An Introduction to Scripting with CalcPlot3D

This tutorial will help you begin using and creating scripts for **CalcPlot3D**. Scripts allow you to customize this multivariable calculus visualization tool.

You can use scripts to create:
Classroom Demonstrations
Guided Student Exploration Activities
Custom Starting Points for the Applet
Your own default initial settings

Classroom Demonstrations: It is often useful to demonstrate how to enter functions, space curves, etc. in CalcPlot3D during class, but sometimes it would be nice to quickly step through a series of visual examples, each of which can be manipulated and explored as much or as little as time allows and you desire.

Guided Student Exploration Activities: Students gain the most understanding from playing with and exploring the concepts visually themselves. The more directly students can be engaged in the visualization and exploration of calculus concepts the more they will internalize and the clearer will be their geometric intuition.

Custom Starting Points for the Applet: Scripts can be used to start the applet with your particular custom functions, graph settings, and 3D viewing window. This may be where you always want your students to start their exploration, or perhaps you may wish to create a whole series of applets that each have a single exploration step, but illustrate a particular topic.

Your own default initial settings: Although this purpose is contained in the previous one, it is worth noting that you can use a script to simply set the default startup settings you wish to have always in place when you start the applet from your webpage.

Scripts can be loaded into **CalcPlot3D** in two ways. The primary method used for all but Demonstrations is to use a webpage to load the **CalcPlot3D** applet and set the script parameter in HTML to be the desired script file. Scripts can also be loaded directly from the applet once it is already running. This is most useful for loading up classroom demonstration scripts. We will use this second method first as we explore how the scripts work below.

Scripts can also be created in two ways. We can create them directly in a text editor using the example scripts provided on the CD as templates. We can also use the applet to create a script by saving steps one at a time. Although this method can be easier initially, it is still very useful to be able to edit the script in a text editor to reorder the steps and to fine tune the settings without having to start over and create a new script. We will explore both methods here.

Scripting is a new feature in **CalcPlot3D**, and it will continue to be expanded and enhanced in the coming year. Please let the presenter know if you have suggestions or discover errors.

Now let's take a look at an example script. We will start with a demonstration script since it will give a general picture of what can be done with scripting in **CalcPlot3D**, without being as wordy as an exploration script.

Wherever I refer to the CD, you can also find these files in an online directory at:
http://web.monroecc.edu/manila/webfiles/calcNSF/JavaCode/scripts/

1. On your CD, find the folder named, **CalcPlot3D Scripts**. In this folder, open the script named **DomainDemo** with Wordpad, Notepad, or another text editor. You should see the script shown below.

Most elements of the script are listed one per line when saved by the applet in the script, but we can place multiple elements on the same line to save space, as is done here.

```
<init steps="6">
<step 1>
        <window>
                xmin ="-2"
                xmax ="2"
                xscale ="1"
                ymin ="-2"
                ymax ="2"
                yscale ="1"
                zmin ="-2"
                zmax ="2"
                zscale ="1"
                zMinClip ="-4"
                zMaxClip ="4"
                anaglyph ="none"
                edgesOn ="true"
                facesOn ="true"
                opaque ="true"
                transparency ="140"
                smooth ="false"
                antialiasAll ="false"
                showBox ="true"
                perspective ="true"
                gridSize ="25"
                zoom ="0.7"
        </window>

        <viewPoint      center="(7.76, -4.22, 4.69)"
                        focus="(0.0, 0.0, 0.0)"
                        up="(-0.275, 0.443, 0.853)"/>

        <function type ="z = f(x, y)" function = "(x+y)/(x-y)" num ="1" visible = "true" />
</step>

<step 2>
        <window>
                xmin ="-2" xmax ="2" xscale ="1" ymin ="-2" ymax ="2" yscale ="1"
                zmin ="-2" zmax ="2" zscale ="1" zMinClip ="-4" zMaxClip ="4"
                perspective ="true"
        </window>

        <viewPoint  center="(8.24, 4.755, 3.09)" focus="(0.0, 0.0, 0.0)" up="(0.0, 0.0, 2.0)" />
```

Init must come first and must specify the number of steps in the script. Note the steps all start with <step #> and end with </step>. This is a form of XML.

These elements set the parameters for the 3D axes.

Note that every element's value must be given inside quotation marks. Also note that case is presently important. (This may change.)

All window settings are listed here since this script was saved from the applet. Generally not all settings need to be set for each step in a script, but in order to set the axes settings, all the axes settings must be set in the script.

These elements set the bottom and top z values at which to clip the surface.

This element specifies which 3D viewing option to use for the step.
This element sets the black edges of the wire-frame surfaces on/off.
This element sets the colored faces of the wire-frame surfaces on/off.
This element sets whether all surfaces should be opaque(or transparent).
Transparency allows you to set the level of transparency (0-255).
If smooth = "true", the surface is drawn with a special gradient fill.
When on, all edges are antialiased. Can be slower to rotate.
By default, a box is shown around the axes.
Usually set to true. Set to false when viewing a 2D trace of surface.
Sets the number of grid-squares to use in each direction in xy-plane.
Sets the relative size of image without changing the axes settings.

These parameters set the viewpoint.
Center sets the location of the eye.
Focus sets the center of the view.
Up sets the vector for the up direction.

Here is a typical function definition. Num tells which of the four functions to set. Visible is true by default. Type sets which of the three combinations of x, y, and z to use for the definition.

```
        <function type ="z = f(x, y)" function = "sqrt(4-x^2-4y^2)" num ="2" visible = "true" />
</step>

<step 3>
        <window>
                xmin ="-2" xmax ="2" xscale ="1" ymin ="-2" ymax ="2" yscale ="1"
                zmin ="-2" zmax ="2" zscale ="1" zMinClip ="-4" zMaxClip ="4"
                perspective ="true"
        </window>

        <viewPoint  center="(8.24, 4.755, 3.09)" focus="(0.0, 0.0, 0.0)" up="(0.0, 0.0, 2.0)" />

        <function type ="z = f(x, y)" function = "ln(xy)" num ="3" visible = "true" />
</step>

<step 4>
        <window>
                xmin ="-2" xmax ="2" xscale ="1" ymin ="-2" ymax ="2" yscale ="1"
                zmin ="-2" zmax ="2" zscale ="1" zMinClip ="-4" zMaxClip ="4"
                perspective ="true"
        </window>

        <viewPoint  center="(0.0, 0.0, 10.0)" focus="(0.0, 0.0, 0.0)" up="(0.0, 2.0, 0.0)" />

        <function type ="z = f(x, y)" function = "ln(xy)" num ="3" visible = "true" />
</step>

<step 5>
        <window>
                xmin ="-2" xmax ="2" xscale ="1" ymin ="-2" ymax ="2" yscale ="1"
                zmin ="-2" zmax ="2" zscale ="1" zMinClip ="-4" zMaxClip ="4”
                perspective ="false"
        </window>

        <viewPoint  center="(0.0, 0.0, 10.0)" focus="(0.0, 0.0, 0.0)" up="(0.0, 2.0, 0.0)"/>

        <function type ="z = f(x, y)" function = "sqrt(4-x^2-4y^2)" num ="2" visible = "true" />
</step>

<step 6>
        <window>
                xmin ="-2" xmax ="2" xscale ="1" ymin ="-2" ymax ="2" yscale ="1"
                zmin ="-2" zmax ="2" zscale ="1"
                zMinClip ="-100"
                zMaxClip ="100"
                perspective ="false"
        </window>

        <viewPoint  center="(0.0, 0.0, 200.0)" focus="(0.0, 0.0, 0.0)" up="(0.0, 2.0, 0.0)"/>

        <function type ="z = f(x, y)" function = "(x+y)/(x-y)" num ="1" visible = "true" />
</step>
```

Some objects have two ways to set them off, but all can use the standard <object> statement to start the definition and </object> to end it. Note that **function** and **viewPoint** are defined using a short-hand version of this. For example, <function function = "x + y" num = "1" />. **Step, text,** and **window** cannot be set off this way, but pretty much everything else can.

2. Now let's try the script by loading it into **CalcPlot3D**. Open the applet and find the option to load a script on the File menu in the upper left corner of the applet. Now locate the script file listed in step 1 and load it into the applet. (If you don't have a CD, and you are online, you can find these scripts on the **Example Script** menu option to find these scripts or you can copy the files from the website shown above to your local drive and then use them there. Right-click and save for each file.)
3. Step through the 6 steps using the exploration box located at the right side of the screen. This script is meant to be used in class as a visual demonstration after specifying and drawing the domains and ranges of various functions with some limits on their natural domains.
4. You may notice that the steps at the end should be placed with their corresponding surfaces from earlier in the exploration. These steps were recorded using the applet, and similar to how a macro works in Excel or Word, we can edit them after they have been recorded. Let's reorder the steps in the script in your text editor so that they show each surface first and then its view from above the xy-plane. Be sure to copy and paste the entire step for each one. Then you can renumber the steps, if you like. The actually step numbers are ignored by the applet. They are just there for our benefit.
5. Save the script to your computer (note it won't save on the CD). Then load it into the applet to check that it works and has the correct order.

6. Now let's add some text to the exploration box for each step. In the text editor, add a line just after <step 1> and just before <window>.
7. On this line, enter:

<text> This is an interesting discontinuous
function of two variables.</text>

Note that everything between the two delimiters <text> and </text> will be displayed on the exploration box, including spaces, carriage returns, and empty lines. Presently, you will need to be careful to keep the lines short, as there is no word-wrapping in the exploration box yet. There are some special formatting commands available, but these will be covered later, and they are still in development. Many HTML formatting codes will eventually be supported. Presently these codes are my own invention.

8. Now try changing the functions in the script. You can vary the window and viewpoint parameters also if you wish. You might try changing the third function to **ln(y - x^2)** for example. Save the script after each change and load it again into the applet.
9. Next let's add a step, using the text editor. Copy an entire step showing a normal view of a function, like step 1. Be sure to start at the <step 1> and end by including the </step> statement. Paste this step at the end of the script.
10. Go back to the top of the script and change the number of steps to 7. Otherwise the step we just added will be ignored by the applet.
11. Now edit the function in the new step to be **arcsin(y – x)**. Edit any other parameters you wish to and then save the script and load it into the applet to try it out.

Next we will add a step using the applet.

12. First let's create the step view. Clear the functions shown and enter the function **z = 1/(y – x^2)** as the first function. Press enter when you finish, or select the checkbox next to Function 1 and click on the Graph button. Adjust the viewpoint using the mouse and make any changes to how the surface appears using the **View Settings** menu.
13. Now from the **File** menu at the upper left of **CalcPlot3D**, choose the option to **Add the current view as a new step to the current script** from the **Script** submenu. After a second, you should see the plot window briefly show the first step of the script and then it will display the step you have just added. Notice that there is now an additional step (8 steps total if you did not add an extra step above).
14. Next we should save the script. Scripts are not automatically saved when you add a new step. Be sure to save it if you wish to keep your additions. To save the script from CalcPlot3D, select the **Save Script** option on the **Script** submenu. The first time you save it, you will be asked to select the folder and file name. After this, the script will simply be saved in the background without asking you anything more, just like the **Save** option works in Word and other applications.
15. Open the script in your text editor. I usually either double-click on the script file or I right-click it and select the editor I wish to use. Check to see that the new step is there and notice that it lists all parameters, line by line, even though many of them really don't have to be set each time. These can be edited or removed, as you wish. Refer to the **Script Reference Sheet** for more information on what the default values are for many of these settings.

Next we will create a script from scratch using the applet and edit it in the text editor.

16. This script will examine a couple of surface intersections. The first thing we need to do is create a new script. Choose this option on the Script submenu in CalcPlot3D. You will be asked for a script name. A default name of **myScript** is supplied. Let's call it **surfaceInts**. When you click OK, nothing special seems to happen, except that the exploration box disappears if you had a script loaded. The old script is discarded and the applet is ready to record new steps.
17. Let's graph the surfaces defined by: $z = x^2 + y^2$ and $y = x$. To enter the second function, you will need to either right click on the z = in front of function 2 or use the **Function Type** submenu on the **View Settings** menu. Select the type y = f(x, z).
18. To make more of the plane show, choose **Format Axes** from the **View Settings** menu (or press the A key). This will bring up the **Format Axes** dialog. Use it to change the **z-max** value to **4**. This will automatically raise the **zClip-top** value to 8. Change this value back to **4** and click on the **OK** button to make these settings take effect. Then close the **Format Axes** window.
19. Make the surfaces appear semi-transparent by selecting the appropriate menu option on the View Settings menu (or by typing Ctrl-T). This will make it easier to see the intersection.
20. Next see if you can determine a space curve that will trace this intersection. Hint: Let $x = t$. Graph this space curve to verify it goes through the intersection.
21. Now it is time to add this step to our script. Select the Add step option on the Script submenu. An exploration box should appear to the right, and the first step is recorded.
22. Next create the intersection of the surfaces defined by: $x^2 + y^2 = 4$ and $z = x^2$. Here you may wish to solve the first equation for x or y and graph the cylinder in two pieces, or you can use cylindrical coordinates to graph it with just one. (To graph in cylindrical coordinates, either right click on the z = or choose the appropriate function type on the Function Type menu.)
23. Next see if you can determine a space curve that will trace this intersection. Hint: Let $x = 2\sin t$. Graph this space curve to verify it goes through the intersection.
24. Add this step to the script and then save the script as we did in step 13 above.

It is now time to create a web-page that calls the applet and automatically loads a script at start-up. As is described above, this technique allows you to customize the start-up options of the applet and/or allows you to create explorations and illustrations for your students.

25. First let's load a template into the text editor. Locate the file named **CalcPlot3Dcall.htm** in the folder named **CalcPlot3D Scripts**. This file is set up to work with the real CalcPlot3D applet on my website. Since we are not on the internet here, we will be using the other .htm file to test our skills here. But first let's look at the real thing and discuss it a minute. Below are the key lines of HTML code:

    ```
    document.write('<table border="3" cellpadding="0" cellspacing="0" width=100% height=100%
    bordercolor=blue>');
    document.write('<tr><td>');
    document.write('<applet name="CalcPlot3D"
    codebase="http://web.monroecc.edu/manila/webfiles/calcNSF/JavaCode/"
    archive="CalcPlot3Dsigned.jar" code="Calc3D/CalcPlot3D.class"');
    document.write('width=100% height=100% Mayscript');
    document.write('alt="Your browser understands the applet tag but isn\'t displaying any applet.">');
    document.write('<param name="FireFox" value='+inFF+'>');
    document.write('<param name="mode" value="script">');
    document.write('<param name="script" value="test.txt">');
    document.write('</applet>');
    ```

26. Note that the **codebase** parameter is set to the URL address where my applet resides. This tells the browser to load the applet from that location, even though your html webpage is somewhere else.

    The script parameter is the one you will be changing. Notice that presently the value assigned to the script parameter is "test.txt". Be sure to place your script file's name in quotation marks. You don't need any path information if the script file is in the same folder with the webpage, but you can use a relative path instead, if the file is not located in the same folder as the webpage.

27. Now open the other .htm file named, **CalcPlot3D.htm**. This file has a path to the applet locally referenced. Note that it does not have a codebase parameter at all. This indicates that the applet is being called on a relative path from the current directory. (You do NOT need this when online!)
28. Change the value of the script parameter to one of the scripts we have created. Then save the file.
29. Now double-click on this file to open it in a browser. The script you chose should automatically appear when the applet runs. (If you are online, use the **CalcPlot3Dcall.htm** file above for steps 28-29.)

This concludes this part of the tutorial. Your next task is to create an exploration or demonstration script of your own. You can try any of the following ideas or come up with something of your own.

An exploration/demonstration of:

- velocity and acceleration similar to the one I have provided in the scripts folder.
- functions in spherical and/or cylindrical coordinates.
- Parametric surfaces
- More intersections of surfaces
- Lagrange Multiplier Optimization
- Level Surfaces
- Gradient vectors for functions of 2 and/or 3 variables.