an often used commandscript to a Rhinoceros alias

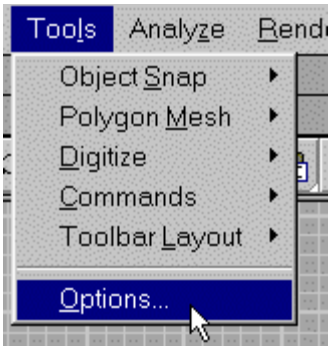
Rhinoceros aliases can be used to make command scripts smaller by substituting a word for a whole macro or part of a macro. You find aliases under the Menubar Tools\options\aliases. It saves on typing if you are repeating a whole macro often too. This word or alias for an entire script can then become part of another script. In fact a whole script can be composed of only command aliases which are really names of complex command scripts. Usually though I would create an alias of a certain script when you create a function that is universal to all or many different macros. Then you can include that command script easily in a different command script by typing its alias instead of the whole script.

This is a great example of a commandscript that should be made into a Rhinoceros Alias because it can be included in many kinds of different commandscripts. It is universal to all

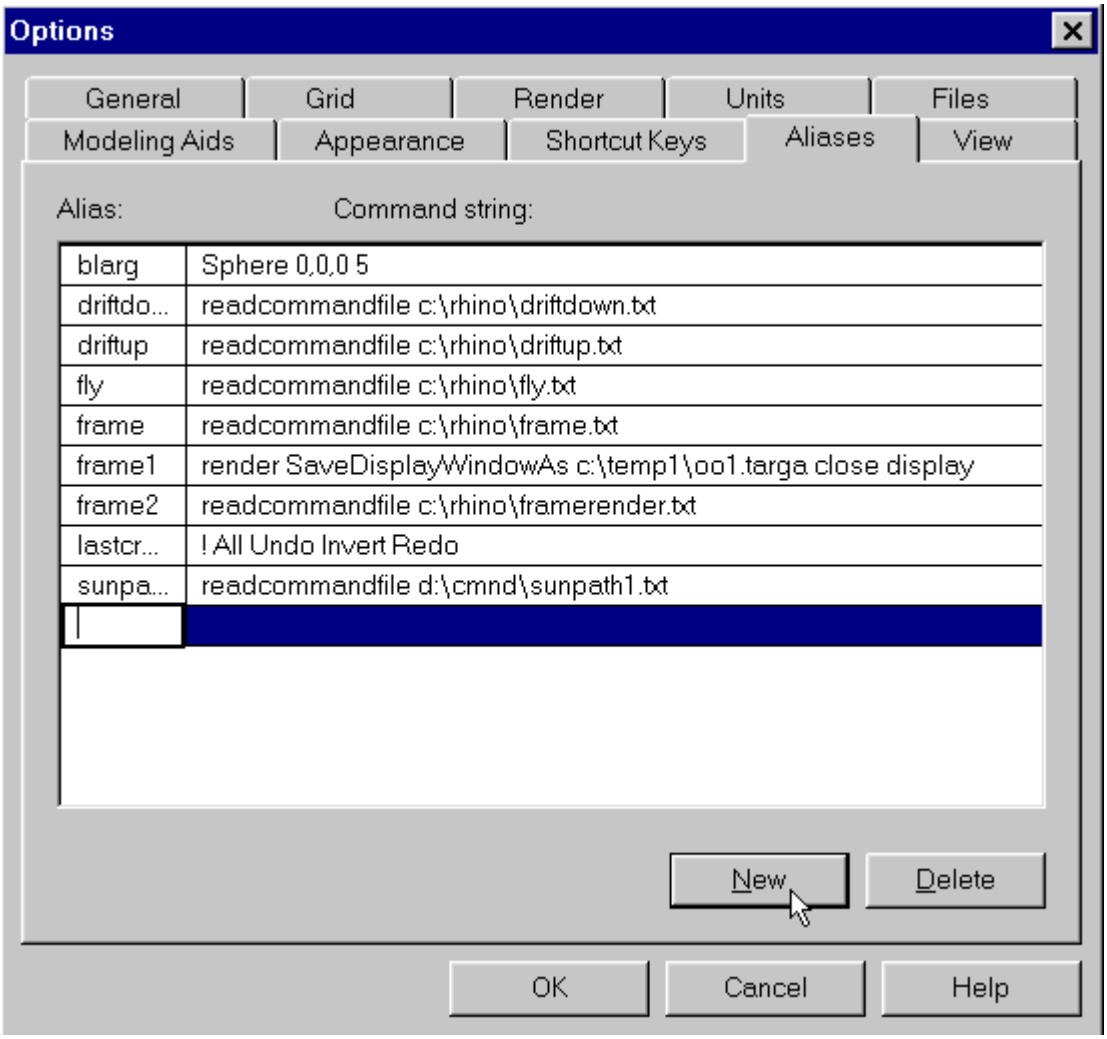
commandscripts. Rhinoceros is missing one often used 3d modeling command, that is the command to select the object you last made or created. The commandscript Lastcreated performs this function.

Important *you must have the command Lastcreated as an **alias** for the following chapter and for the rest of the command scripts to run.*

1. Go to the main Menubar and then click the Tools\options\aliases.

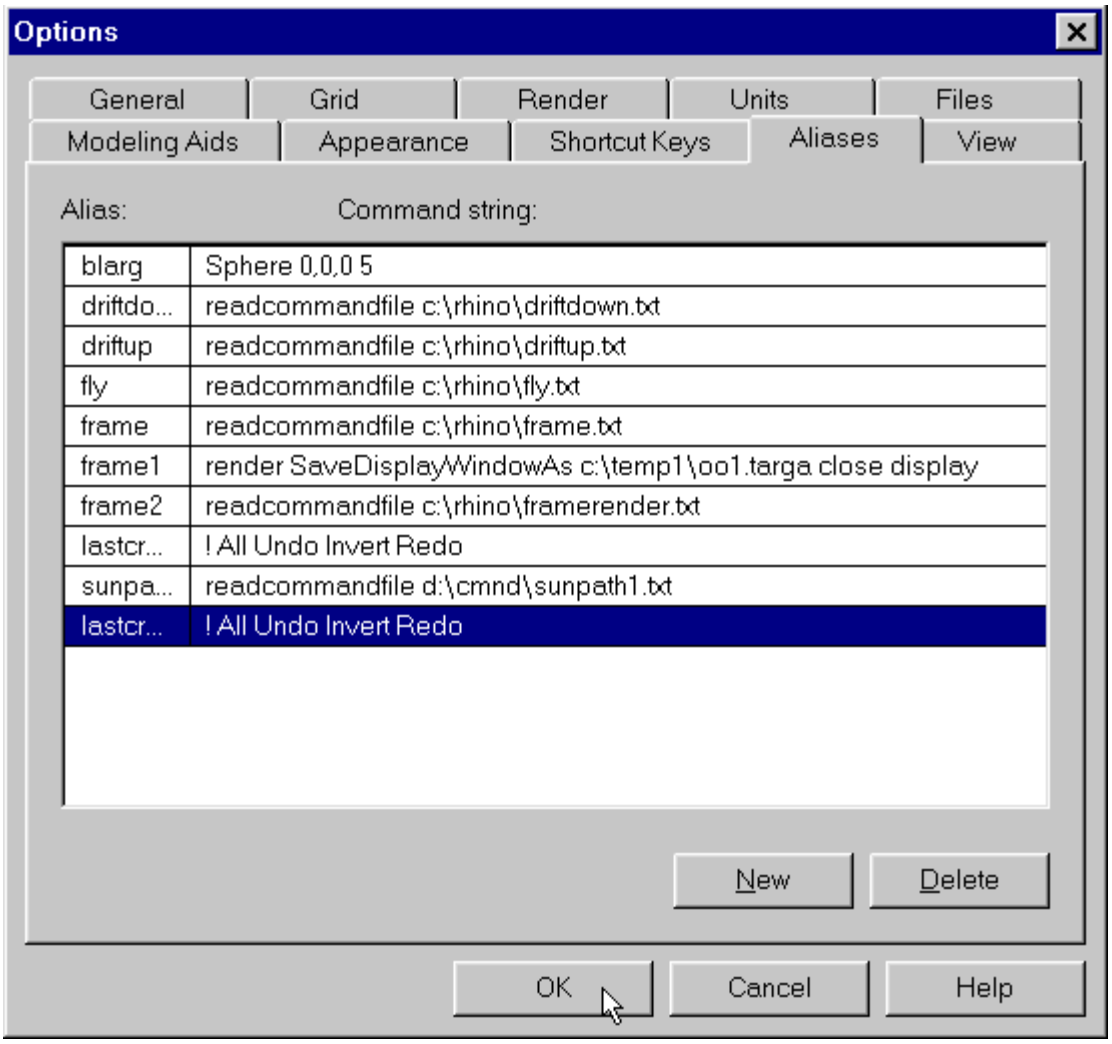


2. Click New

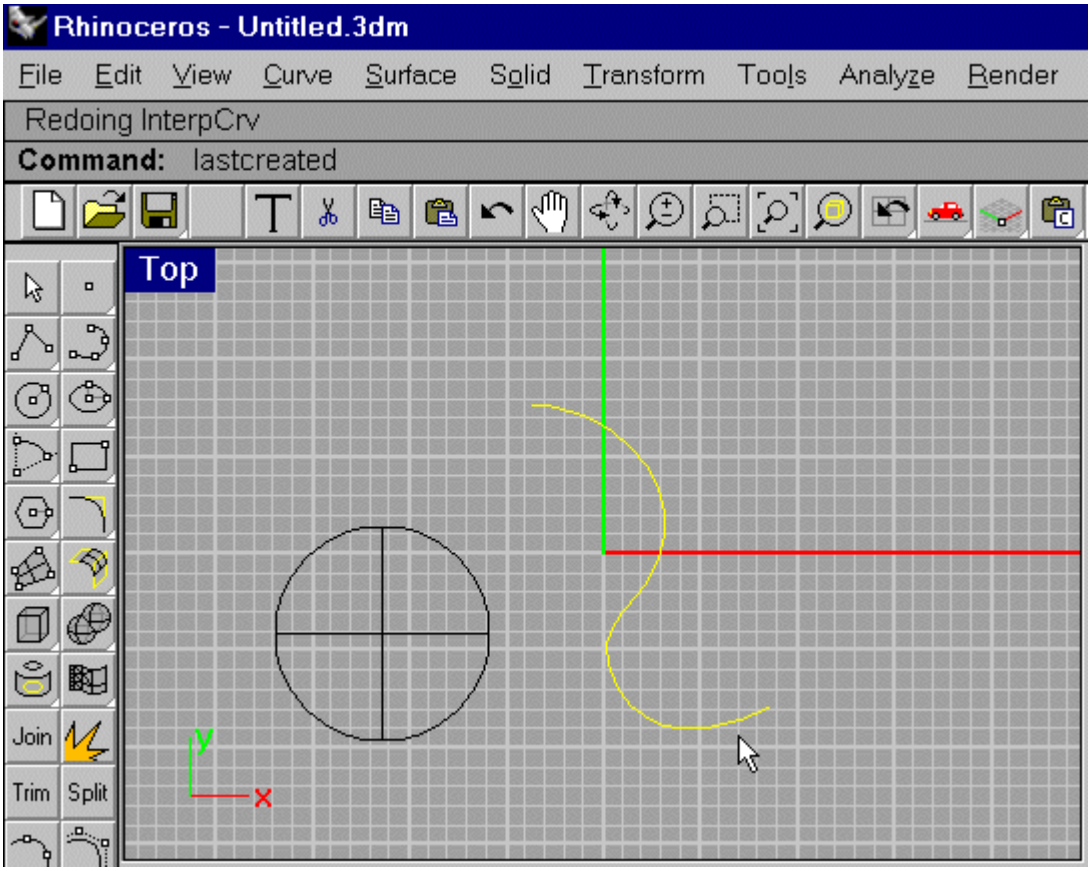


3. In the Alias Box Type in...

lastcreated
In the Command string box type in
! All Undo Invert Redo
4. click ok.



5. Create a sphere
6. Then create a curve
7. Click in the Rhinoceros workspace to make sure nothing is highlighted or pre-selected.
8. Type lastcreated at the Rhinoceros command prompt and hit the enter key. Notice it selects the last object you made or created, in this case the curve.
9. Assign the Lastcreated command to an icon if you wish.



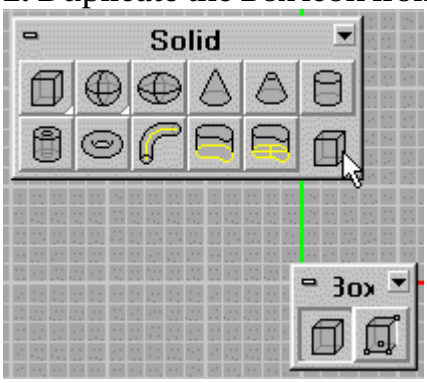
10. Select all and delete the contents leaving the drawing free of objects for the next mini tutorial.

Commandscripts that add extra Solids to the Solid toolbar

Even though Rhinoceros ships with a great set of solids. You may find that it is lacking some, or that you may want to make your own solids designed to fit your needs. This is easy to do the following examples show you how.

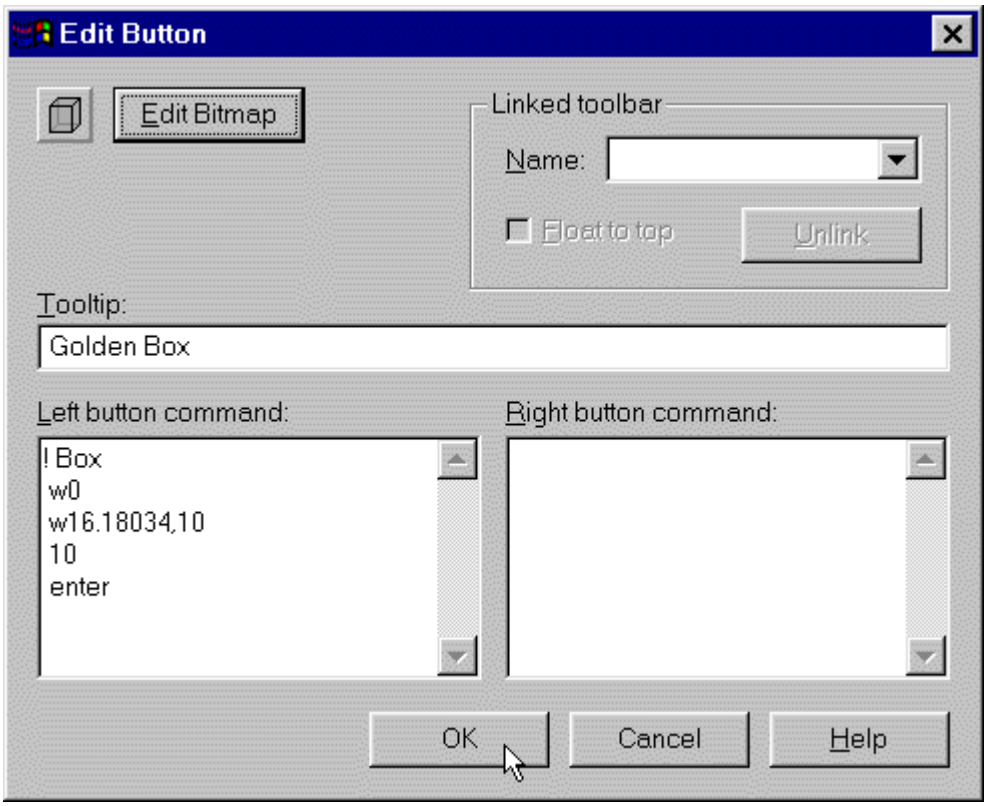
Adding a Golden rectangle box.

1. Drag the Solid toolbar into the Rhinoceros workspace.
2. Duplicate the Box icon from the box toolbar and drag /move it to the Solid toolbar.

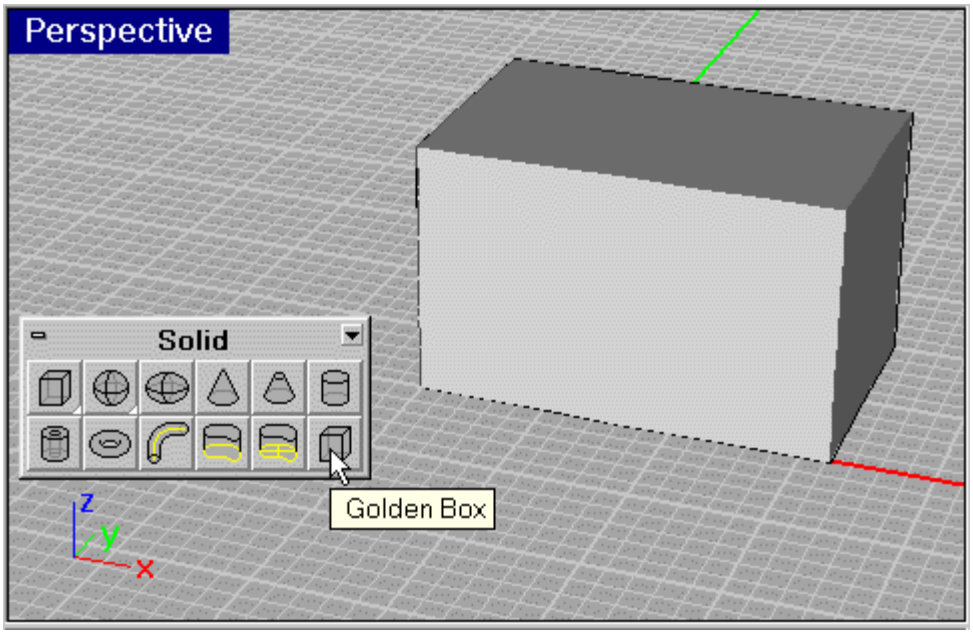


3. In the Left button command box type:
! Box
w0

w16.18034,10
10
enter
4. In the tooltips write :
Golden Box



5. Click the Golden Box icon to create a golden section box.

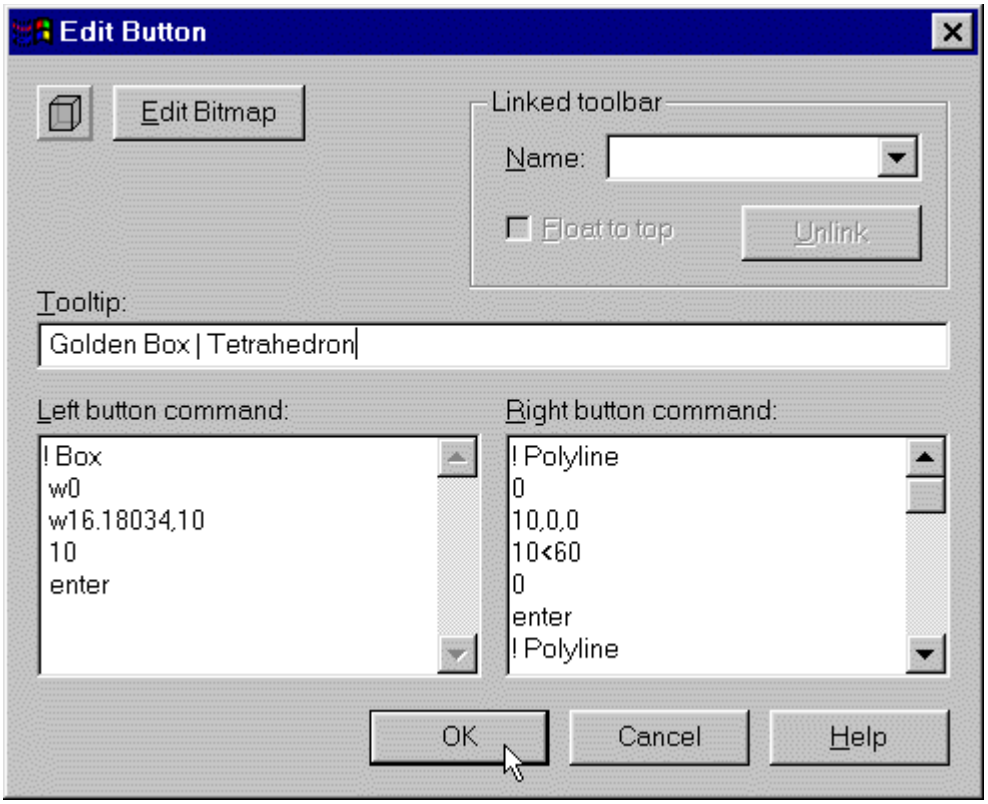


6. Select all and delete the contents leaving the drawing free of objects for the next mini tutorial.

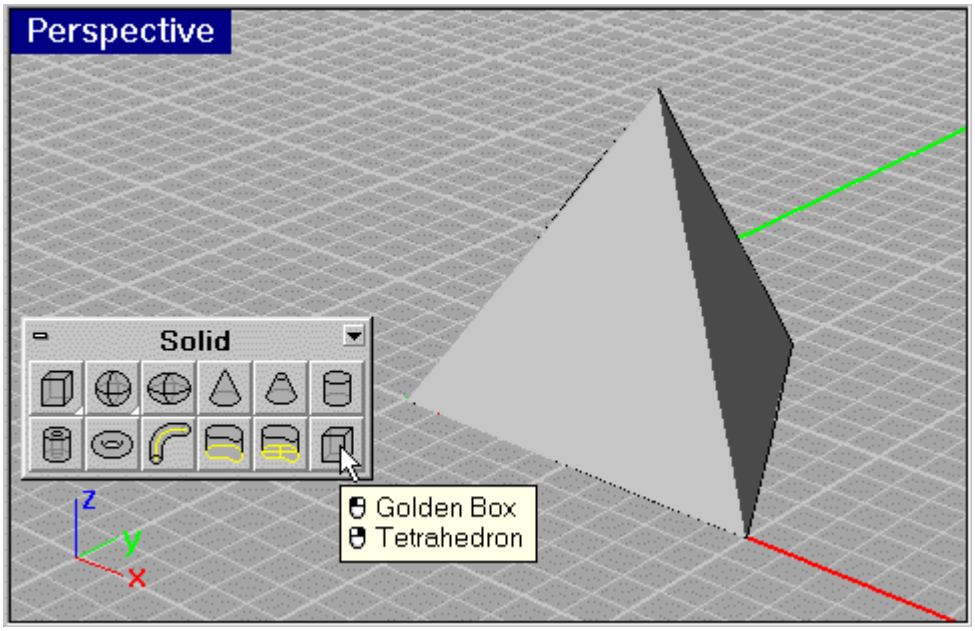
Adding a tetrahedron to the Rhinoceros solids toolbox.

Open the Windows Notepad and then open the file tetrahedron1.txt from the secrets of Rhinoceros CD.The commanscript is also located below you can copy and paste it from there too.

1. Copy the entire contents of the file to the Windows clipboard.
2. In Rhinoceros drag the Solid tool bar out into the workspace.
3. Edit the Golden box icon.
4. In the Right Button box paste the tetrahedron1.txt file or copy and paste it from the command script below.
5. In the Tooltips box write Tetrahedron.
6. Click Ok



7. Right mouse click Golden Box | Tetrahedron icon.



8. Select all and delete the contents leaving the drawing free of objects for the next mini tutorial.

Tetrahedron Commandscript

```
! Polyline
0
10,0,0
10<60
0
enter
! Polyline
0
10,0,0
5,2.88675,8.16497
0
enter
! Polyline
10,0,0
5,2.88675,8.16497
10<60
10,0,0
enter
! Polyline
0
10<60
5,2.88675,8.16497
0
Enter
! Selcurves
! PlanarSrf
! Selcurves
```

```
! delete
all
Joinfaces
enter
```

Writing Scripts That Build Models

One of the ultimate goals of Commandscripting is the creation of a complete model from a script. Besides the benefit of creating a model at the click of a mouse button, scripts that build models can also function as a parametric command history. This makes it easy to change dimensions in your script and see the changes reflect in your model. Although it is not true parametric modeling it gives a certain amount of parametric control that is now missing in Rhino.

By now you have a command over how to perform basic modeling functions in Rhino. You know the proper syntax of the scripting language. You have customized your workspace. You understand the 3 ways to run a command script and their differences. Below are outlined the basic steps to go from a concept model to a working and reliable commandscript.

The general steps for taking a concept model to a working commandscript.

1. Gather the information the dimensions and plans for the model you are building.
2. Build the model in Rhino.
3. Analyze and break the modeling process you used down into its basic components and steps it took to build the model.
4. Write a commandscript based on that analysis in notepad or any text editor making sure to save your script as a text file. For these tutorials I only used notepad since it takes up little resources to run, automatically saves files as text files and more importantly you only need Rhino and windows to write commandscripts.
5. Using commandpaste in Rhino run your commandscript to see how it runs. If it doesn't work use commandhistory to find out why it didn't.
6. Save your error free and working commandscript in a common folder where you place your scripts. You may also want to assign it to an icon or hotkey if it is an often-used script.
7. If you created new custom icons with commandscripts assigned to them. Save your workspace by typing savews at the Rhino command prompt. This way you won't loose your last created icons if you crash your computer.

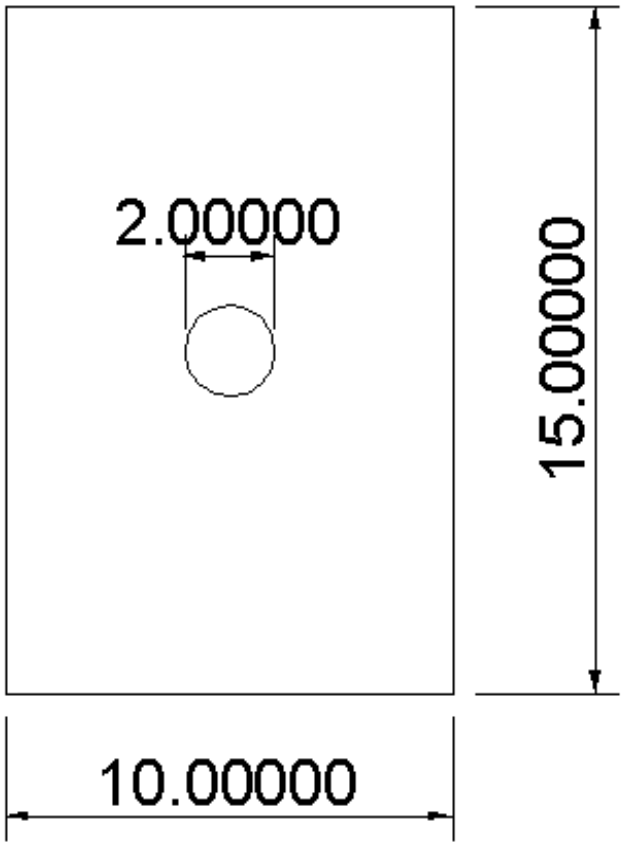
In order for you to grasp a hold of the process I will take you through a simplified case study of writing a commandscript that creates a model of a very simple Block manifold. I explain how to apply the general steps of going from concept model to a working commandscript, the syntax of the commandscript line by line, how to use the command script as a lowly form of parametric commandhistory and as a record to track changes.

Creating a manifold block: A simplified case study.

Suppose as a designer your firm has a client that needs many different types of the same model, thankfully this time their project is extremely simple a block with a hole or holes in it. The trouble is the design team can't seem to make up their mind about any of the particulars of the design so you must be prepared to constantly rebuild the model.

Scripting is great in a case like this instead of rebuilding the model every time and saving it to a new file you can manage its' design from scripts. The script also shows you exactly where the drilled holes are in the block and the size of your block at a glance so it functions as a lowly form of command history and a parametric modeler. You can also include a date in each file to track changes you made to the script. Imagine then that you had to generate many changes to the block size and drilled hole location you now have a way to solve that problem along with a way to easily track changes.

Analyzing the modeling process before you begin writing the script.

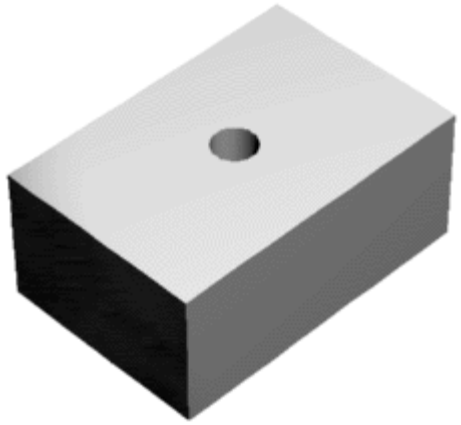


Above is a sketch illustrating the block manifold.

This fills the first requirement of making a command script that builds a model, that of gathering the information. In Rhino it is an easy matter to build the block. The first inclination would be to use the box tool for the block shape then create a cylinder the size of the cutout and Boolean subtract that from the block to create the cutout. Because Boolean operations cannot be used in Rhino scripts you can't use the previous example.

There is an easy alternative and one that doesn't use the Boolean subtract operation at all, so you actually save a modeling step. That is using the extrude command instead. The extrude command in Rhino is fantastic because it can recognize line shapes placed within

larger shapes as holes. In this case a circle placed within a rectangle then extruded becomes a block with a hole in it. These are easily scriptable procedures. See image below

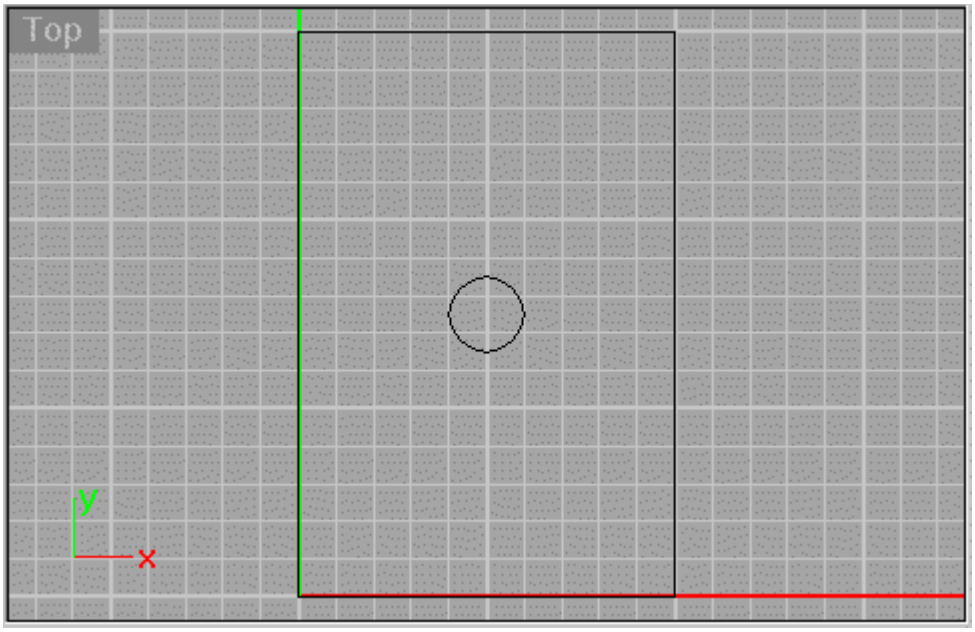


Taking out a piece of paper you write down the basic steps it took you in Rhinoceros using the extrude method to create the block manifold.
1. Rectangle
2. Circle
3. Placed circle
4. Selected curves
5. Extruded the curves

You have now filled the second requirement. You have found a scriptable procedure to create your model.

Writing the script

1. Open Rhino and then drag the Tools Toolbar scripting toolbar out into the Rhino workspace.
2. Click the your Notepad icon to open Notepad.
3. Then click the windows taskbar and then right mouse click choose Tile vertically, drag Rhino to the left to show more of its' workspace.



In the illustration above I created the rectangle and circle and then moved the circle to it's proper location, because these are the important parameters for my script.
I then open the command history window by pressing the F2 hotkey to see how Rhino processed the information.
I look for small switches in commands that I may need to be specific about. In this case there is the Diameter "d" switch or Radius "r" switch that I must address when using the circle command in a script.
I also look at the dimensions used to create the block and the numbers/values I used to move the circle.
There is also the cap option on the Extrude command that I need to be aware of.

Next mimicking the procedures that you view in Rhinos command history, you begin to write the script in notepad. The first step was drawing a rectangle 10 by 15 units, so that will be the first line of the commandscript. Notice in the commandhistory it says only Rectangle but if you shift Right mouse click over the rectangle icon to see what is written there you will see ! Rectangle. Remember the ! switch cancels all previous commands. In this case we want our Script to cancel all previous commands because you are creating a new model. The proper way is to write "! Rectangle" for the first line of the commandscript. You could easily cut and paste the syntax for rectangle right from the icon and then paste it into notepad as well to make sure that all the spelling is correct.

The first line of the script in notepad should read
! Rectangle

Notice that it is an apostrophe followed by a space then the word Rectangle. Test the first line of the script by highlighting the text and then copying it. In Rhino make sure the top viewport is maximized and click your commandpaste button. Rhino should prompt you for the next step in creating the rectangle, press the escape key to stop the command. This shows you that the first line was correct.

Tip Use command History to debug scripts
Here is how you debug a script or lines in a script. In notepad remove the space between the ! and the word Rectangle. Copy the text and commandpaste it into Rhino. Notice that nothing happens. Look at your command history to see what it says. Notice that it reports

"unknown command !Rectangle". This way you know for sure where and on which commands your script are hanging on. In this case the fix is simple just add the space back in.

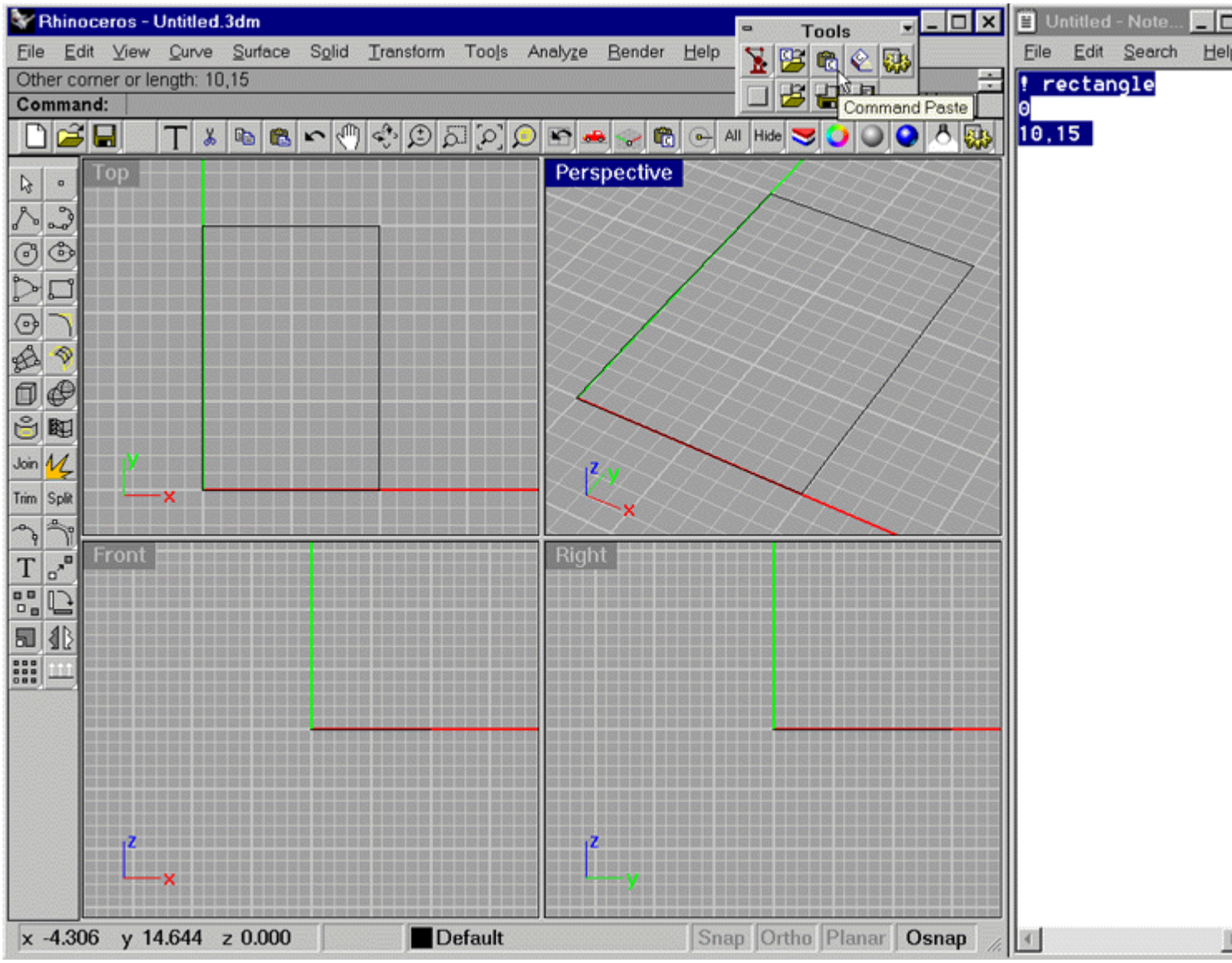
Continuing with the commandscript, you now need to enter the coordinates where the rectangle begins to be drawn from, in this case from 0,0,0. The shortcut for that in Rhino and in a commandscript is 0. We then know that the block is 10 units wide by 15 long.

Putting that information in the script we end up with the following lines:

```
! rectangle
0
10,15
```

Notice how I separated each action on a different line. You could write the script:
! rectangle 0 10,15.
However notice the ambiguity that this introduces. You need to make sure that there is a space between each succinct action because either a space the word enter or a new line acts like the user hitting the enter key as they carry out the command. How can you tell if the zero is part of the 10,15 unit settings or not? It's hard to tell. I highly recommend that you put each action on a separate line. It also makes the script easier to read and debug and lets the script flow in a linear fashion that mimics the activity that the Rhino command line makes when you work.

Copy the text from notepad and commandpaste it into Rhino. You should have a 10 by 15 unit rectangle drawn in the top viewport. See image. Delete the rectangle and continue with the script. If you have an error check the command history to see what it says.



Creating the Circle Cutout

Next you need to create the cutout or circle. Because the circle command is a completely new command we start it with the apostrophe ! symbol.
! Circle

You have a choice of creating the circle in it's proper location or anywhere in 3d space for that matter. I almost always create objects at the world 0,0,0 origin and then move the object to it's correct location. The reason is the two parameters that of the object's size and location can be controlled in your script easily without one effecting the other. Even though Rhino lets you create the circle or objects and their location at the same time, it really is a good idea to separate the two activities. Then you can change and objects size in a script without having to worry about if that change altered the placement coordinates. As you continue with the tutorials you will see how this gives you more control over placing, rotating and scaling of objects in a script.

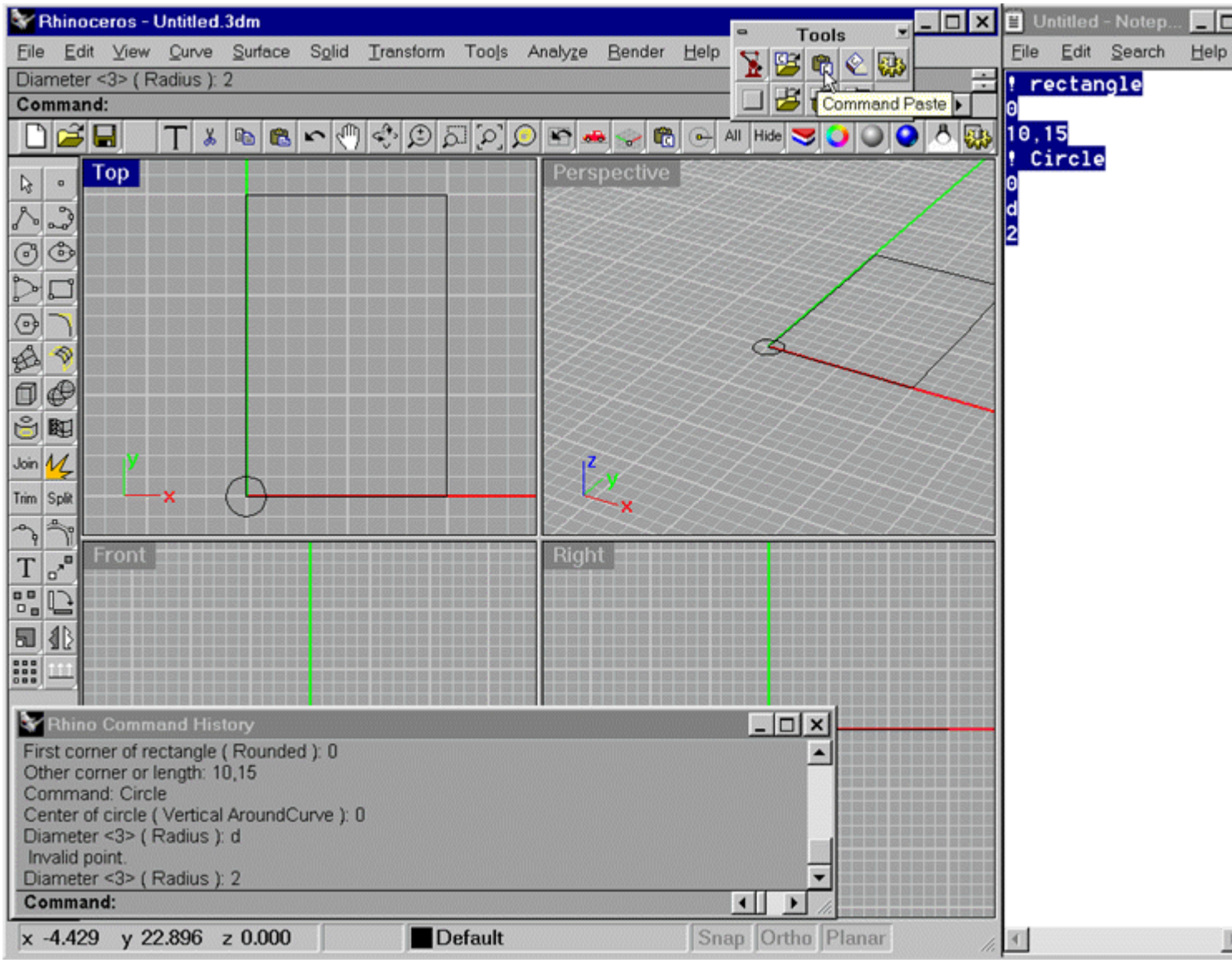
To continue the circle command you enter the 0,0,0 origin the same as you did for the rectangle so you can use the 0 shortcut instead. You are also faced with a command line

choice that of creating the circle based on its' diameter or it's radius. You will choose the diameter or D option in this case. This is a small quirk. If you already created a circle and you used the d option diameter at the command line, after running this script you will see a dimension not known statement in the command history after the d parameter line. This is not a mistake. What is happening is that Rhino internally stores your parameter switches for certain commands like a circle's radius or diameter during a Rhino modeling session so that you don't have to type that switch again. When you reuse the command you don't need to tell rhino to use the d diameter switch again. The command history is noting that Rhino doesn't need to know this command switch because it remembers that you are using the diameter switch already. However imagine that I used or you used the radius switch and then ran the script. The circle, which now has a 1.5 radius, would have a 3-unit radius. You need to be careful with these switches and make your script succinct in case you change parameters or you want to share your script with someone else. What you are in essence doing is you are totally telling Rhino what to do and not leave it up to chance.

Adding in the extra parameter switch and then the actual diameter measurement the script should now read:

```
! rectangle
0
10,15
! Circle
0
d
2
```

Test the command script thus far by highlighting and copying the text from Notepad, activate top view and click the commandpaste icon. See image below. Select all and delete.



Selecting and moving the circle

You need to select the circle to move it into place. At first glance this seems impossible because a select last command is missing in Rhino, but remember the command alias lastcreated or insert the mini command file this selects the entity you made last, in this case the circle. Type lastcreated into your script.

Your script should now read:
! rectangle
0
10,15
! Circle
0
d
2
Lastcreated

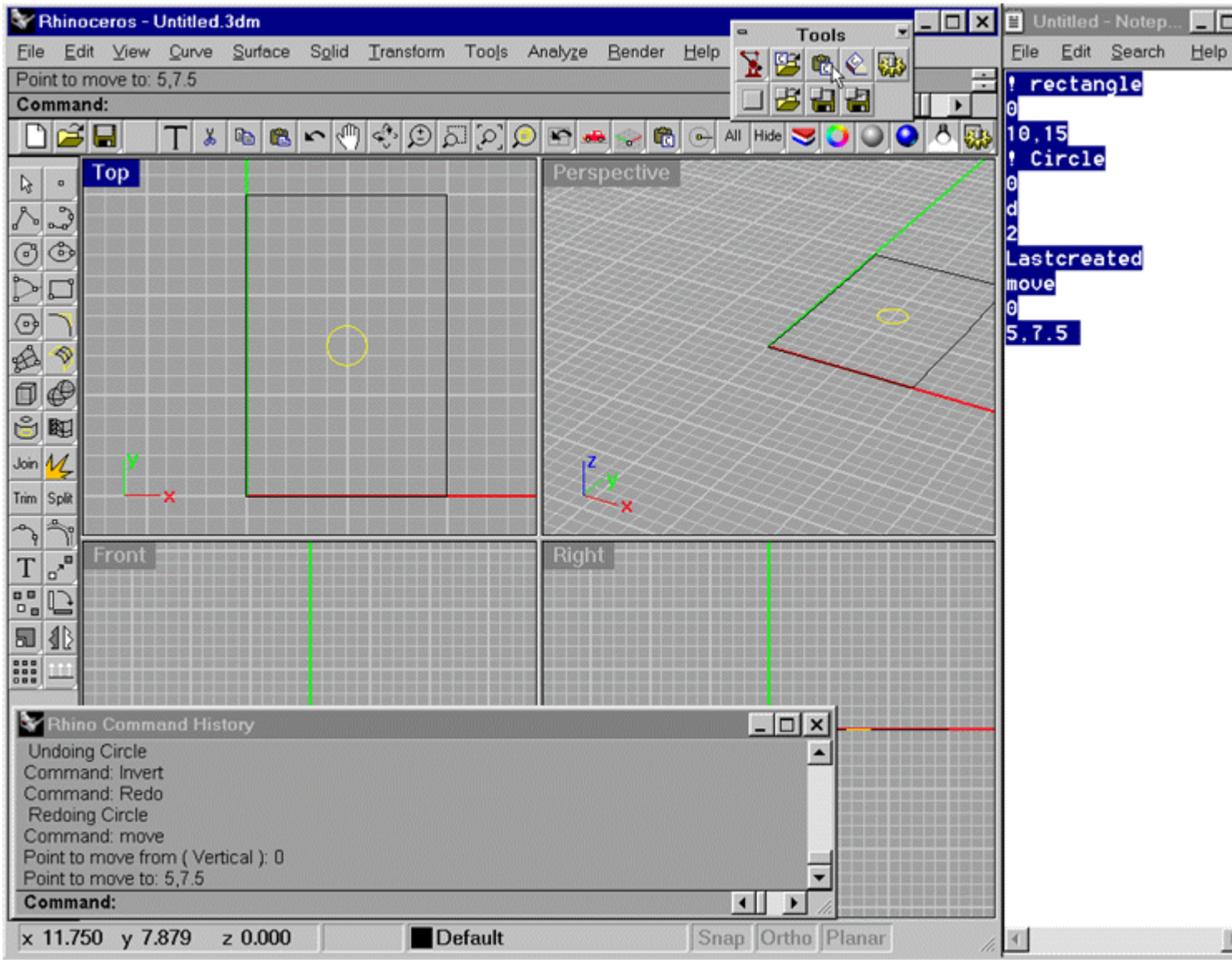
Test the command script by highlighting and copying the text from Notepad, activate top

view and click the commandpaste icon. This should build the rectangle and circle and then highlight the circle. Check figure004. Select all and then delete to clear the model.

Now you move the circle into place using the move command
0 Rhino moves the circle from 0,0,0 coordinate point
5,7.5 Rhino moves the circle to this 5,7.5 coordinate point

Your script should now read:
! rectangle
0
10,15
! Circle
0
d
2
Lastcreated
move
0
5,7.5

Test the command script by highlighting and copying the text from Notepad, activate top view and click the commandpaste icon. This should build the rectangle and move the circle into place. See image below. Select all and then delete to clear the model.



Extruding the Profile Curves

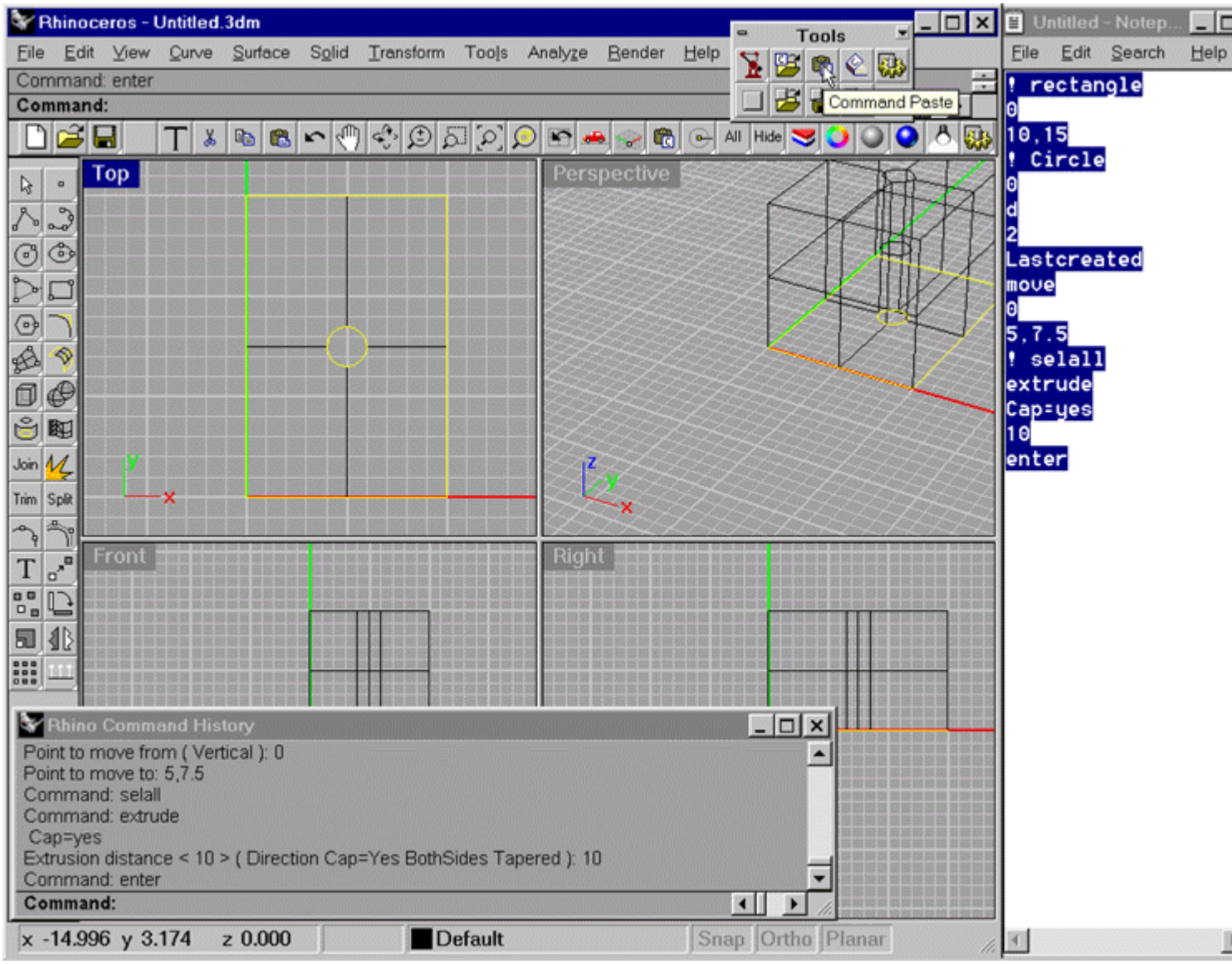
When you modeled the block you may have selected all the objects and then extruded them. This is how you will do it in the commandscript too.

Add these lines to the commandscript:
! selall
extrude
Cap=yes
10

Your script should now read:
! rectangle
0
10,15
! Circle
0
d

2
Lastcreated
move
0
5,7.5
! selall
extrude
Cap=yes
10
enter

Test the command script by highlighting and copying the text from Notepad, activate top view and click the commandpaste icon. This should create the manifoldblock but leave the curves that created it highlighted . See image below. Select all and then delete to clear the model.



Test the Commandscript

You have built the model and the script works fine. These added lines are just some finishing touches that add to the script to make it work cleaner. You can delete the highlighted curves and zoom in on the Block so it fills all the rhino viewports. In scripts that build models, I almost always end the script with the zoom extants all command. This way the script ends with only your model fitting perfectly in every viewport. IO always end the script with Enter this way Rhino knows for sure the script is ended.

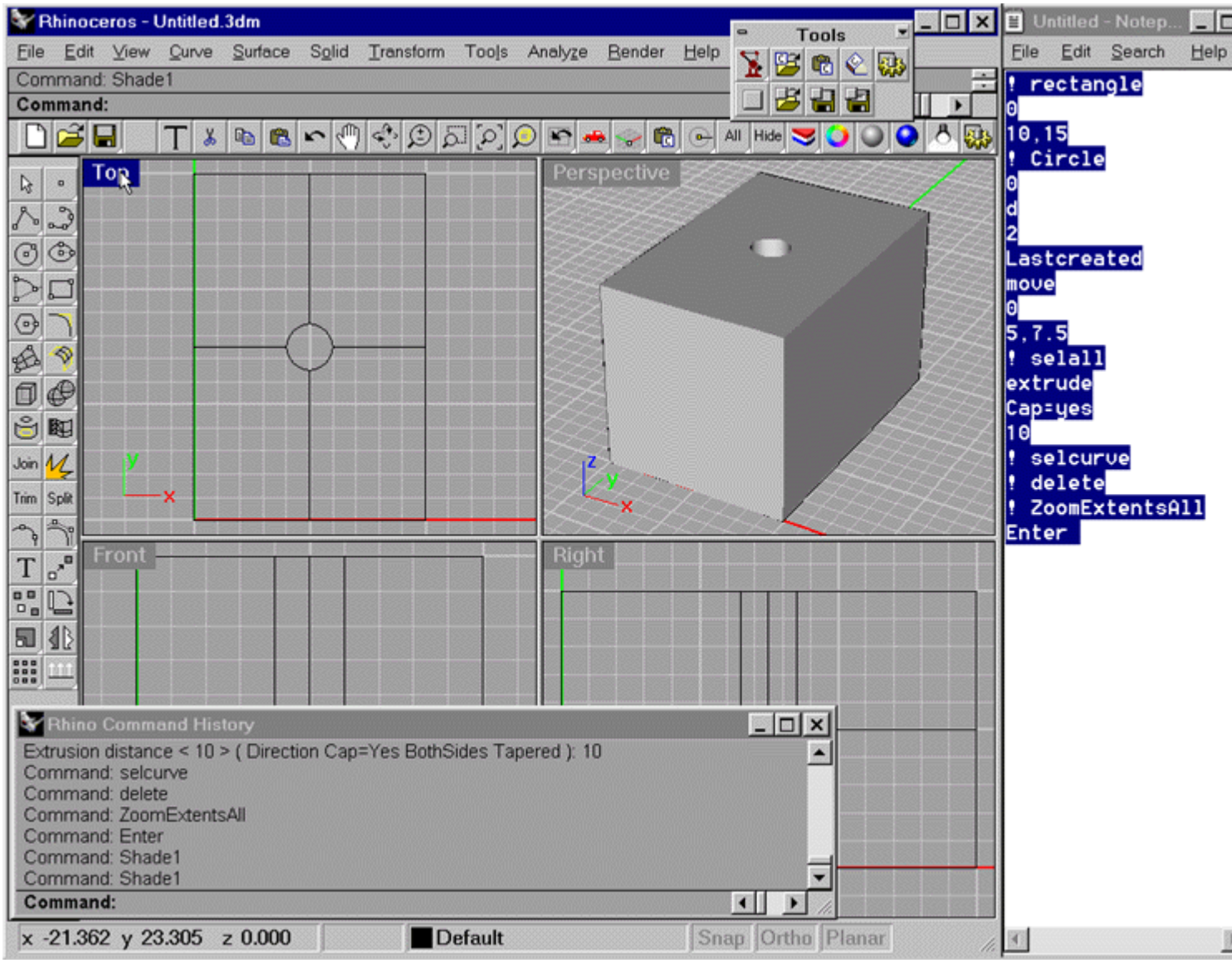
Add these lines to the commandscript:

```
! selcurve
! delete
! ZoomExtentsAll
Enter
```

Your script should now read:

```
! rectangle
0
10,15
! Circle
0
d
2
Lastcreated
move
0
5,7.5
! selall
extrude
Cap=yes
10
! selcurve
! delete
! ZoomExtentsAll
Enter
```

Test the command script by highlighting and copying the text from Notepad, activate top view and click the commandpaste icon. This should create the manifold block delete the curves and zoom all viewports on the created model.



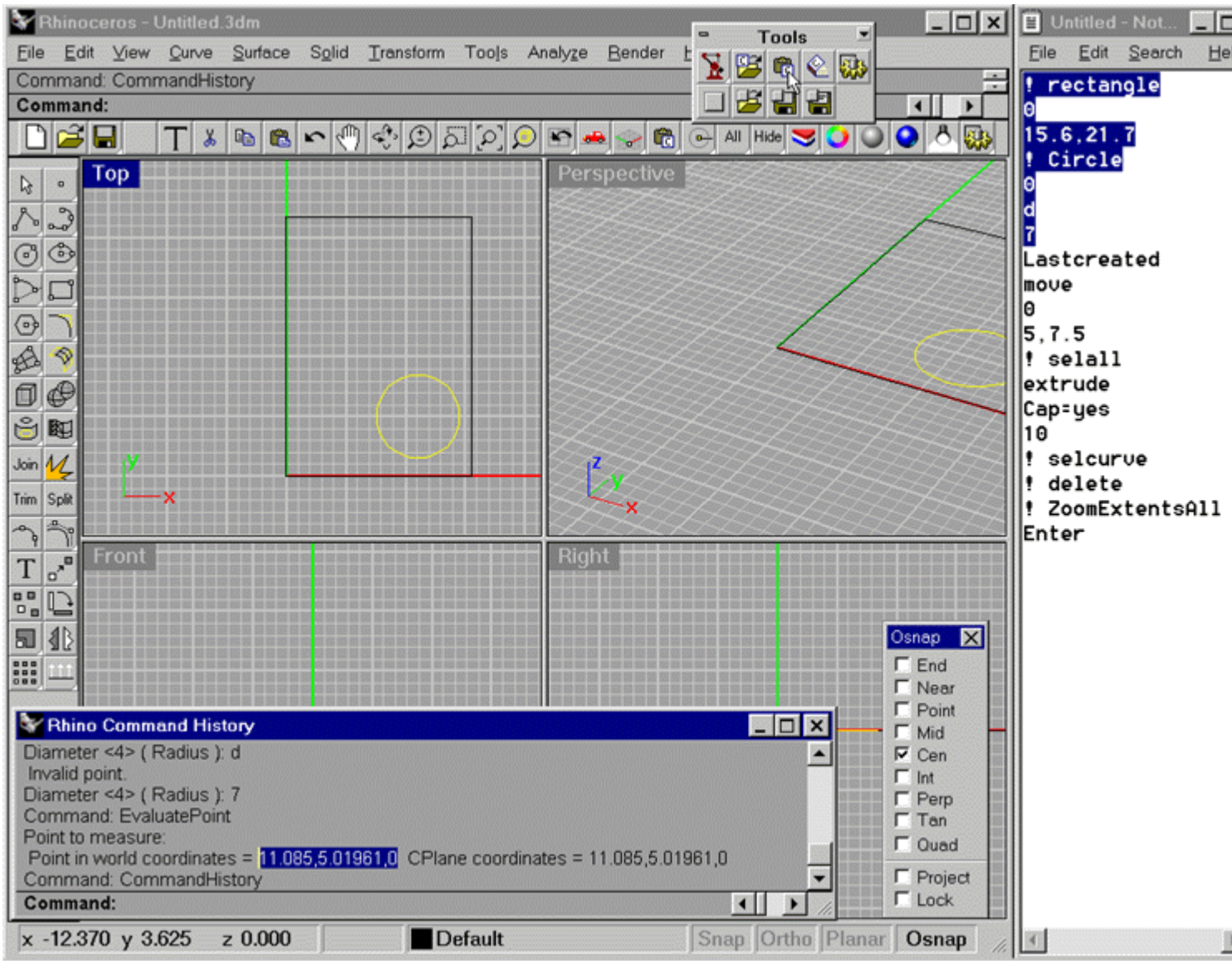
Press the F2 key and bring up the command history notice how the script answers the prompts that Rhino is asking. Quick shade the model.

Editing the Commandscript

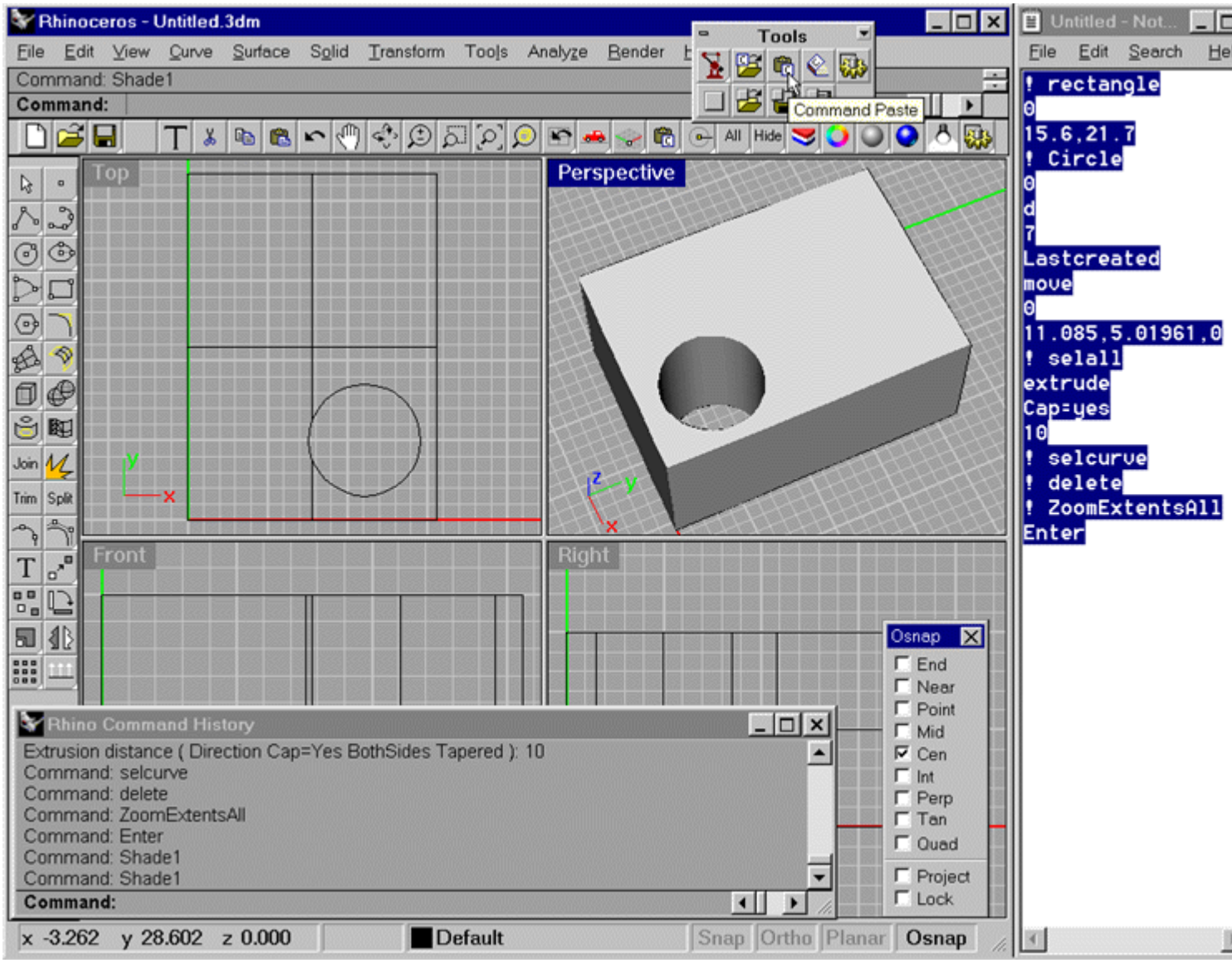
Now that you have control over the script and it works correctly save the script as a .txt file. You can easily alter the models' parameters and create a different sized model. You could add more holes, move the holes to a different location. You could combine this with other scripts or move the block anywhere in 3d space. Notice that by just looking at the script you can see the dimensions that tell you how big your model is. The script shows the cutout diameter and its' placement on the block at a glance. You can see that by easily changing the script you would build a different model. These are the steps you would take to change the commandscript. I have already changed the complete script for you in the table at the bottom of this page

1. In Notepad change the size of the rectangle to 15.6, 21.7 and the diameter or d to 7. See the image below
2. Highlight and copy the text up to the size of the diameter. See image

3. In Rhinoceros make sure the top viewport is active
4. Click the command paste icon
5. Move the circle to it's new location if you want to edit the cutouts location.
6. Using the Analyze Point command and snap to center find the circles' location.
7. Open the command history by pressing the F2 hotkey on your keyboard. From the command history copy the circles location to the windows clipboard. Notice in the image below in the command history the world coordinates are highlighted in blue ready to be copied to the windows clipboard. **Reminder* In Rhinoceros you cannot use the standard Windows right mouse button copy method. So how does one copy the text in the command history? Highlight the text using your mouse and then use the standard Ctrl key plus c key method to copy that text. Oddly enough as you may have noticed Rhinoceros lets you use the standard Windows right mouse button copy method when you edit icons or use the notes function.*



8. Past the circles' location into the appropriate spot in the commandscript in Notepad.
9. Select all and delete the contents in Rhinoceros.
10. Make sure topviewport is active
11. click the command paste button



Here is the complete script you can copy and paste it from here if you wish. The changes are indicated in red.

```
! rectangle
0
15.6,21.7
! Circle
0
d
7
Lastcreated
move
0
11.085,5.01961,0
! selall
extrude
Cap=yes
```

```
10
! selcurve
! delete
! ZoomExtentsAll
Enter
```

Using Commandscripting to Create Complex Models and Scenes

Commandscripting excels at the creation and management of large complex models and files. Many of you may have created beautiful models that you wanted to combine into larger more complex models and scenes only to be thwarted by lack of machine power and time and patience. After all waiting ten minutes for a screen redraw or property change can deter even the most hardy of us from working with huge files. We realize that our machines are capable of so much more but who has the patience or time to work with these huge files and squeeze the most out of our resources?

The complete scene you see above was never saved to file. I never needed to. In this case I used commandscripting to evolve and render this very complex and high surface count model. It was actually faster to create the model by running the Commandscript than it would be loading the complete saved file from hard drive.

The main secret to creating complex scenes or models is the import file merge command in Rhino. By combining many smaller lesser models or components you file merge them into one complex model or scene. Sounds easy you say you don't need command scripting to do

that you think. True but commandscripting allows you to really exploit this feature and to get the most out of it not just use it off the menu.

The important added benefit is that now you can combine many different what if modeling scenarios and just run them from scripts. Bonus is that commandscripting can set up and create complex renderings that can be executed when you are away from your computer. Another plus is that you can set up and manage levels of detail depending on the requirements of your scene or model. Commandscripting gives you an easy way to substitute reuse and replace parts of models or different components depending on what your particular project demands. For instance in the scene above I could substitute more detailed columns or real walls with studs and dry wall sheets instead of simple 3d planes. Let's face it there is a polygon limit but this technique helps you to logically get the most out of your machine. Once you have the placement coordinates for your parts or models you only need to change the files that are referenced to those points in the script in order to change what is in your model. This is much more efficient and easier than changing them by hand.

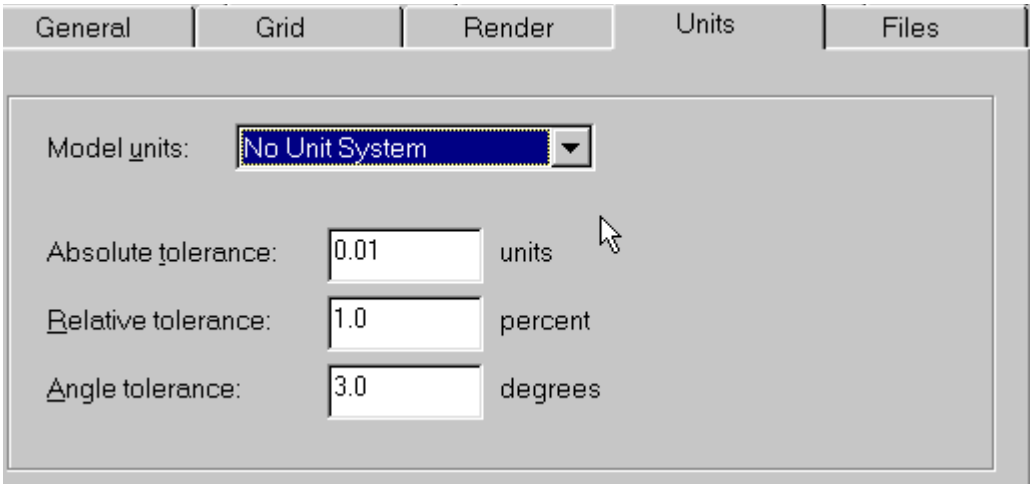
You will build on the techniques you learned in the last chapter when you were cutting and pasting 3d coordinates into your Commandscript and placing the cutout holes in the manifold block.

Even if you are not an architect or interested in historical architectural design this chapter will teach you skills that can be applied to any complex model or scene not only architectural subjects. I will show you how to script and build the two models as well as variations on them. The first model is a rotunda or circular temple. The second is a Pompeian courtyard. You will reuse the same components and learn how using them with different Rhino modeling commands will result in creating entirely different models.

Settings for this chapter 3

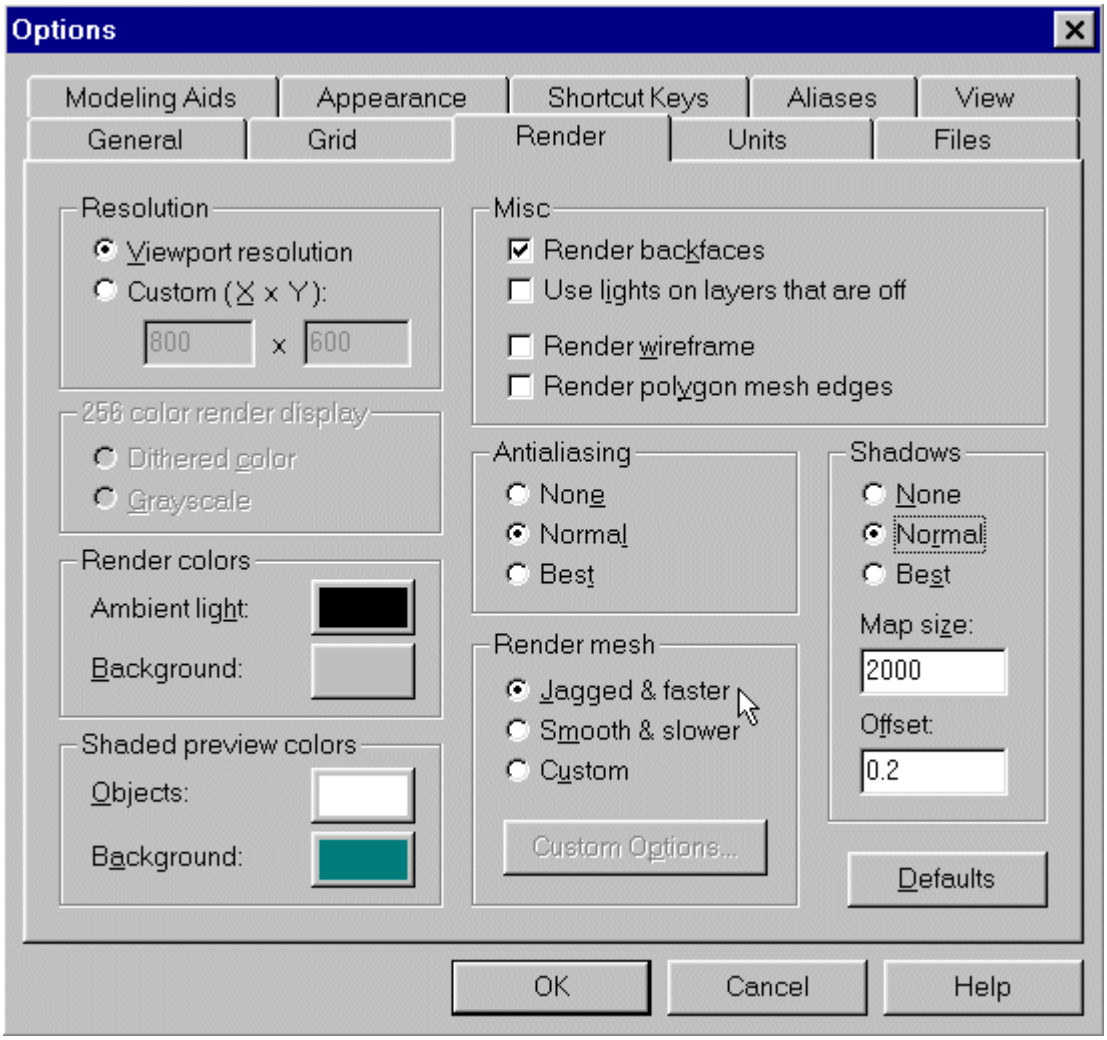
Make sure that you have the following settings:

· Copy the Secretscripts and all sub folders to c:\secretscripts\ on your hard drive. You need this exactly on your c drive to get these scripts to work.



· Set units to no units

· Set shadow offset to .2 this is because the Scale of your model effects the outcome of rendering in Rhinoceros. Because your units for these exercises are set to No units, Rhino assumes the model is really small. The default rendering setting for shadow offset in Rhino is set to .75. Shadow offset is how faraway the shadow is from the shadow casting object. In this tutorial because we are really pretending to work in meters .75 of a meter is too far away from the object to start the shadow of that object. Tighten up the tolerances by making the shadow offset smaller. Set it to.2



· Set rendering settings to jagged and faster.
Warning Do not use smooth and slower or custom for any of this final tutorial. You can use shadows best quality, antialiasing best quality when you want to get a better rendering.

Rotunda Entablature Script

The circular temple and rotunda go back to very ancient times. They probably are derived from circular arrangements of trees that were held sacred and marked the progression of the sun and heavens. There are numerous examples of temples and rotundas both ancient and modern everywhere in the world. These first Commandscripts create two temples based on the writings of the famous architect Vitruvius, the third Commandscript shows how you can further enrich and expand these basic rotunda and scripts into octagonal or many sided plans.

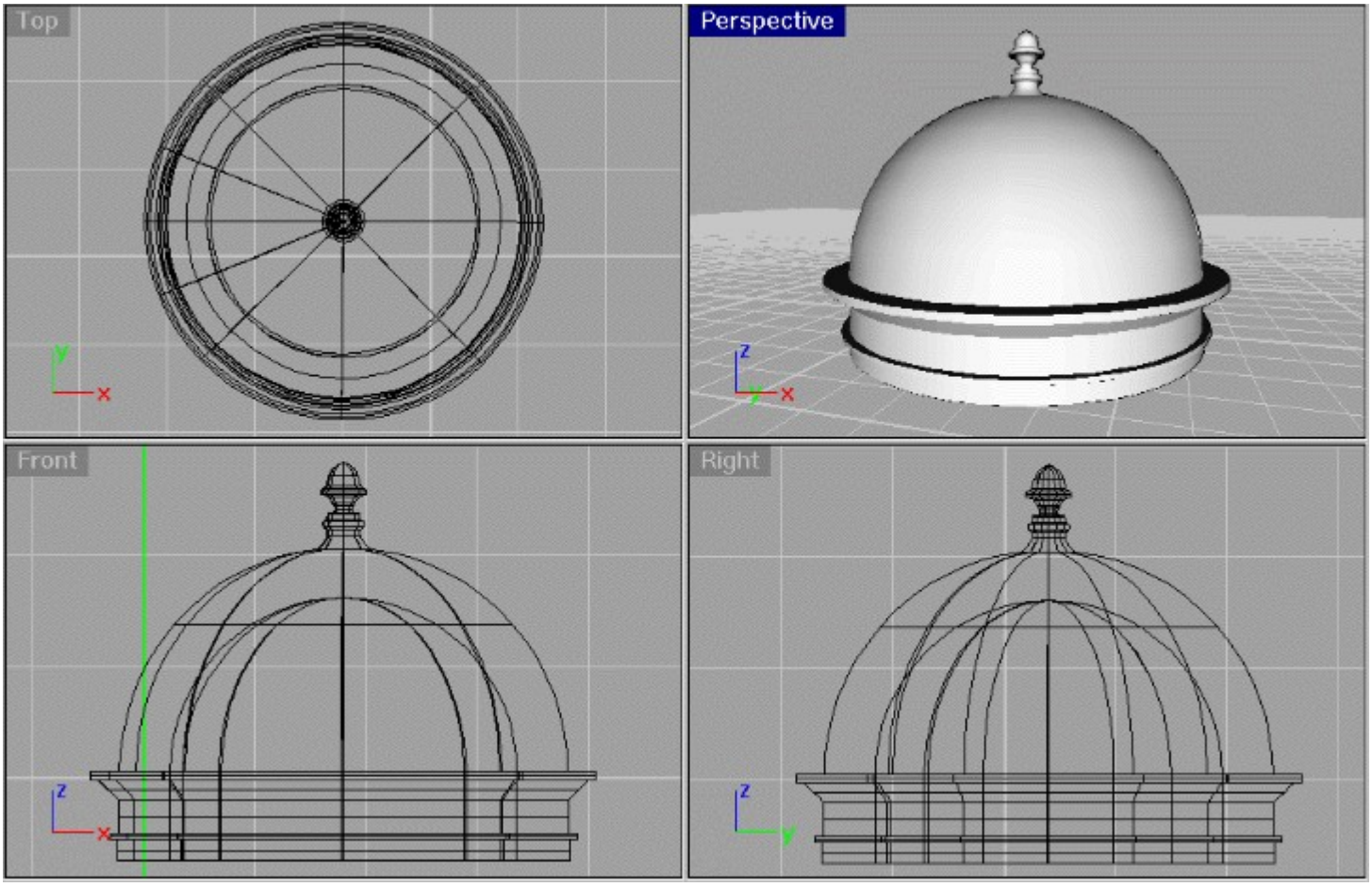
1. The first step is to create the dome and entablature of the Rotunda.
2. Open the file c:\tutorialscripts\circulartemples03.3dm
3. You will see the dome and the entablature curve profile already made for you.
4. Copy the command script from the left-hand column below.

Rotunda Entablature Script

! Circle	This creates a circle for the Entablature profile to sweep around.

w-20.9848,20.403,1.24968 r w-22.5424,20.403,1.24968 selcurve sweep1 enter ! selnone	This is the world x,y,z coordinate center point of the circle. Tells Rhino to draw the circle using radius. This is world x,y,z coordinate of the radius of the circle. This selects both curves. This sweeps the entablature around the circle to create a 3d surface. This completes the command. This de-highlights the curves that were used to create the surface.

5. Maximize the Perspective viewport in Rhinoceros
6. Click the commandpaste button. This creates the Entablature and roof.



7. Quick render the Entablature. Press escape to exit quick render
9. Click undo once. This undoes the entablature creation. Click undo again. This undoes the circle that the curves were rail swept around.

You should now be back to only the original curve profile.

Cplanes and Commandscripting using the Entablature profile as an example.

There are two ways to enter coordinate points in Rhino. One is in world coordinates and one is in cplane coordinates. I have been using world coordinates mainly because this makes a command script work predictable under almost all circumstances.

You can move and designate the cplane coordinates in a script. Than you will be able to use cplane coordinates that act predictably under almost all circumstances too. This is of some help but mostly you can stick with world coordinates for scripts that you want to be universal. Otherwise there is no way of knowing where a fellow user has their cplane origin at and thus your objects or scripted procedures will end up in the wrong places and probably will not work. You can specify a cplane origin but than by that time it is also easy to use world coordinates. So why use the cplane in scripts? The biggest reason is that you can make many of your scripts user interactive and more universal for yourself to use.

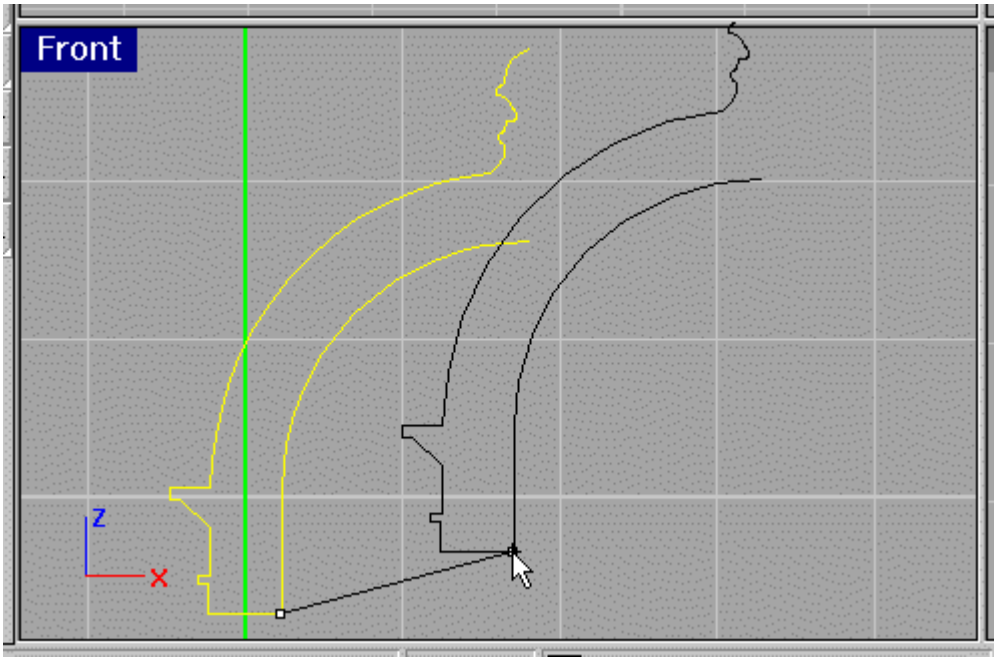
! Circle w-20.9848,20.403,1.24968	This creates a circle for the Entablature profile to sweep around. This is the world x,y,z coordinate center point of the circle.

Here is what I mean. Take the script above, notice where the circle and thus the Entablature profile is located. It is specified in world coordinates. This is ok when you want to share a script or use that same location. It is also not too hard to change the locations' coordinates if you relocate the Entablature profile somewhere else in 3d space, but it is bit of hassle. Here is where using the cplane coordinates instead of world coordinates comes to the rescue.

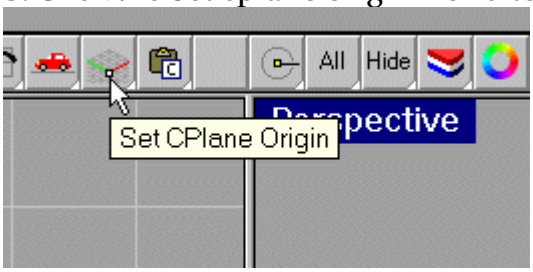
If you remove the world location and enter the radius of the circle which is the first location point and 0 the commandscript now reads...

! Circle 1.55756,0 r 1.55756 selcurve sweep1 enter ! selnone	This creates a circle for the Entablature profile to sweep around. This is the cplane x,y,z coordinate center point of the circle. Tells Rhino to draw the circle using radius. This is cplane x,y,z coordinate of the radius of the circle. This selects both curves. This sweeps the entablature around the circle to create a 3d surface. This completes the command. This de-highlights the curves that were used to create the surface.

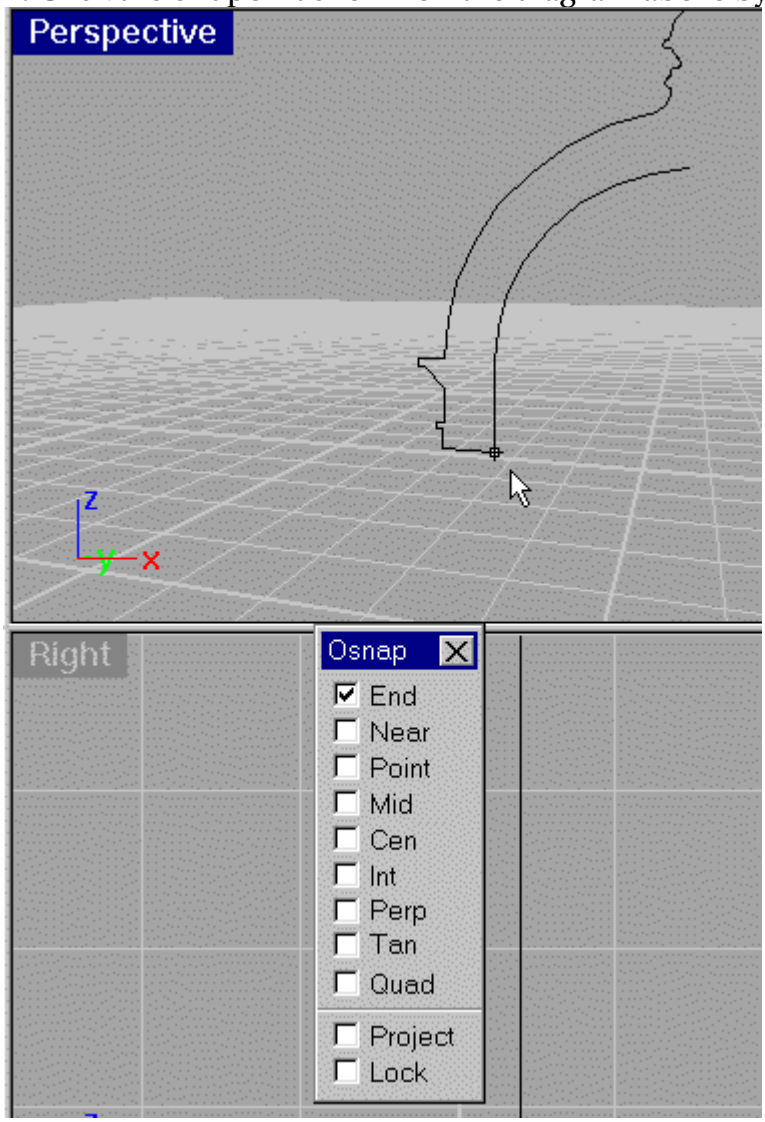
1. Highlight and copy the left-hand column of script above
2. Back in Rhino move the Entablature anywhere in space



3. Click the Set cplane origin world top icon



4. Click the endpoint shown on the diagram above by using snap to endpoint.



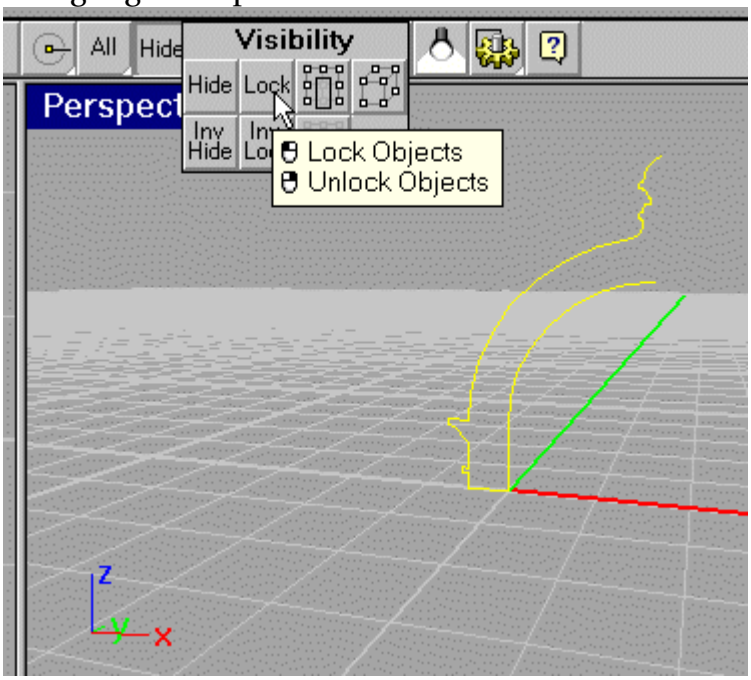
5. Make sure the perspective viewport is active and click the Run Commandscript icon

This is how the Cplane is useful it lets the you the user choose where command scripts take place. By you picking the origin so you don't have to cut and paste world coordinates all the time. This is great for one user only remember scripts that you want to share with others you should be careful with user input of the cplane, you really can't trust things to run smoothly.

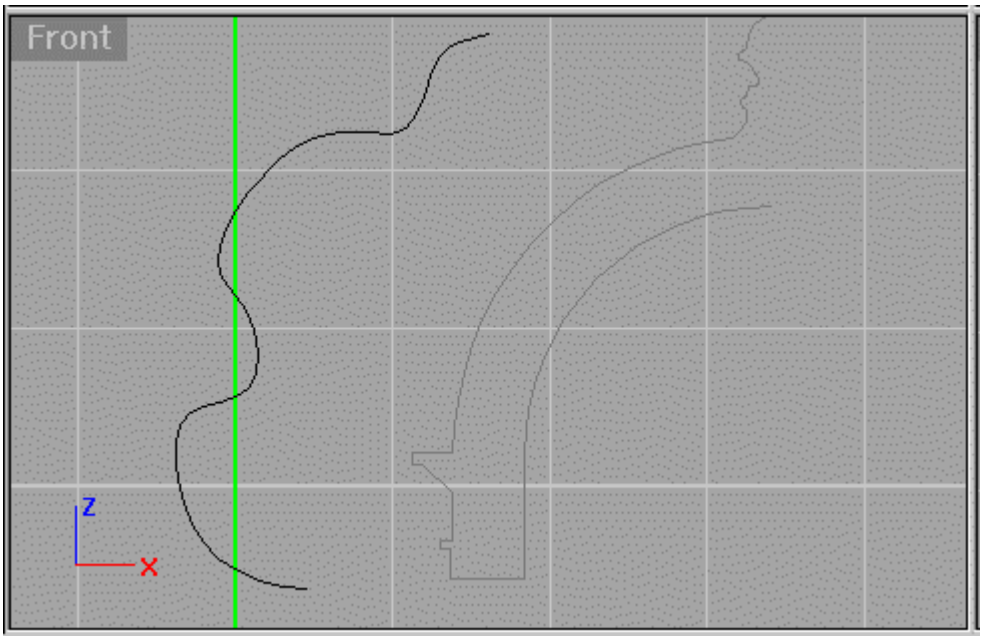
But as you see these are extremely useful for your everyday modeling routines.

Drawing a free form Entablature using the same script

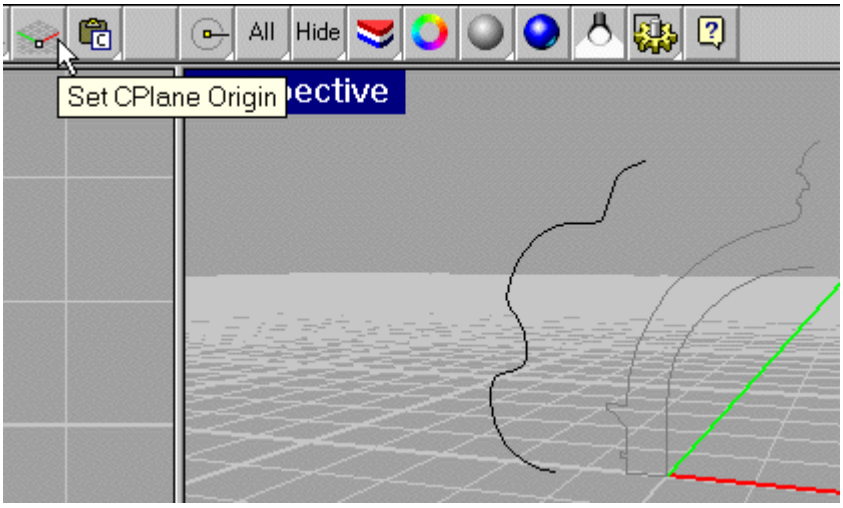
1. Highlight the provided current curve and Lock it...



2. Draw a freeform curve profile similar to the one shown in the illustration below.



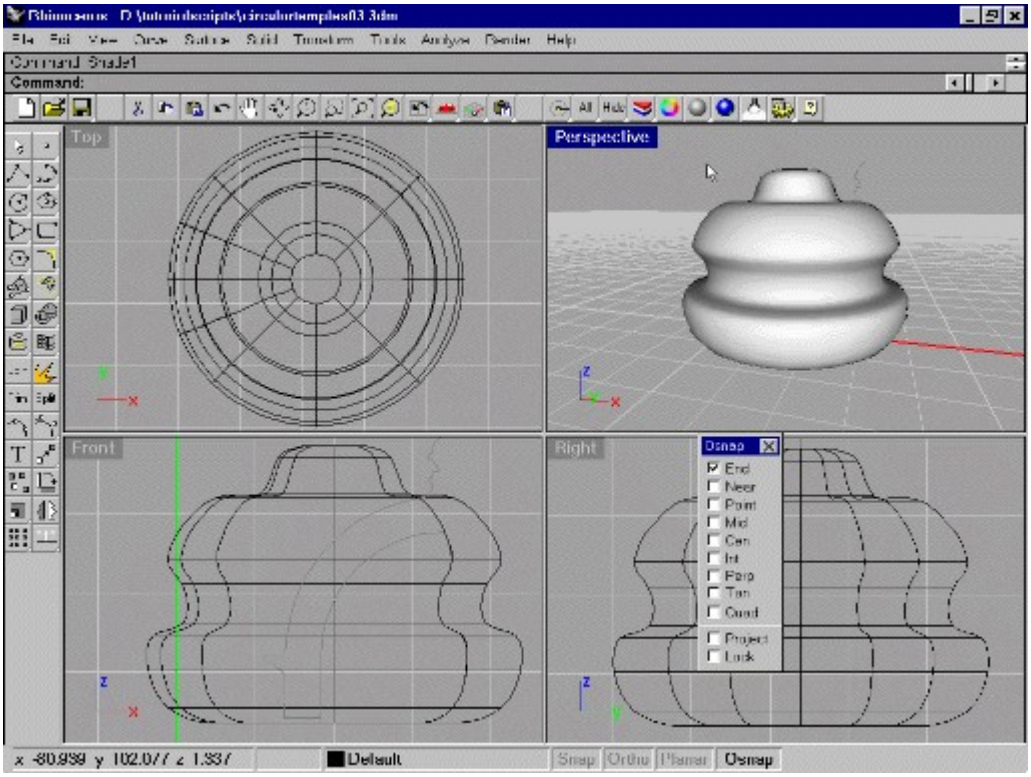
3. Click cplane origin



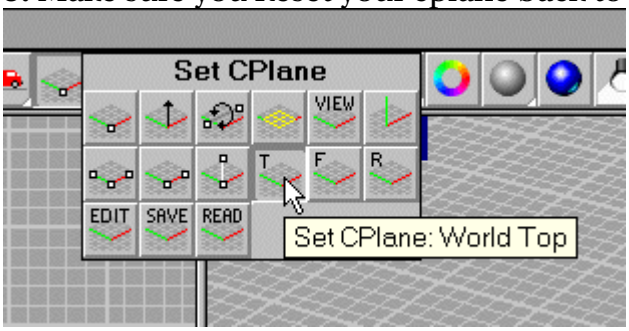
4. Using snap to endpoint click the endpoint on the curve to set the cplane
5. Copy the the Commandscript below...

! Circle
1.55756,0
r
1.55756
selcurve
sweep1
enter

6. Highlight the perspective view
7. Run the commandscript by clicking the run command script icon. You end up with a surface similar to the figure below



8. Select all and delete the freeform shape and its curve
9. Make sure you Reset your cplane back to world origin top



10. Unlock the provided curve profile

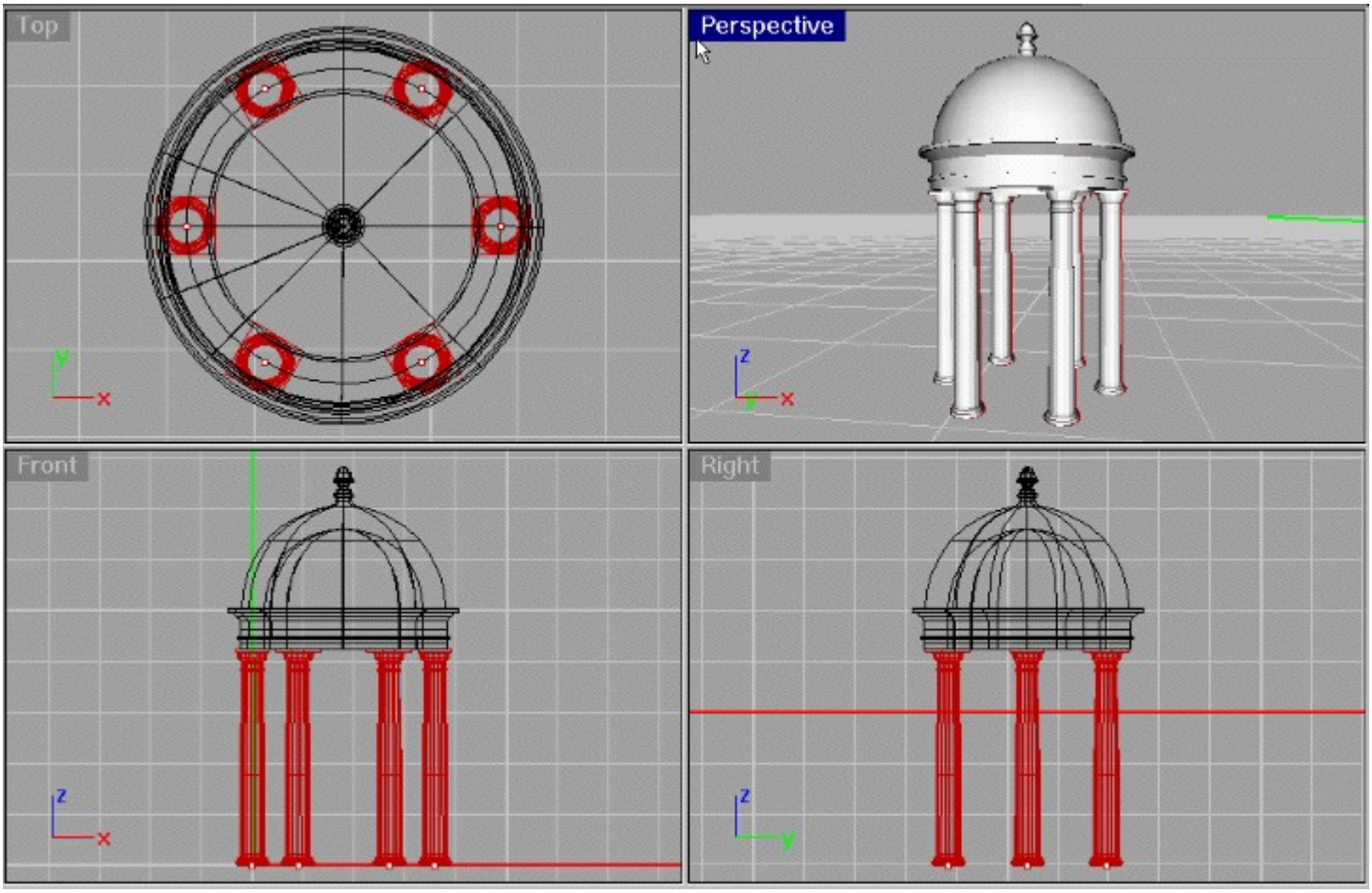
Adding a prebuilt column from a file to the Entablature of the rotunda.

I have created a column for you that is a fairly accurate historical example. This column is already converted to a polygon mesh file. Once you are satisfied that the component is good save it as an optimized mesh file if it is a large nurbs file like a classical column.

1. Copy and paste the left hand column below

! Circle	This creates a circle for the Entablature profile to sweep around.
w-20.9848,20.403,1.24968	This is the cplane x,y,z coordinate center point of the circle.
r	Tells Rhino to draw the circle using radius.
w-22.5424,20.403,1.24968	This is cplane x,y,z coordinate of the radius of the circle.
selcurve	This selects both curves.
sweep1	This sweeps the entablature around the circle
enter	This completes the command.
! merge	This command begins the file merge command.
c:\tutorialscripts\columns\tuscancol01.3dm	This tells rhino what file to merge into the present one.
move	This moves that recently imported or merged file.
w0	The place to move the part from in this case world 0,0,0.
w-22.779,20.403,-3.01278	This moves the part to these world coordinates.
! selmesh	This selects all meshobjects, an easy way to select the columns.
ArrayPolar	This command performs a polar array.
w-20.9848,20.403	This is the origin point for beginning the polar array.
6	This tells how many items or copies to make in the array.
360	This tells the angle in which to distribute the columns.
enter	This finishes the command.
selnone	This deselects all selected objects.

2. In Rhinoceros click the command paste Icon



3. Quick render the model
4. Select the orignal provided curve profile
5. click the Invert Selection icon or type ! Invert at the Rhinoceros command prompt.
6. Delete the inverted selection. You should only have the the orignal provided curve profile in the workspace.

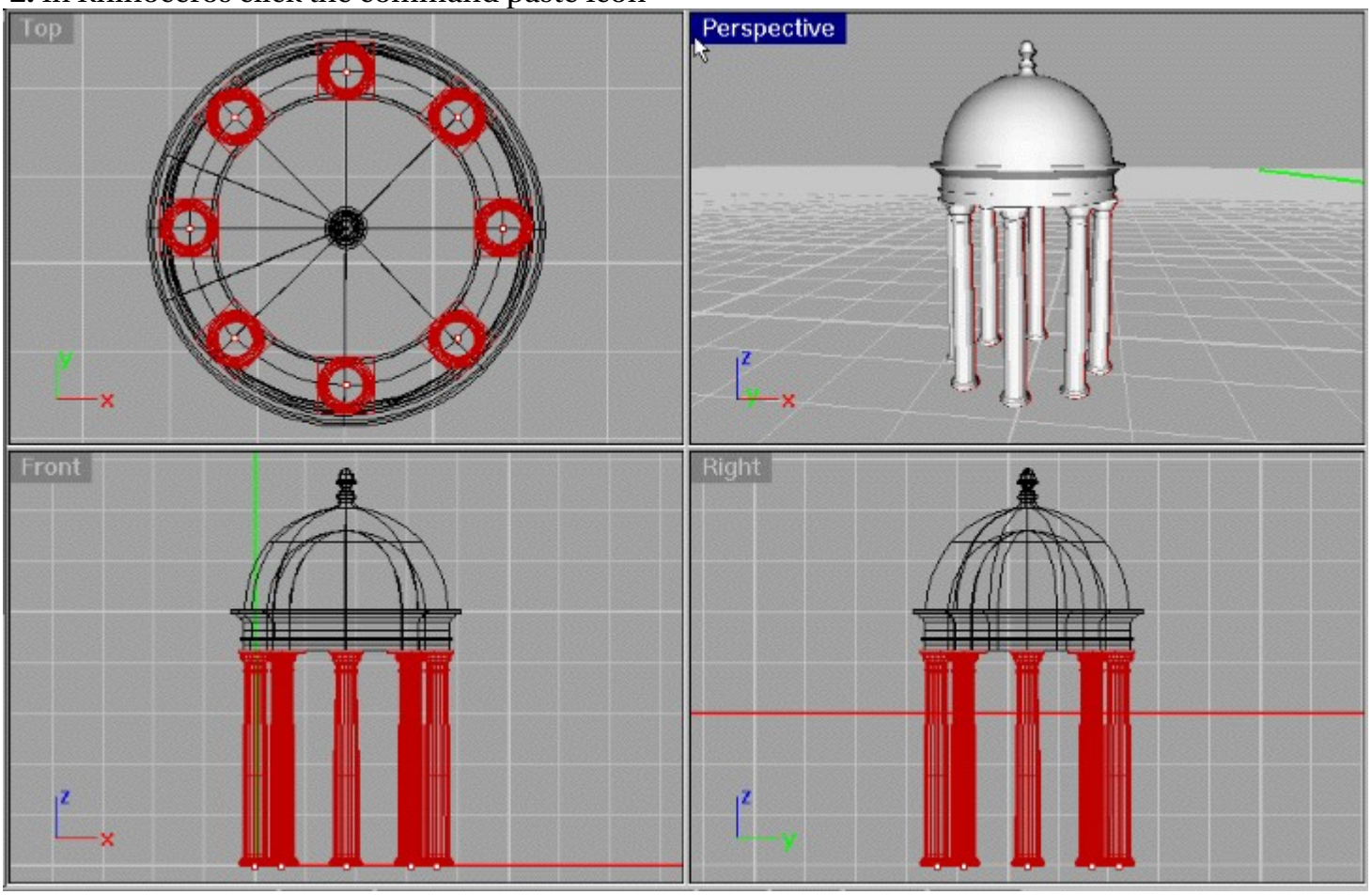
How to easily change the amount of columns
Now make eight columns instead of six. You would change the input to read **8** instead of 6 at line X

1. Copy and paste the left hand column below

! Circle w-20.9848,20.403,1.24968 r w-22.5424,20.403,1.24968 selcurve sweep1 enter ! merge c:\tutorialscripts\columns\tuscancol01.3dm move w0	This creates a circle for the Entablature profile to sweep around. This is the cplane x,y,z coordinate center point of the circle. Tells Rhino to draw the circle using radius. This is cplane x,y,z coordinate of the radius of the circle. This selects both curves. This sweeps the entablature around the circle This completes the command. This command begins the file merge command. This tells rhino what file to merge into the present

w-22.779,20.403,-3.01278 ! selmesh ArrayPolar w-20.9848,20.403 **8** 360 enter selnone	one. This moves that recently imported or merged file. The place to move the part from in this case world 0,0,0. This moves the part to these world coordinates. This selects all meshobjects, an easy way to select the columns. This command performs a polar array. This is the origin point for beginning the polar array. X This tells how many items or copies to make in the array. This tells the angle in which to distribute the columns. This finishes the command. This deselects all selected objects.

2. In Rhinoceros click the command paste Icon



3. Quick render the model
4. Select the orignal provided curve profile
5. click the Invert Selection icon or type ! Invert at the Rhinoceros command prompt.
6. Delete the inverted selection. You should only have the the orignal provided curve profile in the workspace.

How to easily change and replace components with more detailed ones.

Now you will reference more detailed models and profile curves to make a more true to life and complex rotunda.

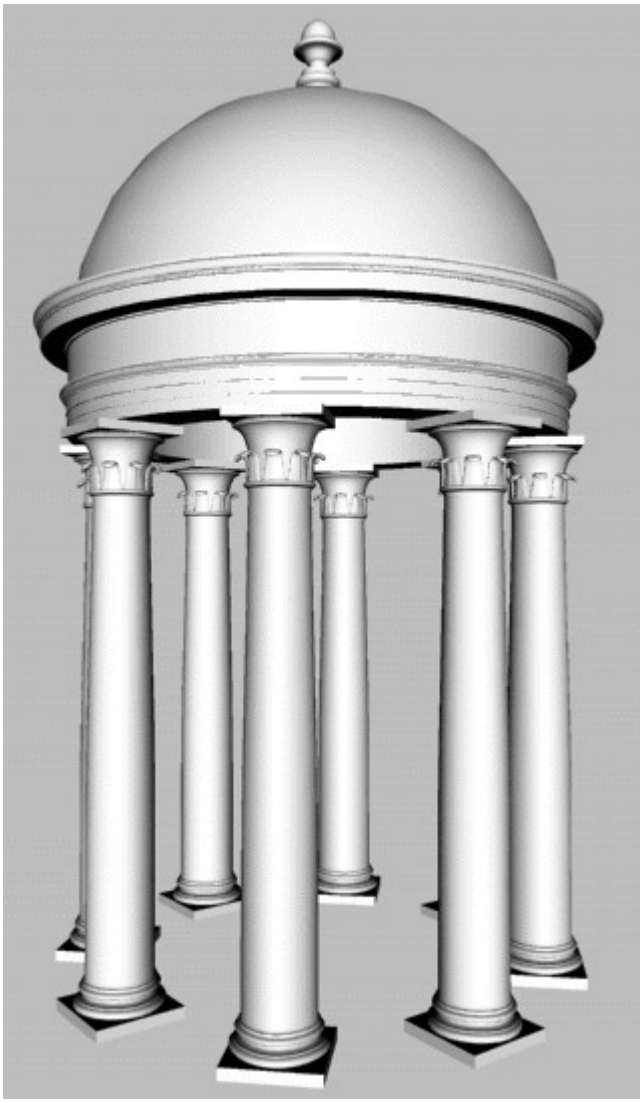
A warning: Cornices or any shapes that create long and thin polygons for rendering will really bog down doing anything in Rhino. For instance the entablature curve profile I have provided you creates a very complex surface for Rhino's mesher. So always keep your rendering settings on ** until the final render. The other provided column is based on the tower of the four winds at Athens in proportion. It is more detailed and only missing the top ornaments. The entablature profile is created from one of the many specifications of Asher Bejamins classical architecture. But this matches quite well in proportion and style with the column from the four winds.

1. Copy and paste the left hand column below

! Merge c:\tutorialscripts\Entablature\entabprofile03.3dm	This command begins the file merge command. This tells rhino what file to merge into the present one.
! Circle w-20.9848,20.403,1.24968 r w-22.5424,20.403,1.24968	This creates a circle for the Entablature profile to sweep around. This is the cplane x,y,z coordinate center point of the circle. Tells Rhino to draw the circle using radius. This is cplane x,y,z coordinate of the radius of the circle.
selcurve sweep1 enter	This selects both curves. This sweeps the entablature around the circle This completes the command.
! merge c:\tutorialscripts\columns\Corinthmesh.3dm	This command begins the file merge command. This tells rhino what file to merge into the present one.
move w0 w-22.779,20.403,-3.01278	This moves that recently imported or merged file. The place to move the part from in this case world 0,0,0. This moves the part to these world coordinates.
! selmesh ArrayPolar w-20.9848,20.403 8 360 enter selnone	This selects all meshobjects, an easy way to select the columns. This command performs a polar array. This is the origin point for beginning the polar array. This tells how many items or copies to make in the array. This tells the angle in which to distribute the columns. This finishes the command. This deselects all selected objects.

2. In Rhinoceros click the command paste Icon

3. Quick or full render the model



4. Select the orignal provided curve profile
5. click the Invert Selection icon or type ! Invert at the Rhinoceros command prompt.
6. Delete the inverted selection. You should only have the the original provided curve profile in the workspace.

Using other Rhino modeling commands to create different structures.

Now instead of using sweep one rail you will create an octagonal pavilion using Polygon and Rail path.
This command script creates an octagonal temple.

Important note Here you have to finish the command by completing the dialogbox. Commands like Loft, revolve, need user input to complete. In the future I am sure that Rhino will make those commands scriptable. You can still use them but if you share them the user must be notified in some way that they have to perform some input at the end of the script.

! Merge
c:\tutorialscripts\Entablature\entabprofile03.3dm
! Polygon

N
8
-20.9848,20.403,1.24968
-22.5424,20.403,1.24968
! enter
RailRevolve

1. Copy the above text in the left-hand column.
2. In Rhino highlight the perspective viewport
3. Click the command paste icon and finish the input. This is where you have to choose first the profile curve than the radius curve.
4. Now copy the text in the left-hand column below
5. Click the Command paste icon.

! merge
c:\tutorialscripts\columns\Corinthmesh.3dm
move
w0
w-22.779,20.403,-3.01278
! selmesh
ArrayPolar
w-20.9848,20.403
8
360
enter
selnone

6. Quick or full render the model

7. Close the file. When prompted to save enter no.

Here you see that by using different simple modeling commands you can build different structures with your components. Notice how easy it is to evolve parts of structures, whole structures and play out different scenarios using command scripting in Rhino.

There are many round or polygonal structures you could emulate or create by using variations of this script.
If you live in the United Kingdom there are many wonderful examples of Rotundas in British garden parks that could easily be scripted by modifying the Rotunda commandscript. Some wonerful Rotundas and Circular temples are at...
Stowe
St Paul's Walden Bury
Anglesey Abbey

Even if you don't have detailed models you can create them in the proper proportions in block or lesser form first than using commandscripting reference more detailed models latter.

Modeling the Oecus

The Oecus is the formal dining room that faces the Peristyle garden and was richly decorated. It was used mainly for special occasions and for dinning with quests. You can really get a sense of the lifestyle the Pompeians had by this facet of their architecture. Imagine banquets looking out onto the garden from your couch through magnificent architectural columns. The walls painted with scenes and illusionistic architecture.

Here you will learn important command scripting skills that will augment what you know because it shows how to locate, place and draw objects in more depth. The most important part of the Oecus is wall facing the courtyard.

There are the 3 main ways to script the wall model:

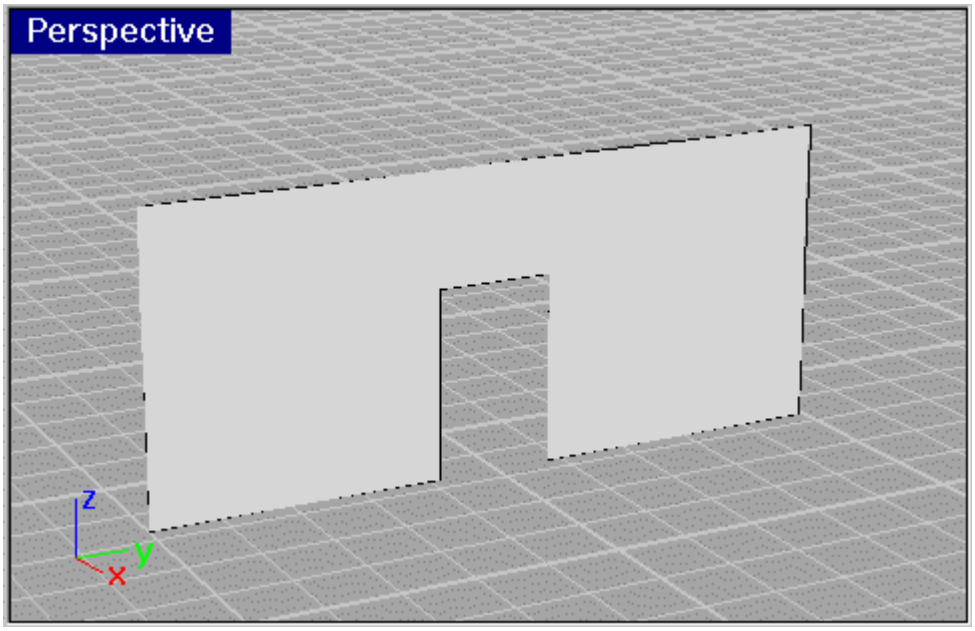
1. The first technique creates the entire wall using world coordinates.
2. The second technique uses cplane coordinates.
3. The third technique is file merged based.

This technique is world coordinate based:
You can build the complete model by using coordinates this is similar to the block manifold technique

1. Copy the entire contents below

```
! polyline
w-13.319,-4.96942,0
w-13.319,-0.823458,0
w-13.319,-0.823458,2.66477
w-13.319,0.823458,2.66477
w-13.319,0.823458,0
w-13.319,4.96942,0
w-13.319,4.96942,4.26246
w-13.319,-4.96942,4.26246
w-13.319,-4.96942,0
! selcurve
! PlanarSrf
! selnone
```

2. In Rhinoceros maximize the perspective window.
3. Click the commandpaste icon
4. The script created a simple wall with a door cutout.



5. Select all and delete the wall.

This technique is Cplane or view/object coordinate based.

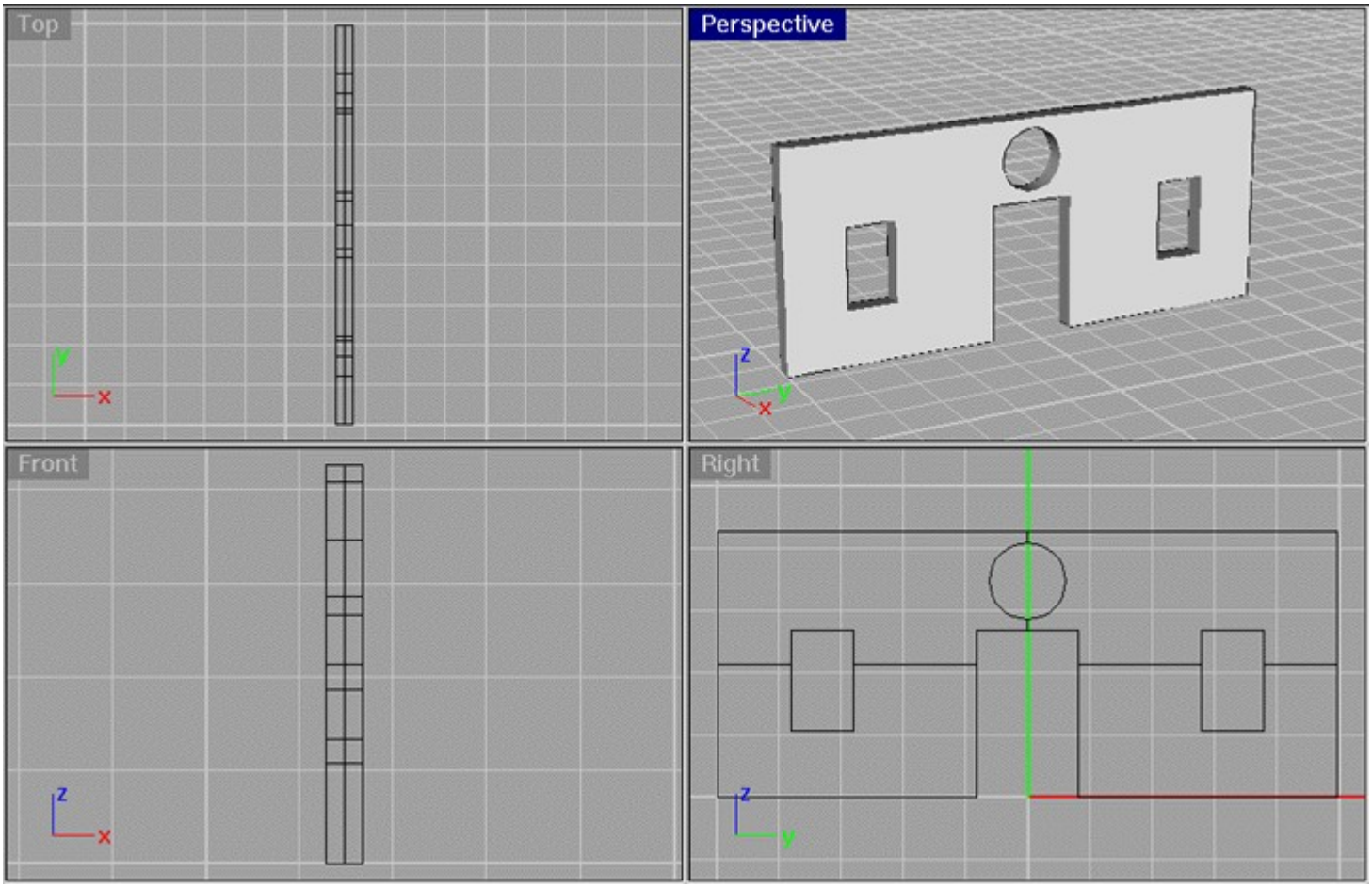
I discussed the benefits of the cplane earlier and here you can examine its use in more detail. You can see a further benefit is that it also creates the wall but lets you enter and track the wall dimensions easier. This is a good technique if you need to change the sizes of your model or component often.

1. Copy the script in the left hand column below.

! Cplanetop	Sets the cplane to the top ortho view w 0,0,0.
! CplaneOrigin	Sets the origin of the cplane.
w-13.319,-4.96942,0	This is the world coordinate of cplane 0,0,0.
! polyline	Draws a polyline that represents the wall outline.
0,0,0	Beginning coordinate points for the polyline or wall.
0,4.14596,0	Further coordinate points for the polyline or wall.
0,4.14596,2.66477	Further coordinate points for the polyline or wall.
0,5.79288,2.66477	Further coordinate points for the polyline or wall.
0,5.79288,0	Further coordinate points for the polyline or wall.
0,9.93884,0	Further coordinate points for the polyline or wall.
0,9.93884,4.26246	Further coordinate points for the polyline or wall.
0,0,4.26246	Further coordinate points for the polyline or wall.
0,0,0	Last coordinate points for the polyline or wall.
! enter	Finishes drawing the polyline.
! CplaneRight	Sets the cplane to Right world ortho view.
! CplaneOrigin	Sets the cplane origin of that view.
w-13.319,-4.96942,0	World Coordinate point for the cplane orign.
! Rectangle	Draws a rectangle or the outline of the windows.
0,0,0	Beginning coordinates for the rectangle.
0.98743,1.5977	Ending coordinates for the rectangle.
Lastcreated	This selects the last created item.
move	This moves the rectangle into place

0,0,0	Moves the rectangle from this coordinate point
1.1904,1.06707,0	Points to finish the move
! Rectangle	Draws a rectangle or the outline of the other window.
0,0,0	Beginning coordinates for the rectangle.
0.98743,1.5977	Ending coordinates for the rectangle.
Lastcreated	This selects the last created item.
move	This moves the rectangle into place
0,0,0	Moves the rectangle from this coordinate point
7.76101,1.06707,0	Points to finish the move
! Circle	Draws a circle outline for the circular window.
0,0,0	Beginning coordinates for the.
radius	Tells rhino to draw the circle using it's Radius.
0.61212	Ending coordinates for the circle.
enter	Ends the circle command.
Lastcreated	This selects the last created item.
move	This moves the circle into place.
0,0,0	Moves the circle from this coordinate point.
4.96901,3.46361,0	Points to finish the move.
! selnone	Deselects all items.
! Cplanetop	Sets the cplane back to world othro top.
! CplaneOrigin	Sets the origin
w-13.319,-4.96942,0	To these coordinates
! Selcurve	Selects all the curves to perform an extrude command
! Extrude	Extrudes the curves
Cap=Yes	Cap = yes option tells rhino to create a closed polysurface so the
-0.4	This is the thickness of the wall
delete	This deletes the curves used to create the wall
! enter	Completes the command
! CPlaneTop	Re-sets cplane to world ortho top

2. Make sure perspective viewport is active.
3. Run the script by clicking the commandpaste icon. The script created a wall with window and door cutouts.



4. Quick Render the model
5. Select all and delete

This technique is File merged based.

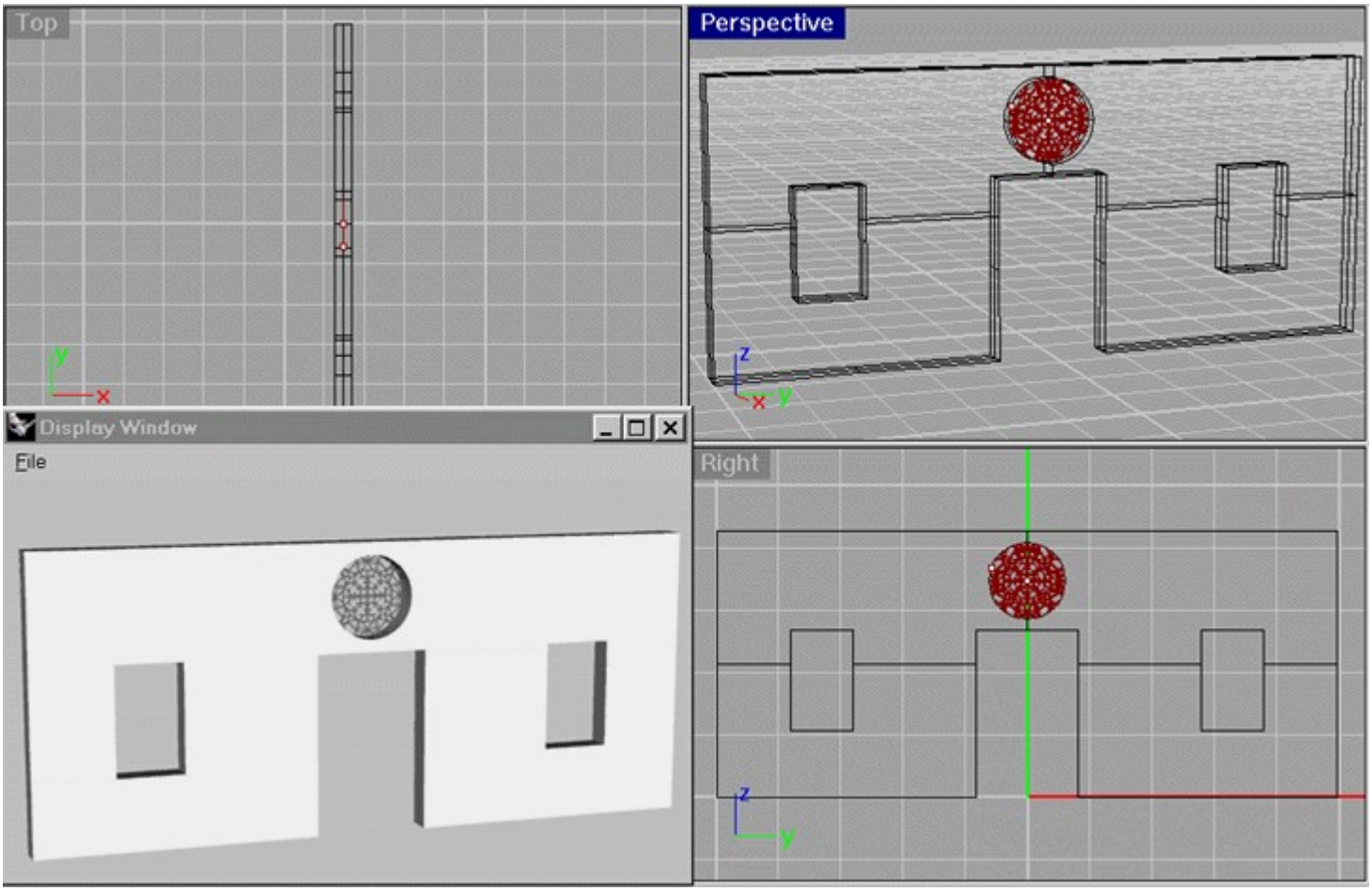
This script also builds the wall with cutouts and places a rose window into the round cutout. This is the script you would use as your final script.

1. Copy the script in the column below.

```
! all
! Hide
! merge
c:\tutorialscripts\plans\Oecusfacade.3dm
! Selcurve
! Extrude
Cap=Yes
-0.4
delete
! enter
! merge
c:\tutorialscripts\elements\rosewindow.3dm
Lastcreated
move
```

```
w0,0,0
w-13.519,-0.00041,3.46361
selnone
show
```

2. Make sure perspective viewport is active.
3. Run the script by clicking the commandpaste icon.
The script created a wall with window and door cutouts and placed a Rose window grille in the round cutout.



4. Quick Render the model
5. Select all and delete. Exit the drawing when prompted to save enter no.

The complete Oecus file

I have already created the complete Oecus file for you. For your study you can examine the rest of the Oecus room script below. You don't need to command paste this it is only for study.

```
! merge
c:\tutorialscripts\plans\Oecusfacade.3dm
! Selcurve
! Extrude
Cap=Yes
```

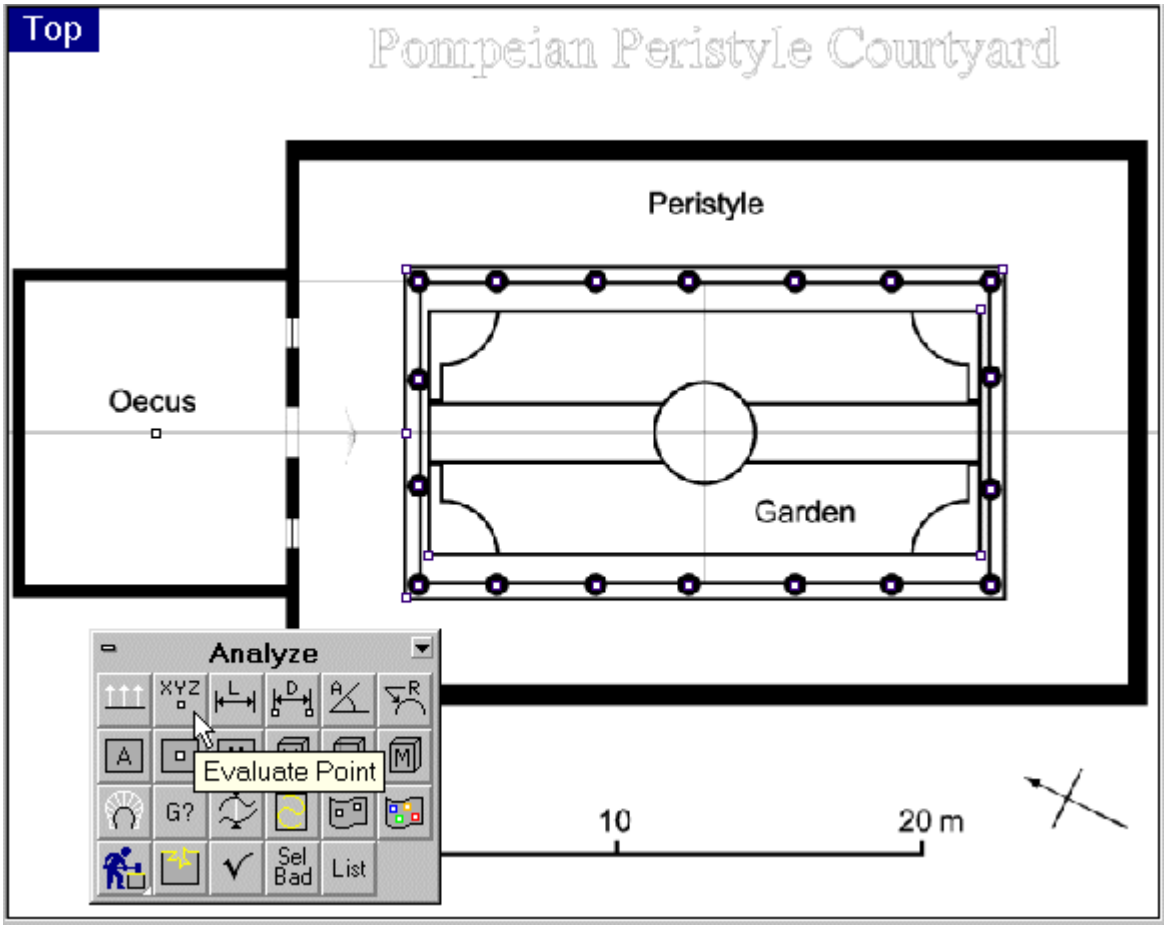
```
-0.4
delete
! merge
c:\tutorialscripts\elements\rosewindow.3dm
Lastcreated
move
w0,0,0
w-13.519,-0.00041,3.46361
! all
! Hide
! polyline
w-13.719,-4.96942,0
w-22.2691,-4.96942,0
w-22.2691,4.96942,0
w-13.719,4.96942,0
enter
Lastcreated
! Extrude
4.26246
! Plane w-13.319,-4.96942,0 w-22.2691,4.96942,0
lastcreated
copy
inplace
move
w-13.319,-4.96942,0 r0,0,4.26246
all
Hide
! merge
c:\tutorialscripts\plans\Oecusfloorplan.3dm
SelCrv
! Extrude
4.26246
SelSrf
copy
inplace
move
w-13.319,-4.96942,0 r0,0,4.26246
! enter
! selnone
! show
```

Creating the Column placement script

The next part to script is the column placement.

This is easy because the column height equals the ceiling height you only need to find the points where the columns would be placed. I did this by placing points in the center of the columns on the plan then using the evaluate point command I cut and pasted the necessary

coordinate points into the script. I only needed half the coordinates because the other half is a mirror copy. You used this method in the Block manifold tutorial in chapter 2. See image below.



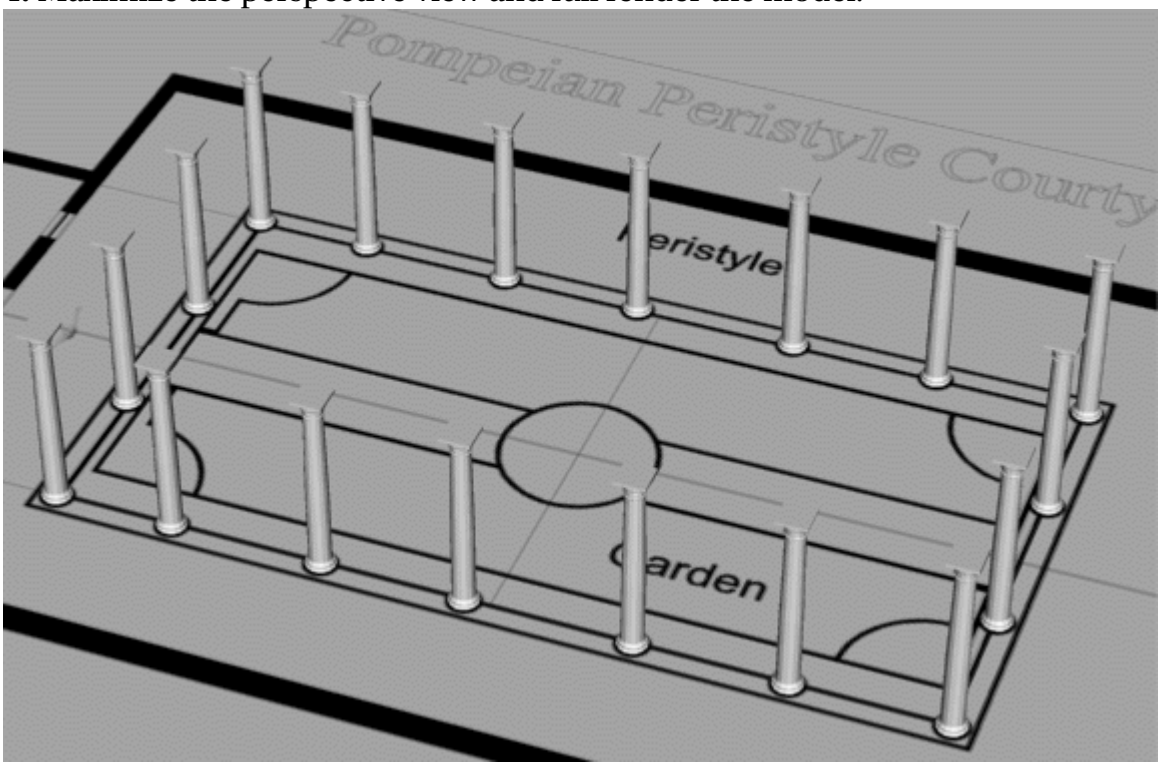
Here is what the column placement script looks like you will reference a less detailed column first than latter you will reference the more detailed column.

1. Copy the script in the column below.

```
all
! Hide
! merge
c:\tutorialscripts\columns\tuscancol01.3dm
copy
w0
-9.37803,-4.96942,0
-9.37803,-1.73372,0
-6.82304,-4.96942,0
-3.57283,-4.96942,0
-0.522624,-4.96942,0
2.97761,-4.96942,0
6.12782,-4.96942,0
9.37803,-4.96942,0
9.37803,-1.84309,0
! delete
```

```
! SelMesh
! Mirror
0,0,0
12,0,0
enter
! selnone
! show
```

2. Make sure perspective viewport is active.
3. Run the script by clicking the commandpaste icon.The script places columns at the correct locations.
4. Maximize the perspective view and full render the model.



5. Select all and delete.

Creating the Entablature and Roof Script

The next step is create the entablature and the part of the Peristyle roof that is visible

There is a potential problem and one that had me stumped for a while. As I mentioned surfaces created from a complex aggregate curve profiles like the detailed entablature will bog down computer memory and create huge files that are almost impossible to render in Rhino or any other program.

- One trick is to just use planes and texture map them. This is a decent but not great solution since you loose the outlines and shadows of a true 3d model.
- The other solution is to use the complex curve profile and then use only "render jagged and faster". This is actually ok but is not good for export unless the program you are exporting to supports trimmed nurbs.
- The third solution is to mesh the nurbs model to a polygon mesh. This is good a

solution because rendering is fast and it makes updating and exporting files fast and seamless. Do this last because it represents a finished part of the model.

I use all three methods. This is because command scripting makes it easy and you get the benefit of all worlds depending on your particular needs.

Reminder: If you want to add textures to your polygon mesh files for rendering in Rhino you need to add them to the nurbs model before you mesh the model. This is because Rhino will then establish UV coordinates for the bitmaps that will translate into the polygon mesh. But if you mesh a nurbs object without any UV or a bitmap assigned to it there won't be any UV info to transfer. So if you want to texture your mesh objects for rendering in Rhino remember to texture map them when they are nurbs objects before you polygon mesh them.

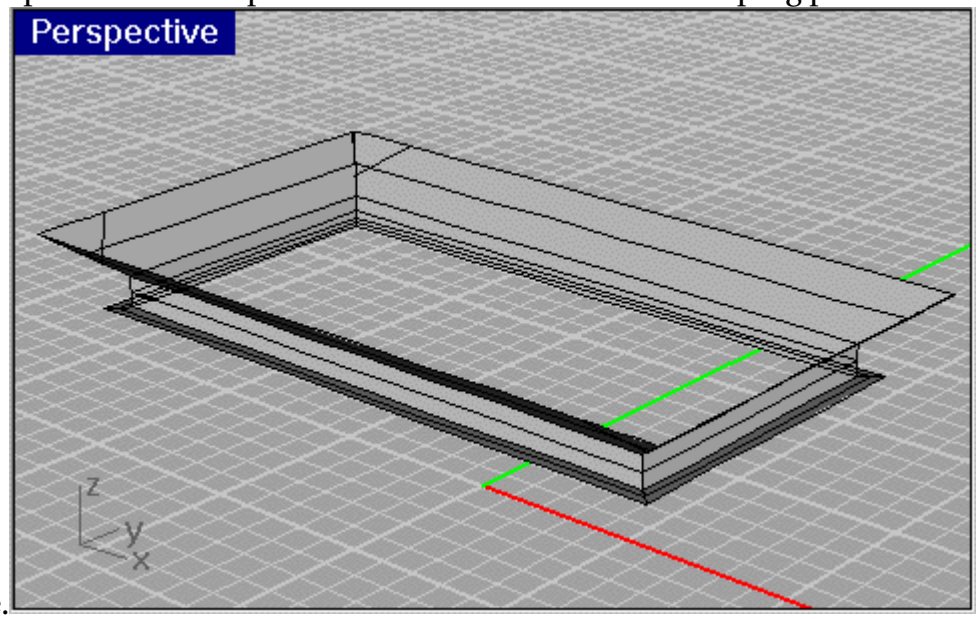
Notice the entablature is really a picture frame molding in an abstract sense. In this case we can use shear first to find the correct in-between curve profile and than use the modeling commands Loft or for scripting sweep one Rail.

First the simple plane method. This is really good for fast visualization

1. Copy the script in the left hand column below.
2. Make sure perspective viewport is active.
3. Run the script by clicking the commandpaste icon.

```
! merge
c:\tutorialscripts\entablature\entabprofile04.3dm
selcurve
sweep1
enter
selcurve
delete
selnone
```

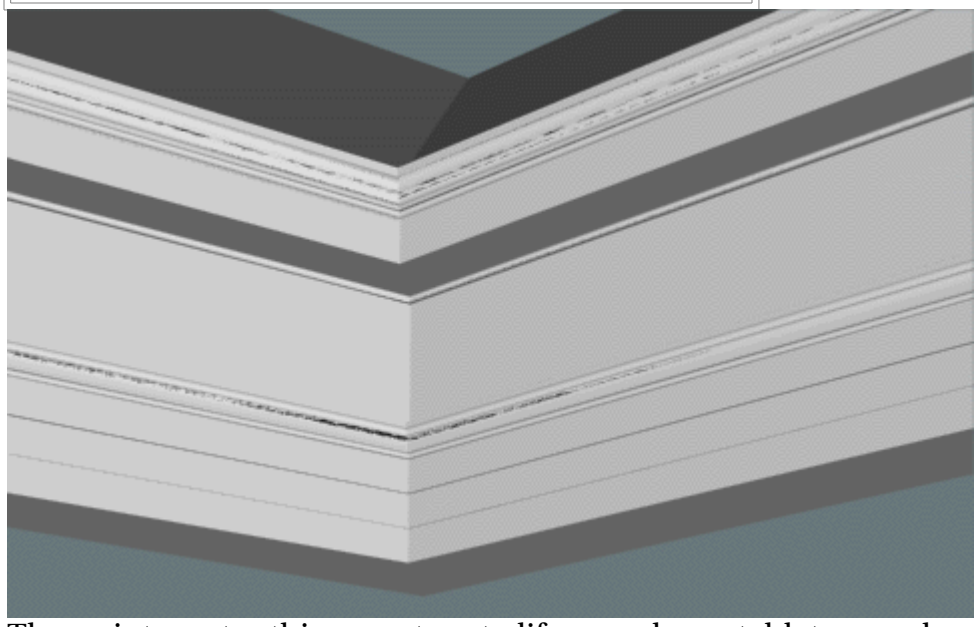
The script creates a simplified flat Entablature and the sloping parts of the Roof that are visible.



Create a more complex entablature by referencing the more complex curve profile. Import the entabprofile01.3dm file

1. Copy the script in the left hand column below.
2. Make sure perspective viewport is active.
3. Run the script by clicking the commandpaste icon.

```
! merge
c:\tutorialscripts\entablature\entabprofile05.3dm
selcurve
sweep1
enter
selcurve
delete
selnone
```



The script creates this more true to life complex entablature and roof. Select all and delete

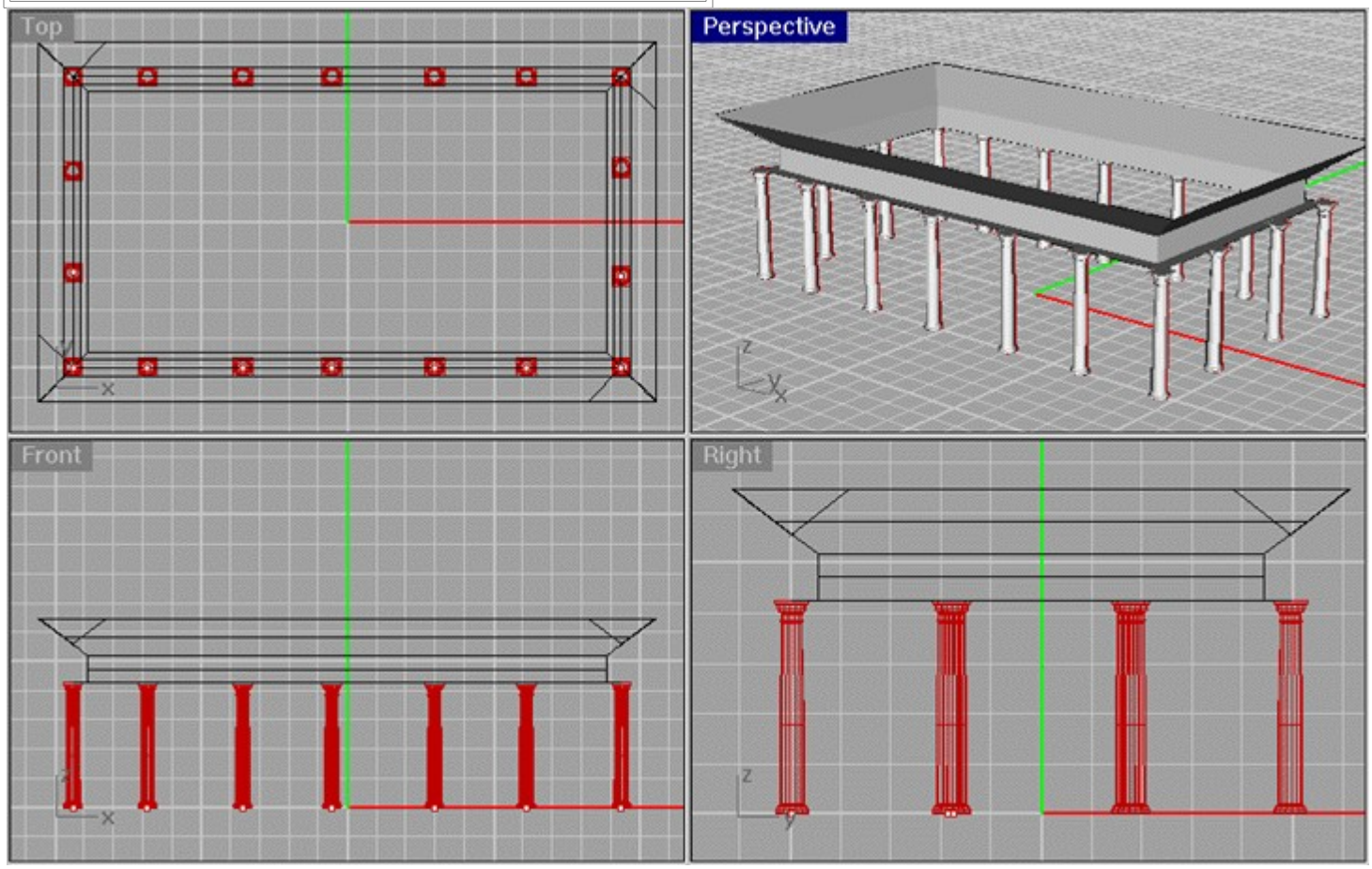
Entablature and Roof plus Tuscan Columns Script

Now add the simplified columns to the simplified entablature

1. Copy the script in the left hand column below.
2. Make sure perspective viewport is active.
3. Run the script by clicking the commandpaste icon.
4. Quick and full render the model if you wish.
5. Select all and delete to clear the workspace for the next step.

```
! merge
c:\tutorialscripts\entablature\entabprofile04.3dm
selcurve
sweep1
enter
selcurve
delete
selnone
```

```
all
! Hide
! merge
c:\tutorialscripts\columns\tuscancol01.3dm
copy
w0
-9.37803,-4.96942,0
-9.37803,-1.73372,0
-6.82304,-4.96942,0
-3.57283,-4.96942,0
-0.522624,-4.96942,0
2.97761,-4.96942,0
6.12782,-4.96942,0
9.37803,-4.96942,0
9.37803,-1.84309,0
! delete
! SelMesh
! Mirror
0,0,0
12,0,0
enter
! selnone
! show
```



Complete Entablature and Roof Script with Detailed Columns

The final step would be to mesh the complex Nurbs Entablature and roof and combine it with the more detailed columns. I have already done this for you.

1. Copy the script in the left hand column below.
2. Make sure perspective viewport is active.
3. Run the script by clicking the commandpaste icon.
4. Quick and full render the model if you wish.
5. Select all and delete to clear the workspace for the next step.

```
! merge
c:\tutorialscripts\entablature\vettientab03.3dm
Selnone
! all
! Hide
! merge
c:\tutorialscripts\columns\Corinthmesh.3dm
copy
w0
-9.37803,-4.96942,0
-9.37803,-1.73372,0
-6.82304,-4.96942,0
-3.57283,-4.96942,0
-0.522624,-4.96942,0
2.97761,-4.96942,0
6.12782,-4.96942,0
9.37803,-4.96942,0
9.37803,-1.84309,0
! delete
! SelMesh
! Mirror
0,0,0
12,0,0
enter
! selnone
! show
```

Test the various scenarios

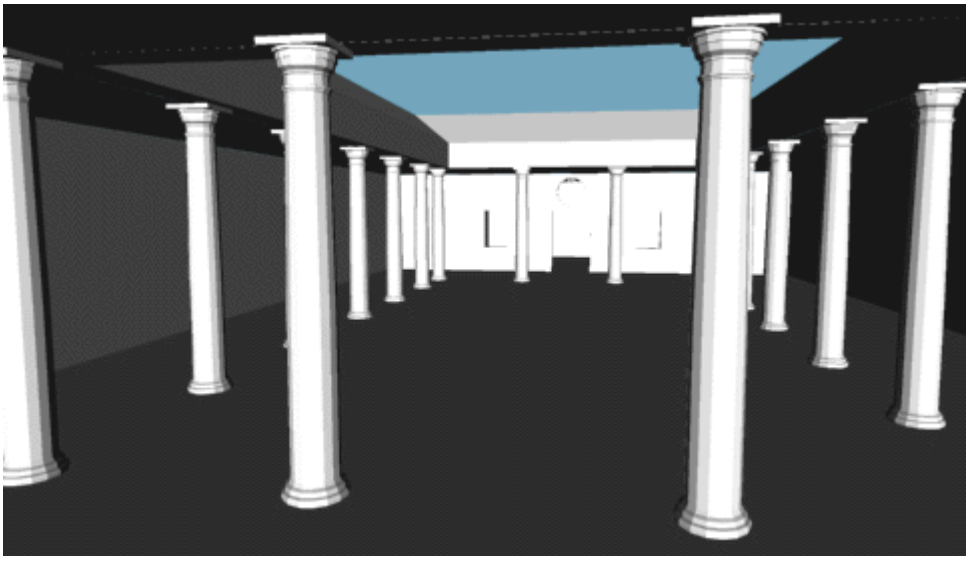
Test the various scenarios you have created thus far by combining them into command scripts. Always test what you have done.

Here is a complete basic test script it should render fairly quick.

```
***Oecuscomplete***
! merge
c:\tutorialscripts\plans\Oecuscomplete02.3dm
! selnone
```

```
***Peristylewallsnontextured***
! merge
c:\tutorialscripts\plans\Peristylcomplete01.3dm
selnone

***Entablature*and*Roof*file***
! merge
c:\tutorialscripts\entablature\vettientab04.3dm
selnone

! merge
c:\tutorialscripts\plans\shield01.3dm
selnone

***Column*Placement*Script***
all
! Hide
! merge
c:\tutorialscripts\columns\tuscancol01.3dm
copy
w0
-9.37803,-4.96942,0
-9.37803,-1.73372,0
-6.82304,-4.96942,0
-3.57283,-4.96942,0
-0.522624,-4.96942,0
2.97761,-4.96942,0
6.12782,-4.96942,0
9.37803,-4.96942,0
9.37803,-1.84309,0
! delete
! SelMesh
! Mirror
0,0,0
12,0,0
enter
! selnone
! show
```

- Change your point of view to 25 degrees and zoom in similar to the view below.
- quick render the model
- Full render the model
- Select all and delete to clear the workspace for the next step.



This is a good time to test lighting and camera setups because this model is a low surface count model. You can move around in it and render the model easily. Check out the images on the next page to see how I developed the current lighting. After figuring out the best lighting scheme I saved only the light set up to a file, combined that as a file merge in a script and tested that.

Lighting and rendering details in Rhino

This would be a good time to test lighting and camera setups because the model created from the simple test commandscript is low surface count model. You can move around in it and render the model easily. Check out the images below to see how I developed the current lighting. After figuring out the best light scheme I saved only the light set up to file, combined that as a file merge in a script and tested that on the higher more detailed surface count model script.

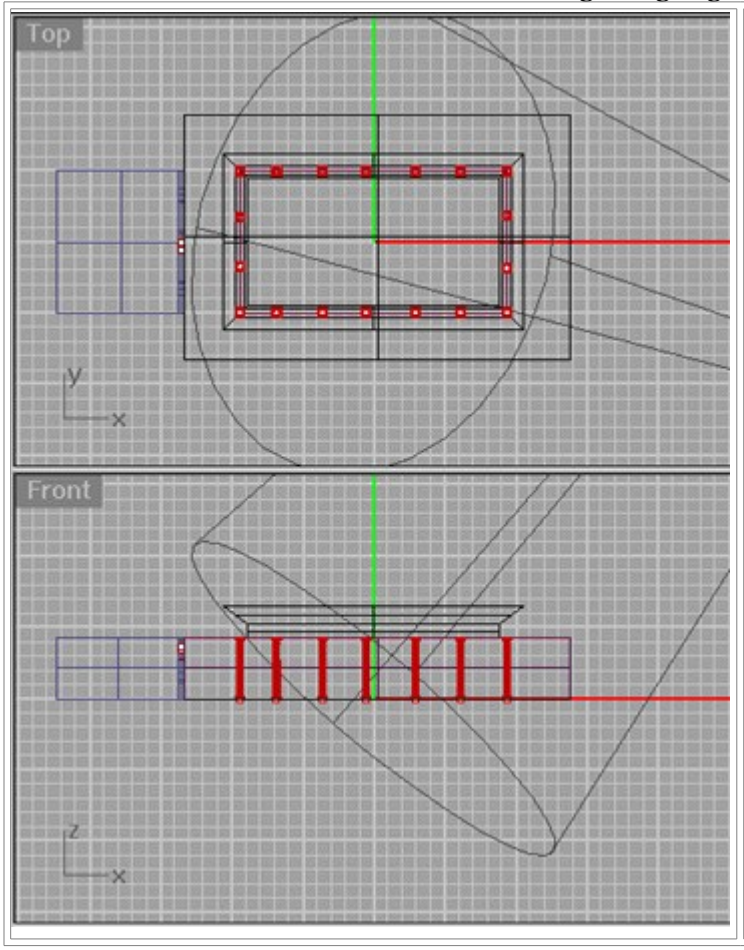
Command scripting really helped here because I was able to test various spotlights on all aspects of the file. This will help because I can now judge where it needs to be lit for export too. When I liked the arrangement of the lights I highlighted the lights and exported them as lights 01.3dm I than opened this file and purged the empty layers and bitmaps and resaved it. I created many light files until I found one I liked.

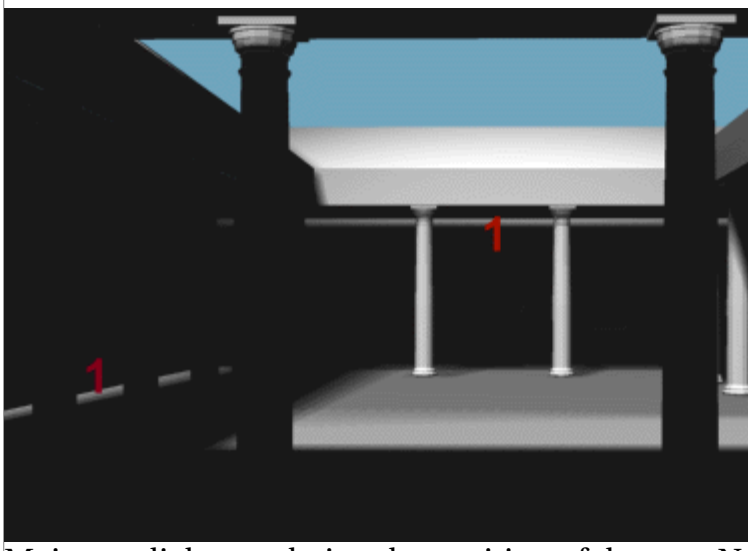
Reminder: The Scale of your model affects the outcome of rendering in Rhino. Because your units for these exercises are set to No units, Rhino assumes the model is really small. The default rendering setting for shadow offset in Rhino is set to .75. Shadow offset is how faraway the shadow is from the shadow casting object. In this tutorial because we are really pretending to work in meters .75 of a meter is too far away from the object to start the shadow of that object. Tighten up the tolerances by making the shadow offset smaller. Set it to.2 . Please make sure your rendering settings are always set to jagged and faster **never** use smooth and slower or custom for any of this final tutorial.

Because Rhino doesn't support advanced rendering features I could only use spotlights of varying intensities and colors. I used the large bright light as the main sun shadow casting light. I introduced the fill lights to solve certain problems. Notice from figure 2-4 how spots solved the darkness of the foreground columns and as a bonus lit the inside of the Oecus. The other colored spotlights simulate radiosity and help lighten the rest of the Peristyle and

Entablature. There is one last thing you need to make in Rhino to get a good rendering. It does not create many polygons. You may notice that there are light leaks between surfaces that are not joined when you render these surface in Rhino, *see Lighting Figure 1*, this is easily solved by creating a light shield around the model. I have already made this for you.

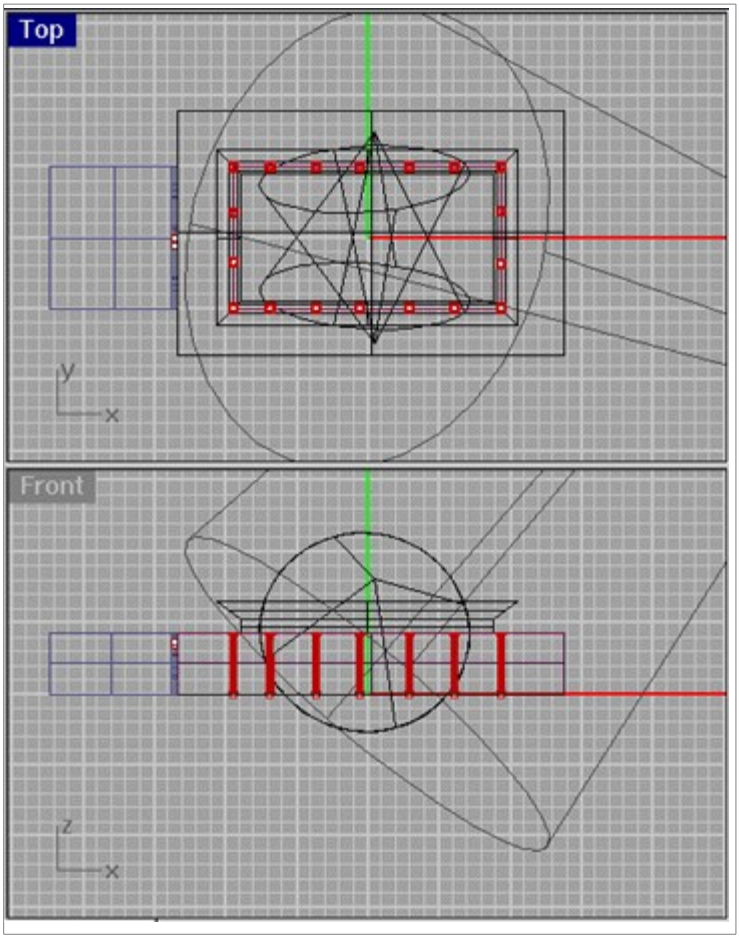
Lighting Figure 1

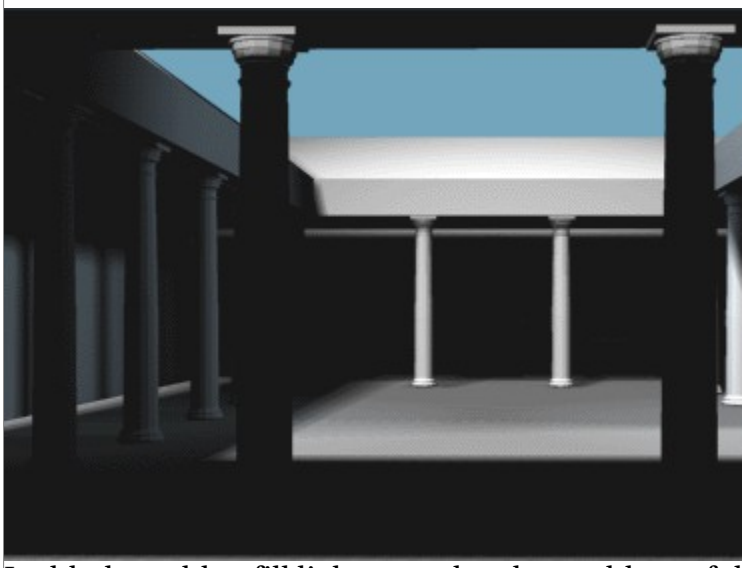




Main spotlight emulating the position of the sun. No leaks at numeral 1. You can see my geometric positio the spotlight in the plan views to the left.

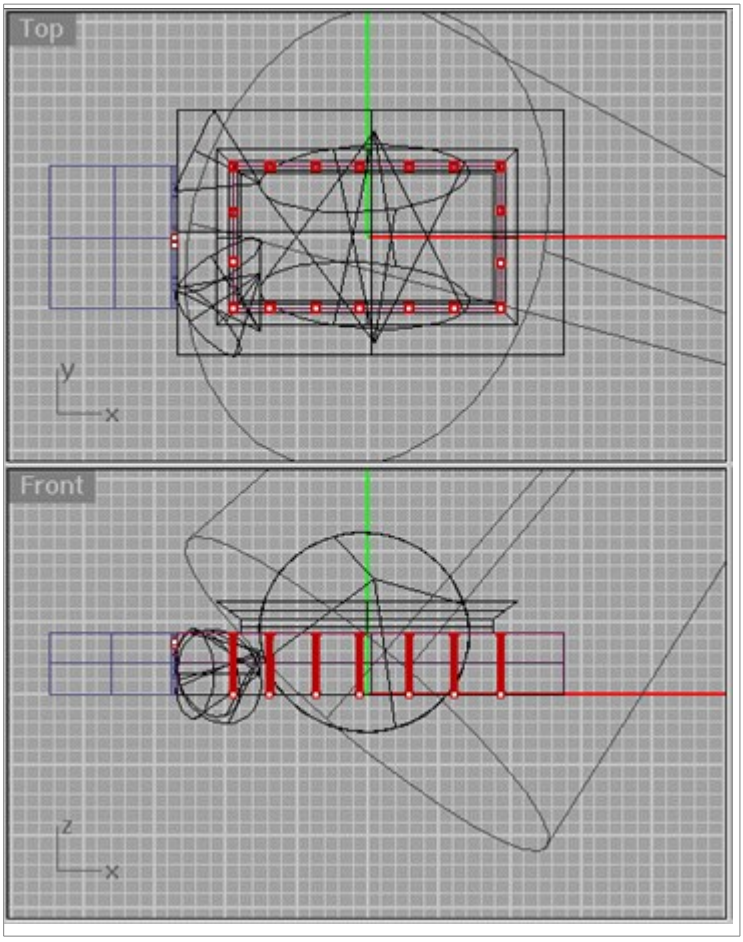
Lighting Figure 2

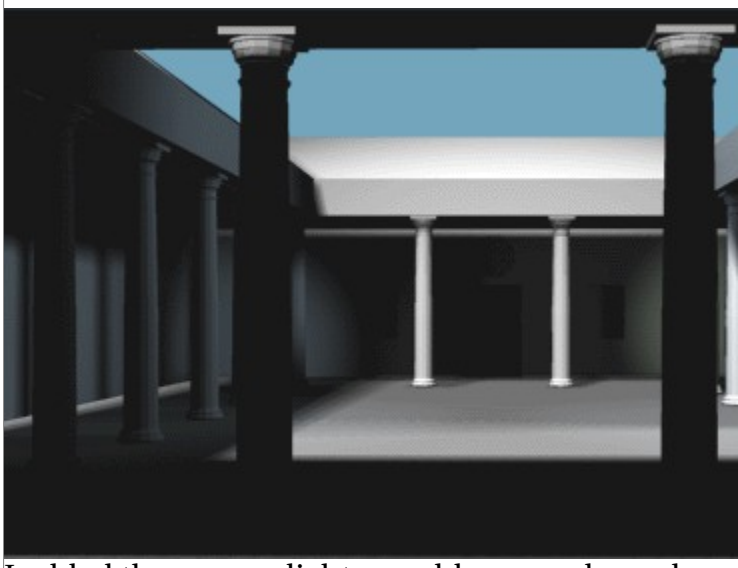




I added two blue fill lights to solve the problem of th
on the sides of the Peristyle.

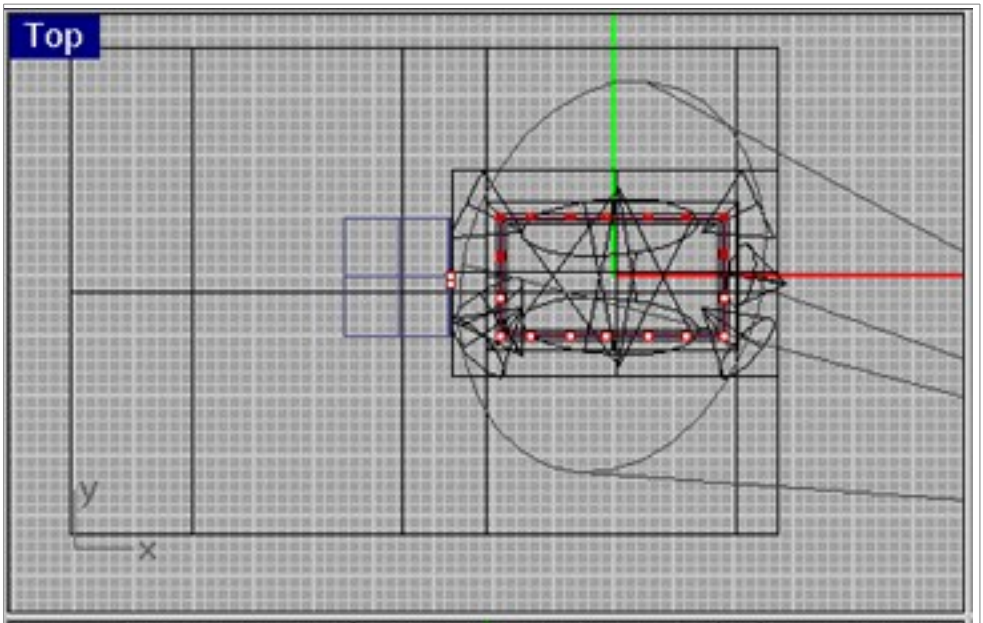
Lighting Figure 3

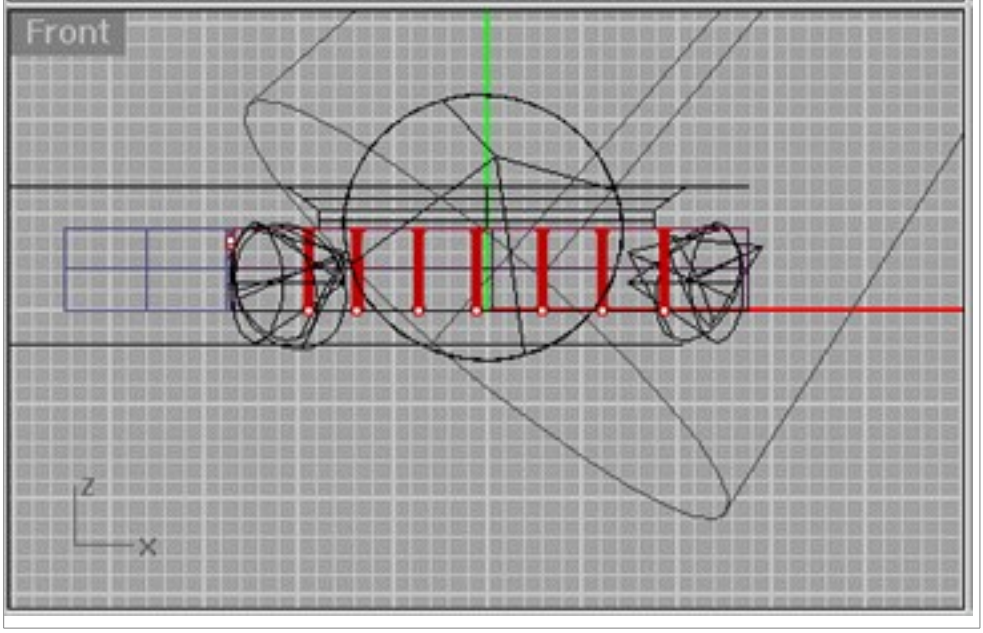


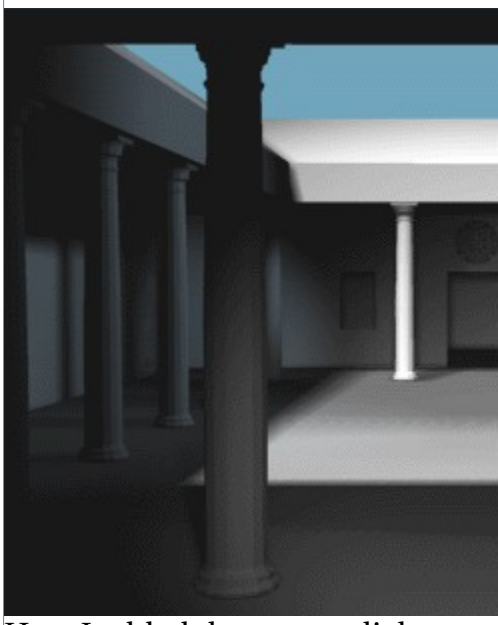


I added three more lights one blue, purple, and gre
the front facade of the Oecus.

Lighting Figure 4







Here I added three more lights nea
the darkness in the foreground. As
only lit up the foreground columns
the background as well.
I also added the Lightshield so tha
can see the Lightshield I added in t
see the settings for the individual
...

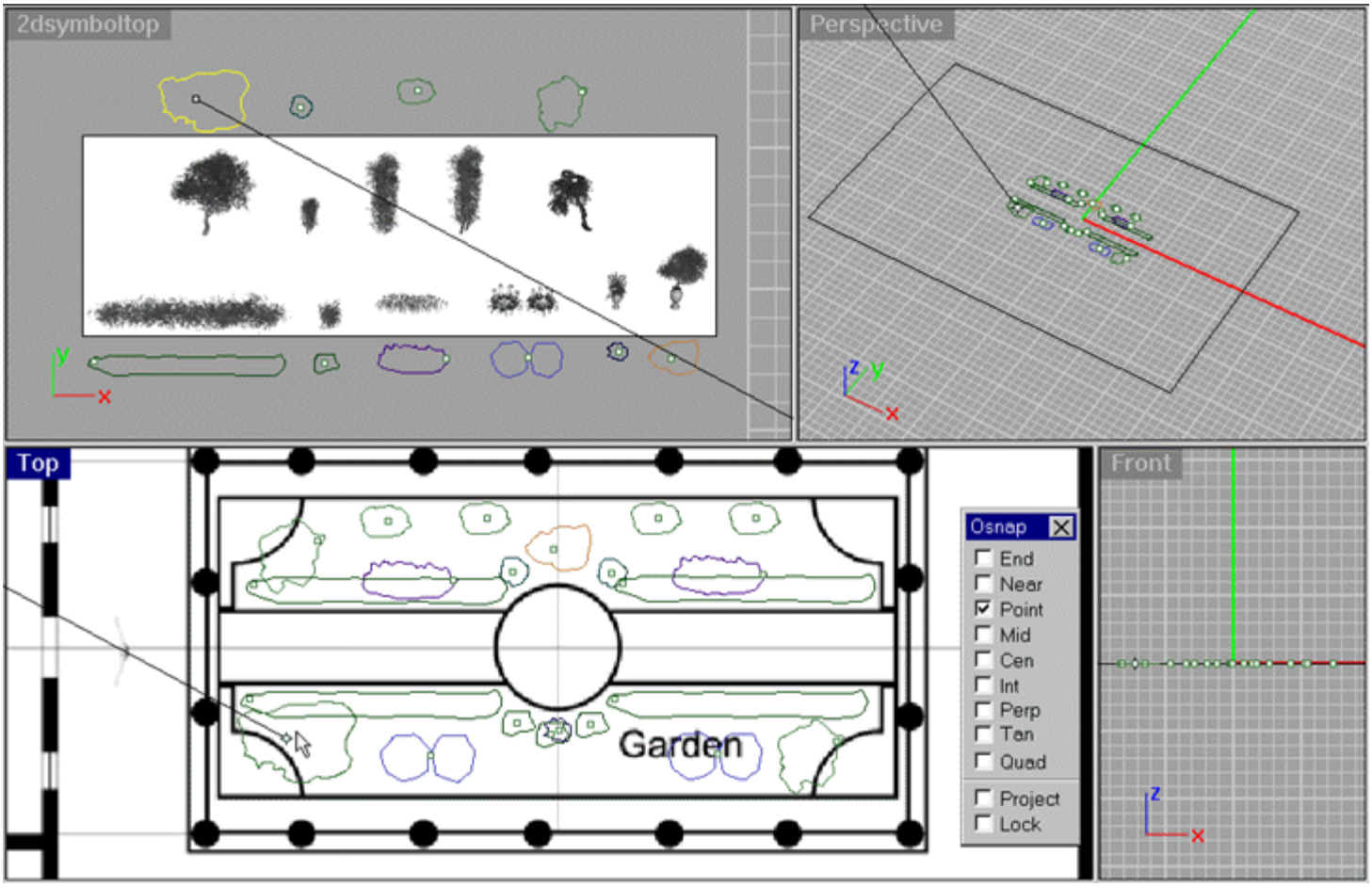
Spotlight*file
! merge
c:\tutorialscripts\plans\lights02.3
selnone

Creating the Garden

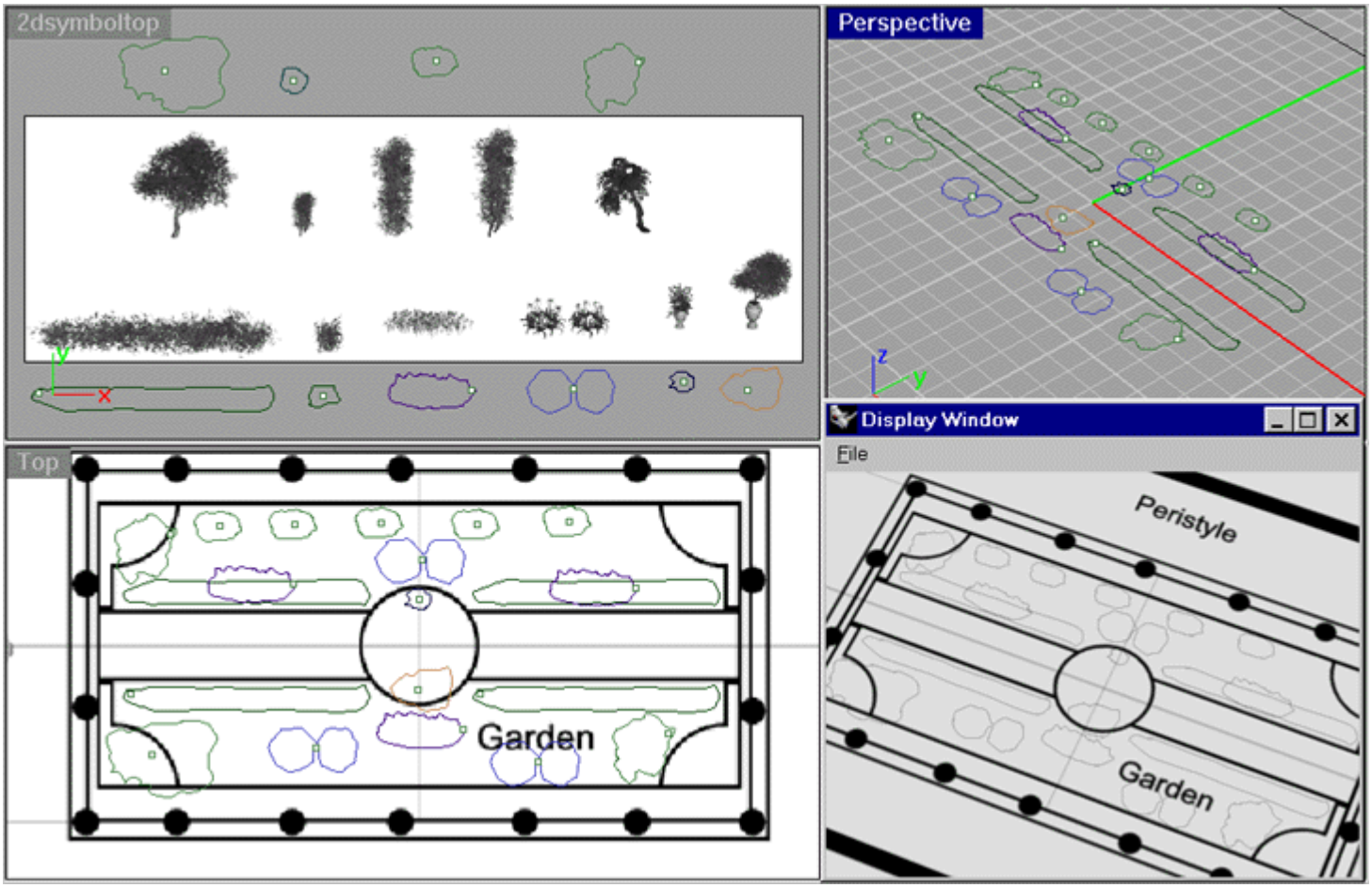
Here you come to some of the fruits of commandscripting. Because 3d plants of any kind use up great amounts of computer resources command scripting is invaluable in this case because it lets you create an unlimited amount of landscaping layouts without the hassle of waiting for screen redraws. You then save the layout to a commandscript.

You do this by creating 2d plant symbols with a point attached to them. By arranging the 2d plant symbols over any plan you can create quickly and efficiently an infinite variety of landscaping layouts. Latter you combine the location of the 2d plants with a file merge of the 3d plants into a script and you have a high polygon count 3d landscape without getting into screen redraw times. I have included all the plant models for you. Latter after you see how the tutorial works you can reference your own 3d plants.

1. I created a library of 3d plants first at the proper scale and saved them at world 0,0,0 as polygon meshes.
2. I also created a library of 2d plants (*see image below*) and traced over the top view of each 3d plant. I assigned their file names to layers but you could also assign them to icons that are on a custom Toolbar called plant library.
3. I arranged the plants using the 2d symbols ontop of a plan of the peristyle garden. Creating different garden layouts was easy because I did it in 2d. *See figure below of a the placement of a tree in action*. The techinique is to keep your 2d plant symbol library away from the top plan view of your subject in this case the Peristyle garden. You copy the 2dsymbol and its' point using snap to point from the 2dsymbol viewport. You then move your mouse into the top viewport and arrange your 2d symbols on the plan. You can play with this yourself by opening the file 2dgrd01.3dm. and moving the symbols around.



A different garden layout below.



4. I opened notepad and pasted the file name and path of each plant file. At first this is a bit tedious but this reference can grow and be used again and again. I have already done this for you in the script below.

5. Using evaluate point command I copied and then pasted the location of each plant into the script. To learn the complete details behind this process and to leran more about my approaches on creating realistic terrain's and landscaping tutorials in Rhino visit my website at www.3dEdge.com

This is only one of many commanscripts that I tested.

Always put your landscaping scripts last in the command script. This is because these are the heavy polygon objects and once imported you can forget doing anything besides clicking the render button.

```
***Complete*Gardenlayout*Script***
! merge
c:\tutorialscripts\plants\grass01.3dm
! selnone

! merge
c:\tutorialscripts\plants\Edgingflowers.3dm
! selnone

! merge
c:\tutorialscripts\plants\fruittree01.3dm
move
w0
-7.22178,-2.40791,0
! enter
! selnone
! merge
c:\tutorialscripts\plants\flowersring01.3dm
! selnone

! merge
c:\tutorialscripts\plants\smalltree02.3dm
copy
```

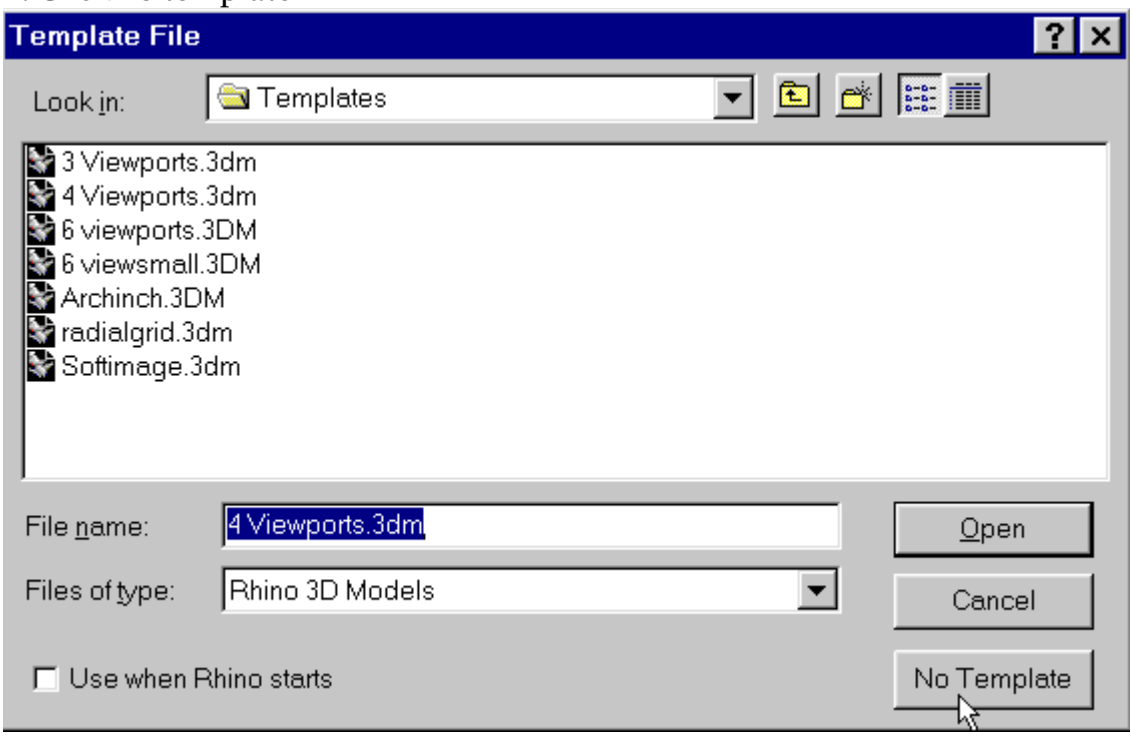
```
w0
-6.39603,2.85085,0
7.23727,-2.60423,0
! delete
! selnone
enter

! merge
c:\tutorialscripts\plants\orangetreevase.3dm
move
w0
0.0268901,2.31754,0
enter
Rotate
R0,0,0
47
! selnone
! merge
c:\tutorialscripts\plants\hibiscusvase.3dm
move
w0
0.036179,-1.26676,0
Rotate
R0,0,0
60
! selnone

! merge
c:\tutorialscripts\plants\cypress.3dm
copy
w0
-4.52721,3.37238,0
-1.8761,3.4593,0
2.68729,3.4593,0
5.29494,3.4593,0
! delete
! selnone
enter
```

Putting it all together

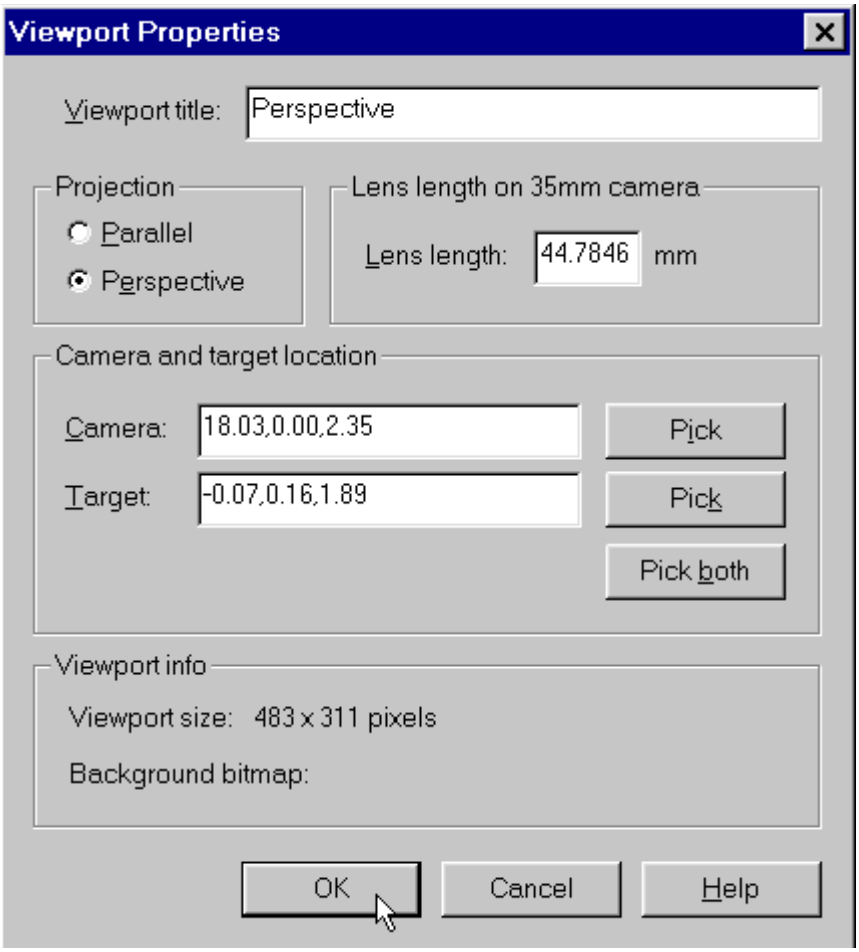
First the simpler parts commandscript that references less detailed models without the garden script added yet. It really is almost same as the last script we ran only with the light files added. One convention I have adopted in writing scripts is to put annotations in the scripts imbeded in *** symbols. This way Rhino cannot read these annotations and passes them up in the command line. I try to keep annotations to a minimum mainly explaining

the filename that is to be merged. I also Include all camera data and field of view information. You must set this first because when the script creates dense surface models you will only want to hit the render button so you wan to set your camera settings before you run the commandscript.

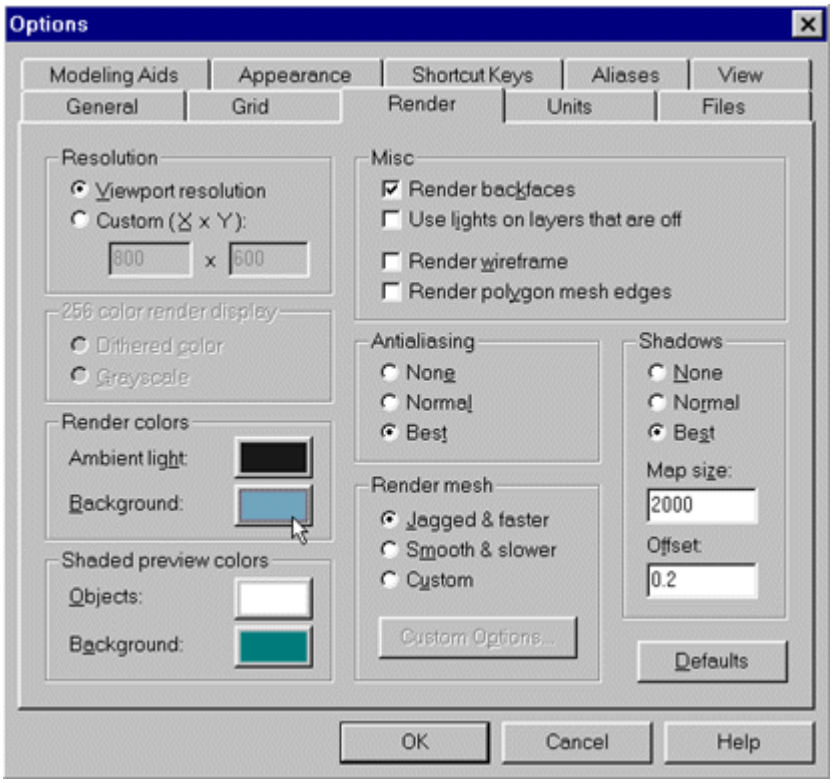
1. Open a new Rhinoceros session
2. Click no template



3. Right mouse click the perspective viewport title. Then click the properties tab copy and paste the following into the appropriate boxes ...
Lens length 44.7846096908265
Camera 18.03,0.00,2.35
Target -0.07,0.16,1.89



4. Click tools/options/render:
Click the Ambient light box and enter 24 24 24 in the RGB boxes not the hue,sat, val boxes.
Click the Background box and enter 112 164 188 in the RGB boxes

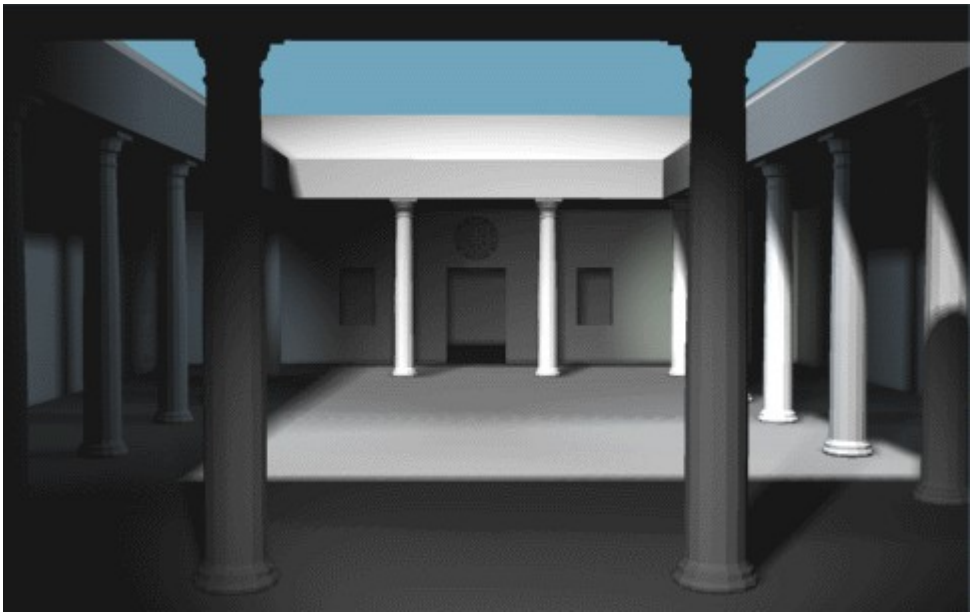


5. Copy the script from ***Oecuscomplete*** to the end of the lefthand table below.
6. Make sure the Perspective viewport is active.
7. Click the command paste icon or type commandpaste at the Rhinoceros commandprompt.

```
*Do*not*copy*and*paste*this*for*reference*only*
background***112 164 188
ambient light***24 24 24
Lens length *** 44.7846096908265
Camera *** 18.03,0.00,2.35
Target *** -0.07,0.16,1.89

***Oecuscomplete***
! merge
c:\tutorialscripts\plans\Oecuscomplete02.3dm
! selnone

***Peristylewallsnontextured***
! merge
c:\tutorialscripts\plans\Peristylcomplete01.3dm
selnone

***Entablature*and*Roof*file***
! merge
c:\tutorialscripts\entablature\vettientab04.3dm
selnone
```

```
! merge
c:\tutorialscripts\plans\shield01.3dm
selnone

***Column*Placement*Script***
all
! Hide
! merge
c:\tutorialscripts\columns\tuscancol01.3dm
copy
w0
-9.37803,-4.96942,0
-9.37803,-1.73372,0
-6.82304,-4.96942,0
-3.57283,-4.96942,0
-0.522624,-4.96942,0
2.97761,-4.96942,0
6.12782,-4.96942,0
9.37803,-4.96942,0
9.37803,-1.84309,0
! delete
! SelMesh
! Mirror
0,0,0
12,0,0
enter
! selnone
! show

***Spotlight*file***
! merge
c:\tutorialscripts\plans\lights02.3dm
selnone
```

8. Create a full rendering of the file.



9. Go to file menu and select new file when prompted to save answer no.
10. Click no template

You should have a new Rhino file/drawing open with no objects in it.

Complete script that references detailed models

Now comes the complete script that references detailed models with the garden script.

1. Make sure you have a fresh Rhino file open you should have this if you went up to step 10 on the previous exercise. Set the View preferences the same as in the last lesson.
2. Make sure perspective viewport is active and maximized.
3. Close this tutorial file. Close all other programs that are running except Rhinoceros.
4. Click the Read Command File or type at the Rhinoceros ! ReadCommandFile and choose from the file picker...
c:\tutorialscripts\scripts\vetticomplete01.txt
5. Click the full render Icon. Notice the rendering starts, it could take an hour or more to render depending on your computer hardware. After rendering exit Rhino and don't save the file there is no need to at this point.

```
*Do*not*copy*and*paste*this*for*reference*only*
background***112 164 188
ambient light***24 24 24
Lens length *** 44.7846096908265
Camera *** 18.03,0.00,2.35
Target *** -0.07,0.16,1.89

**Oecuscomplete***
! merge
c:\tutorialscripts\plans\Oecuscomplete01.3dm
! selnone

***Peristylewallstextured***
```

```
! merge
c:\tutorialscripts\plans\Peristylewalls01.3dm
! selnone

***Entablature*and*Roof*file***
! merge
c:\tutorialscripts\entablature\vettientab03.3dm
selnone

***Peristyleceiling*file***
! merge
c:\tutorialscripts\plans\Peristyleceiling.3dm
selnone

! merge
C:\tutorialscripts\Plans\Peristyleflrgarden.3dm
selnone

! merge
c:\tutorialscripts\plans\shield01.3dm
selnone

***Column*Placement*Script*for*Vettii***
! all
! Hide
! merge
c:\tutorialscripts\columns\Corinthmesh.3dm
copy
w0
-9.37803,-4.96942,0
-9.37803,-1.73372,0
-6.82304,-4.96942,0
-3.57283,-4.96942,0
-0.522624,-4.96942,0
2.97761,-4.96942,0
6.12782,-4.96942,0
9.37803,-4.96942,0
9.37803,-1.84309,0
! delete
! SelMesh
! Mirror
0,0,0
12,0,0
enter
! selnone
! show

***Spotlight*file***
```

```
! merge
c:\tutorialscripts\plans\lights02.3dm
selnone

***Complete*Gardenlayout*Script***
! merge
c:\tutorialscripts\plants\grass01.3dm
! selnone

! merge
c:\tutorialscripts\plants\Edgingflowers.3dm
! selnone

! merge
c:\tutorialscripts\plants\fruittree01.3dm
move
w0
-7.22178,-2.40791,0
! enter
! selnone
! merge
c:\tutorialscripts\plants\flowersring01.3dm
! selnone

! merge
c:\tutorialscripts\plants\smalltree02.3dm
copy
w0
-6.39603,2.85085,0
7.23727,-2.60423,0
! delete
! selnone
enter

! merge
c:\tutorialscripts\plants\orangetreevase.3dm
move
w0
0.0268901,2.31754,0
enter
Rotate
R0,0,0
47
! selnone
! merge
c:\tutorialscripts\plants\hibiscusvase.3dm
move
w0
```

```
0.036179,-1.26676,0
Rotate
R0,0,0
60
! selnone

! merge
c:\tutorialscripts\plants\cypress.3dm
copy
w0
-4.52721,3.37238,0
-1.8761,3.4593,0
2.68729,3.4593,0
5.29494,3.4593,0
! delete
! selnone
enter
```