Creating Interactive Mathematics Web Pages Using CaluMath

Introduction: CaluMath is free software designed for the creation of interactive web pages involving mathematics. CaluMath pages are written entirely in HTML, which is the standard markup language for web pages, and JavaScript, a well-known programming language native to web browsers that allows pages to be dynamic. Web pages constructed using CaluMath can display graphs or animations and contain all of the items commonly found in web pages, such as text boxes, drop down menus, and buttons that allow the user to provide input, make selections, manipulate graphs, answer questions, and interact with the pages in a meaningful way. In response to user interaction, items on the page can appear or disappear, graphs can be drawn or modified, new windows opened, new information displayed, or contrasting scenarios presented. Correct user answers can be challenged with a new scenario to see if the answer was the result of a memorized algorithm. Incorrect user responses can lead to activities that help the user see their response as problematic in terms of other knowledge the user possesses.

The interactive CaluMath web pages are constructed using the CaluMath Page Maker, a web page interface designed for the easy construction of CaluMath pages. The CaluMath Page Maker main menu is a table listing all of the mathematical and web objects that can be created using CaluMath. When a page author selects an object, a screen appears with boxes and menus pertaining to the construction of the object. The author selects the properties of the particular object that she wants, and clicks a button to finish. This returns her to the main menu where the process can be repeated. Using this procedure, functions and random numbers can be defined, graphs created, points plotted, and tangent lines drawn. In the same manner, titles, paragraphs, tables, menus, text boxes, and buttons can be created. The actions that occur when a user clicks on a button or graph are also created in a similar fashion. No knowledge of HTML or JavaScript is required.

The CaluMath Page Maker can load any saved CaluMath web page, regardless of who authored it. When this is done and the Edit button is clicked, a window opens that reveals the structure of the page and lists all the items created. This gives someone who did not construct the page the ability to modify it and save it to their own web site.

CaluMath is not tied to a particular mathematical content or grade level and can be employed at any level of the K-12 school curriculum and at the junior college and university levels. CaluMath pages employ standard HTML and JavaScript, ensuring that they will not become obsolete as technology changes. CaluMath is open-source software and is licensed under the GNU license.

Note that everything in CaluMath runs in a web browser, therefore you do not need to install anything on your computer.

The Components of CaluMath:

1. The CaluMath program consists of libraries of JavaScript code written specifically to enable web pages to display meaningful mathematics and allow users to interact with it. Users of CaluMath do not need to concern themselves with the underlying program.
2. The CaluMath Page Make is a web interface that allows users to construct their own interactive web pages and modify pages constructed by others.
3. The CaluMath web pages that have been authored using the CaluMath Page Maker can be used by anyone without any training. They can be employed by being projected in front of a

classroom or assigned as homework.

Everything in CaluMath runs in a web browser, therefore you do not need to install anything on your computer. You can download the entire CaluMath web site and save it to your hard drive and open CaluMath web pages by clicking on the files; there is nothing to install. When you interact with a CaluMath web pages on the internet, all calculations are done on your computer, no further connection with a web site is necessary. This means that instructors can create CaluMath pages and place them on their web sites just as they would place any other web page there. In particular, you do not need to configure your web server in order to host CaluMath pages.

CaluMath design criteria: CaluMath was designed to meet three objectives:

1. **CaluMath is designed to be available to the widest possible audience, including all grade levels K-16.** To meet this objective, CaluMath is web-based, open source, and does not rely on plug-ins.

2. **CaluMath is designed for mathematical education and produces fully functional web pages. The page designer controls the functionality that the student can bring to a given task and the responses the page makes to user interactions.** A page designer can display or not display the equation of a graph that is presented, can reveal or hide the scaling of axes in a graph, allow or preclude the student from ascertaining exact coordinates of points on the graph, and include or not include the functionality to draw tangents to a curve at a given point. This allows the page designer to take students outside of their comfort zone by not giving them the tools they would normally rely on in a given situation, thus forcing them to contemplate a scenario in more than one way. Equally importantly, the page designer chooses how students' responses are to be evaluated by the page; for example, if a student drags a graph, the page can determine the new position and provide the student with questions dealing specifically with the new position. This cycle of capturing student's responses or interactions with the page in order to provide new questions or scenarios is one of the distinguishing features of CaluMath. In addition, since CaluMath pages are fully functional web pages, CaluMath pages can respond to the user in many ways, such as displaying a paragraph analyzing what the student did, graphing arrows that point out features to which the student should pay attention, opening a new window containing new information or graphs, or displaying buttons that provide a graph with new functionality that the student can now employ.

3. **CaluMath pages can be continually improved by the mathematics and mathematics education communities, leading to cycles of creation, evaluation, improvement, and adaptation.** A previously constructed CaluMath page can be opened by anyone using the CaluMath Page Maker; the structure of the page will be revealed and modifications can be made. Changes can be in response to the latest research in mathematics education, assessment results, or simply because a new instructor has a fresh perspective. Once the changes are made, the instructor can post the page on their own web site, share it with their peers, the original page author, and the mathematics community in general.

Current Version of CaluMath: Version 1.01 is currently on the CaluMath web site: http://ems.calumet.purdue.edu/mcss/psturbek/CaluMath/CaluMath_HomePage.html.
Version 2.0 will be released by Summer 2010. The Beta version of the CaluMath Page Maker for this

new version is available now, and will be used in this tutorial. It can be found here:
http://mathvserver.calumet.purdue.edu/~psturbek/CaluMath/CaluMath Beta Home Page.html

The 2.0 version of CaluMath contains significant improvements in terms of ease of use of the CaluMath Page Maker. Since it is still in Beta testing, it may have tiny bugs. The Beta version works very efficiently in the browsers Firefox, Google Chrome and Safari. It currently works sluggishly in Internet Explorer and Opera, because we have not completed the code optimization yet for this release. When CaluMath 2.0 is released, it is expected to work well in all major browsers. CaluMath is also compliant with older versions of many web browsers.

How Do I Put CaluMath On My Web Site?

To place CaluMath on your web site, you merely need to copy CaluMath web pages and some supporting library files to your web site. There is nothing to install, and your web server does not have to be configured in any way. There are two ways to do this.

1. Go to the CaluMath web site, and download the zip file that contains the entire web site. Unzip it on your hard drive. Then copy the entire CaluMath folder to your web server. You can then place any CaluMath Pages you create on your web server in the MyWebPages directory.
2. To place your own CaluMath web pages on your web site, you do not need to actually upload the entire CaluMath web site to your web server as described above. We now describe the minimum number of files and folders you need to place on your web server. To do this, do the following. Go to the CaluMath web site and download the zip file that contains the entire web site. Unzip it on your hard drive. You should use the CaluMath Page Maker (run off your own computer) to create web pages. When you want to place them on your web server, this is the minimum number of files and folders you need to create.
   a) Create a folder named CaluMath on your web server.
   b) Go to the copy of CaluMath on you computer and find the library folder inside the CaluMath directory. Copy this entire folder to the CaluMath folder you just created on your web server.
   c) Copy the MyWebPages folder to the CaluMath folder you just created on your web server.
   d) Place any CaluMath pages you create in the MyWebPages folder on your web server.

After you have completed either steps 1) or 2) above, you should be able to go to any web page on your server in the MyWebPages folder and open it in your web browser and it should display correctly.

Once you have CaluMath on your web server, you may want to create special folders for specific types of pages. For example, suppose you want to create a folder called Algebra in which you will place CaluMath web pages you create. To do this, you must observe the following.

1. The Algebra folder must be created somewhere inside the CaluMath folder. The Algebra folder can be placed directly inside the CaluMath folder, or it can be placed in any folder inside (or a folder inside a folder inside etc.) the CaluMath folder.
2. The following files must be placed inside the Algebra folder: *place_this_in_folder_with_web_pages_vml.html, place_this_in_folder_with_web_pages.svg, place_this_in_folder_with_web_pages.js, and place_this_in_folder_with_web_pages.html.* These files are necessary for the CaluMath pages in the Algebra folder to work correctly. You can cut and paste them from the MyWebPages folder into the Algebra folder.
3. Never delete the four files above from the MyWebPages folder.

Tutorial

This tutorial is divided into two parts.

1. An overview of CaluMath including defining functions, constructing graphs, creating buttons, opening and saving pages.
2. The construction of a fully functioning web page in which two points are plotted and the graph of a cubic function h whose local maximum and minimum occur at the two points is displayed. The graph of h′ is also displayed. The user is asked to drag the points so that h(3) = h′(3) +12. The user's answer is checked and appropriate responses are displayed.

Although we want to produce a functional web page in 2) above, we will do so in a roundabout manner that allows us to discuss features in CaluMath in greater depth.

Preparation:

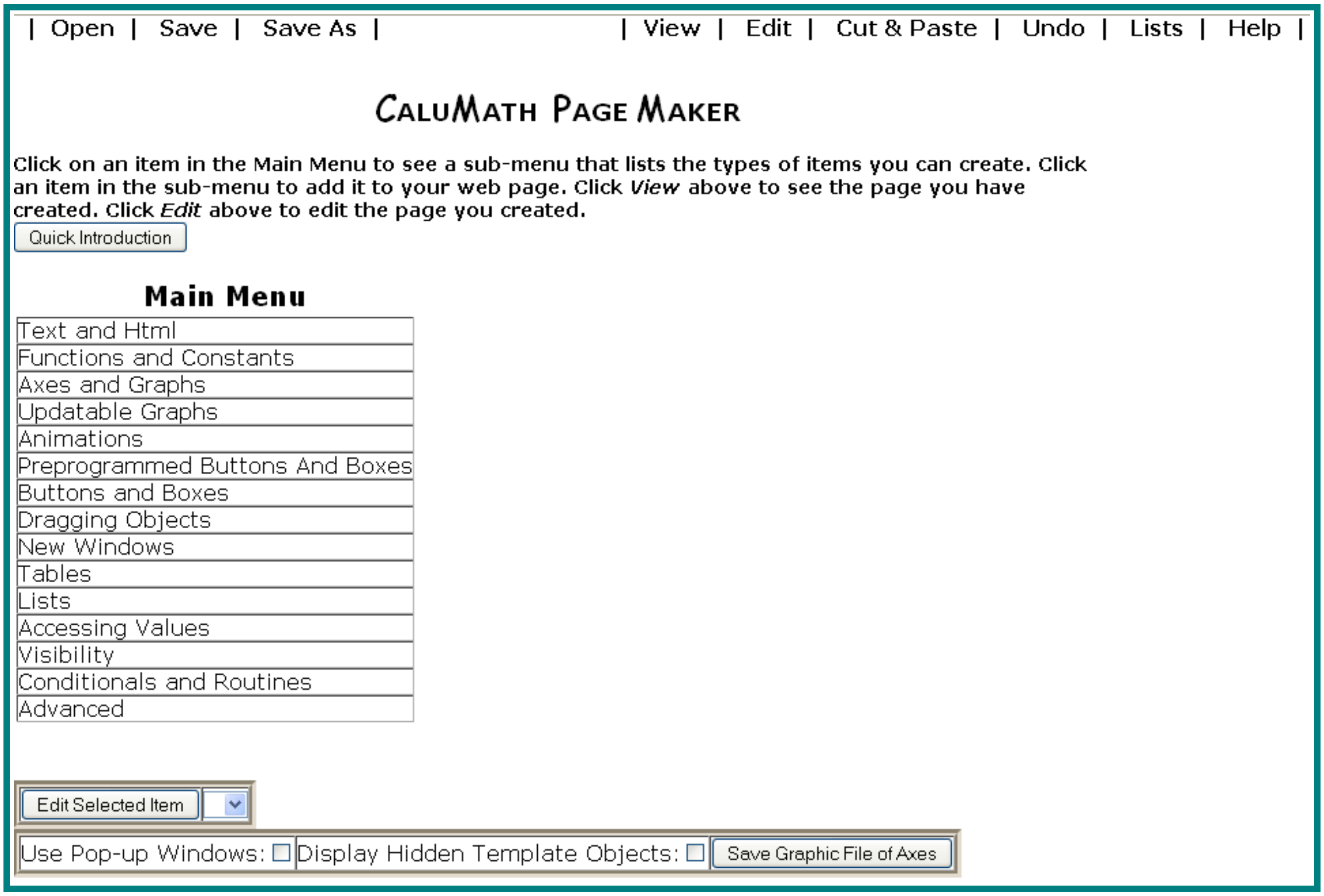
To construct a CaluMath web page, you need to download the CaluMath Page Maker to your computer. Go to the Beta version of CaluMath at
http://mathvserver.calumet.purdue.edu/~psturbek/CaluMath/CaluMath Beta Home Page.html
download the zip file to the desktop your computer, and unzip it there. This should created a folder named CaluMath on your desktop. Note that, once the CaluMath folder is created, it can be placed anywhere on your computer. The only restriction is that it cannot be placed in a folder that has CaluMath as part of its name.

Open the CaluMath folder on your computer, and use the Firefox web browser to open the file named CaluMath_Page_Maker.html that is inside the CaluMath folder. If you do not know how to open the file with Firefox, do the following: right click on the file, and in the menu that pops up, go down to Open With, and click on Firefox in the new small menu that pops up. If Firefox does not appear in the small menu, click on Choose Program and look for Firefox there. Although the CaluMath Page Maker can be opened in any web browser, Firefox provides helpful warnings which can often be used to find errors, for example, if you type sinn(pi) instead of sin(pi).

Once you have the CaluMath Page Maker opened, you should observe the following precautions:

1. Never use your browser's back or forward buttons. If you navigate away from the CaluMath Page Maker, you will lose any unsaved information.
2. If your web browser has a pop up blocker, it should be turned off. To do this in Firefox, go to Tools and then Options. Click the Content item and then make sure that Block pop-up windows is unchecked. After doing this, you may need to close your browser and then open it again for the changes to take place.

Below is a screen shot of the CaluMath Page Maker. I f you click on an item in the Main Menu, a sub-menu opens listing the items in that category that can be created. Clicking on an item takes you to a screen for the creation of that object. Each type of object has options pertaining to it from which you can select. When you are finished creating the object, you exit the screen and are returned to the main window of the CaluMath Page Maker with the Main Menu visible. After creating one or more objects, you can click the View item at the top of the page to see the page you have created. If you click the Edit item, you are taken to a screen that lists the items you have created; clicking on an item takes you back to the screen where the item was created. You can also edit an item by selecting it from the menu that appears below the Main Menu and clicking the Edit Selected Item button.

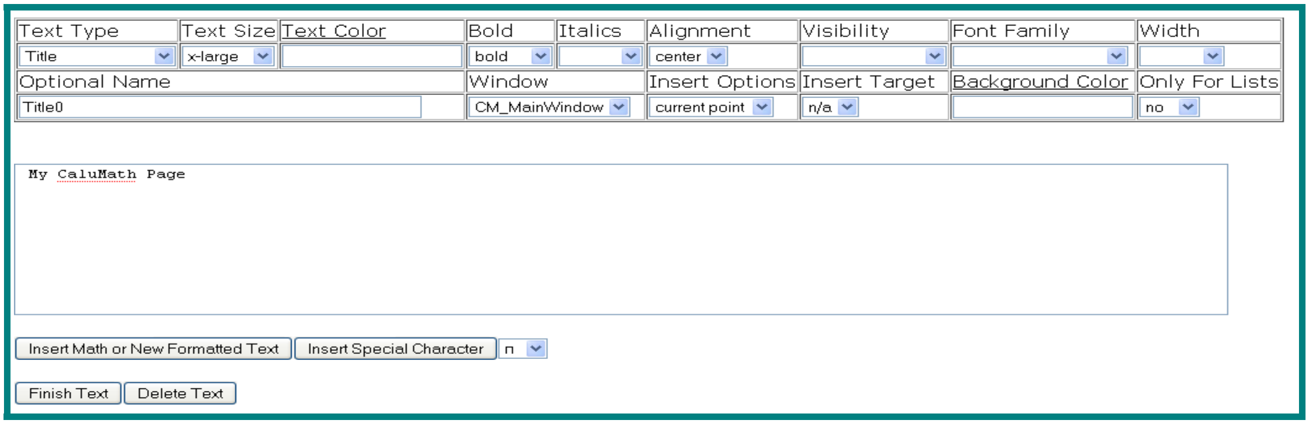

| Open | Save | Save As |
| View | Edit | Cut & Paste | Undo | Lists | Help |

CaluMath Page Maker

Click on an item in the Main Menu to see a sub-menu that lists the types of items you can create. Click an item in the sub-menu to add it to your web page. Click *View* above to see the page you have created. Click *Edit* above to edit the page you created.

We will begin by creating a title for our page, we will define several constants and functions, and create several graphs. This will give us a feel for how the CaluMath Page Maker functions and also allow us to discuss several key ideas that are important to keep in mind while constructing CaluMath web pages.

Title: To construct the title, click the Text and Html item in the Main Menu and click Text in the sub-menu that opens. Select *Title* in the Text Type menu, in the Text Size menu select *x-large*, in the Bold menu select *bold*, select *center* in the Alignment menu, and then type the text for your title in the large text area as is shown in the screen shot below. Click the Finish Text button, which returns you to the CaluMath Page Maker main window.

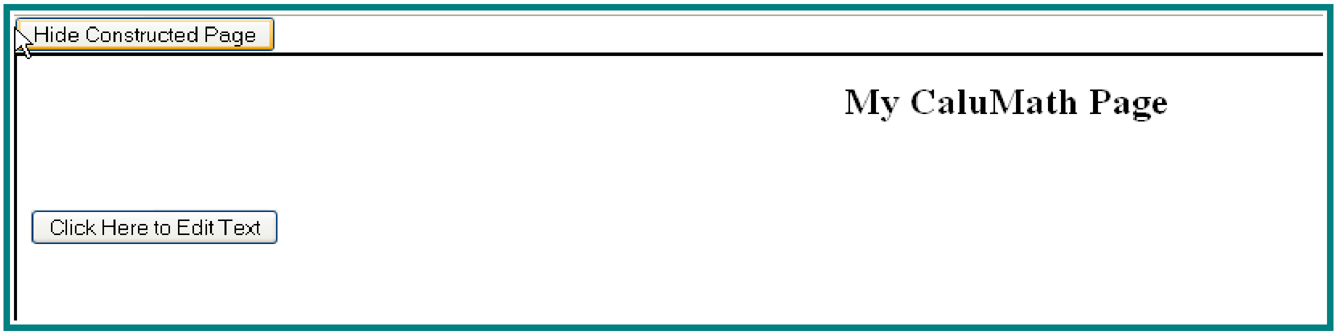


Text Type	Text Size	Text Color	Bold	Italics	Alignment	Visibility	Font Family	Width
Title	x-large		bold		center			

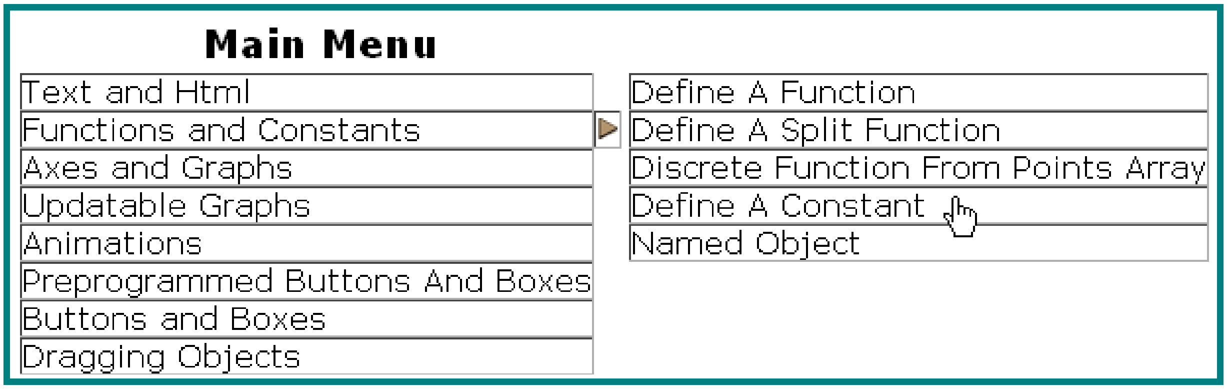
Optional Name	Window	Insert Options	Insert Target	Background Color	Only For Lists
Title0	CM_MainWindow	current point	n/a		no

```
My CaluMath Page
```

View Page: To view the page we have created so far, click the View item at the top of the page. The page we are creating will appear in a box, whose technical name is an iframe, that appears near the bottom of the CaluMath Page Maker. The page contains the title we are creating and an extra button labeled Click Here to Edit Text. This button only appears while you are constructing the page with the CaluMath Page Maker; after you save your page this button will not appear when you open it in a web browser. Click on the button now, and then click on the title and observe that it takes you back to the screen where you created the title. You can modify the title if you want; click the Finish Text button to return to the CaluMath main window.

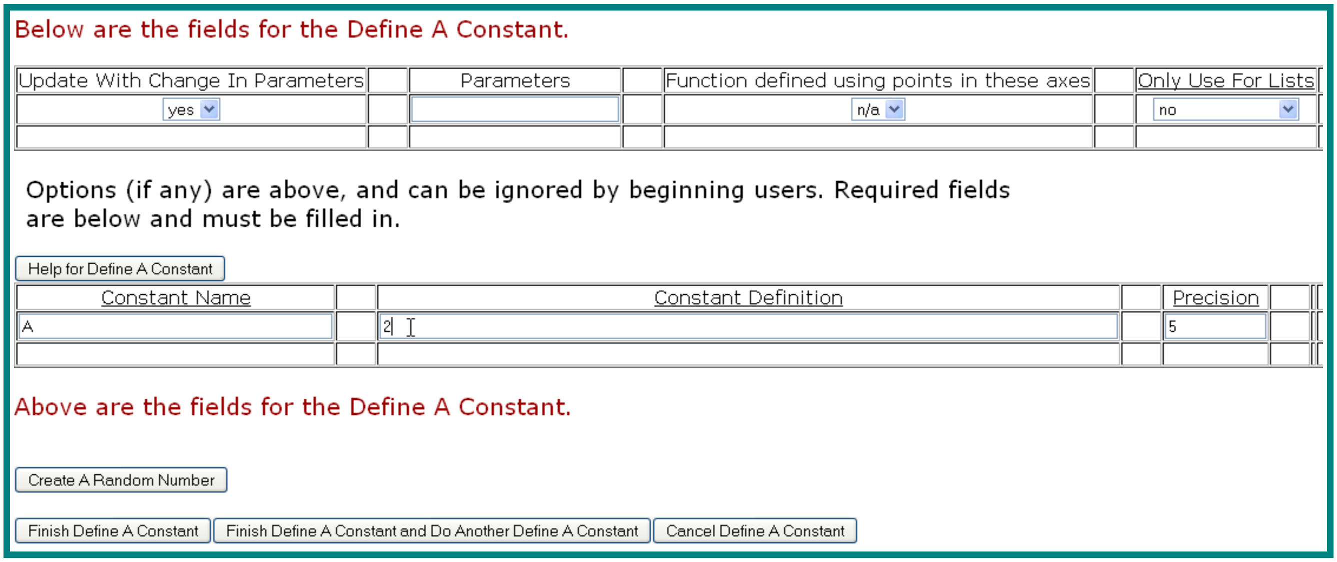


Constants: We will construct two constants, the constant A=2 and a constant B that takes on a random nonzero integer value between -4 and 4. Click Functions and Constants in the Main Menu and click Define a Constant in the sub-menu.



The screen that appears consists of two tables. The bottom table consists of required fields that must be filled out. The top table consists of options that can safely be ignored until you become more familiar with CaluMath. Between the two tables is a help button. The screens for the construction of almost all objects in CaluMath have a similar layout, with required fields in the bottom table, options in the top table and a help button in between the two tables.

In the Constant Name field, enter *A*. In the Constant Definition field, enter *2*. Since we want to also create the constant B, instead of clicking the Finish Define a Constant button, click the Finish Define a Constant and Do Another Define a Constant button. This saves the constant A that we just created and allows us to define another constant. A pop up window will tell you that you finished one constant and can now create another one. Close the pop up window. Note the CaluMath Page Maker leaves all of the fields as they were; this is helpful if you want to create several constants with similar properties. Note however, that an asterisk is put next to the name A; this forces us to change the name.


Help for Define A Constant
Create A Random Number
Finish Define A Constant
Finish Define A Constant and Do Another Define A Constant
Cancel Define A Constant

Below are the fields for the Define A Constant.

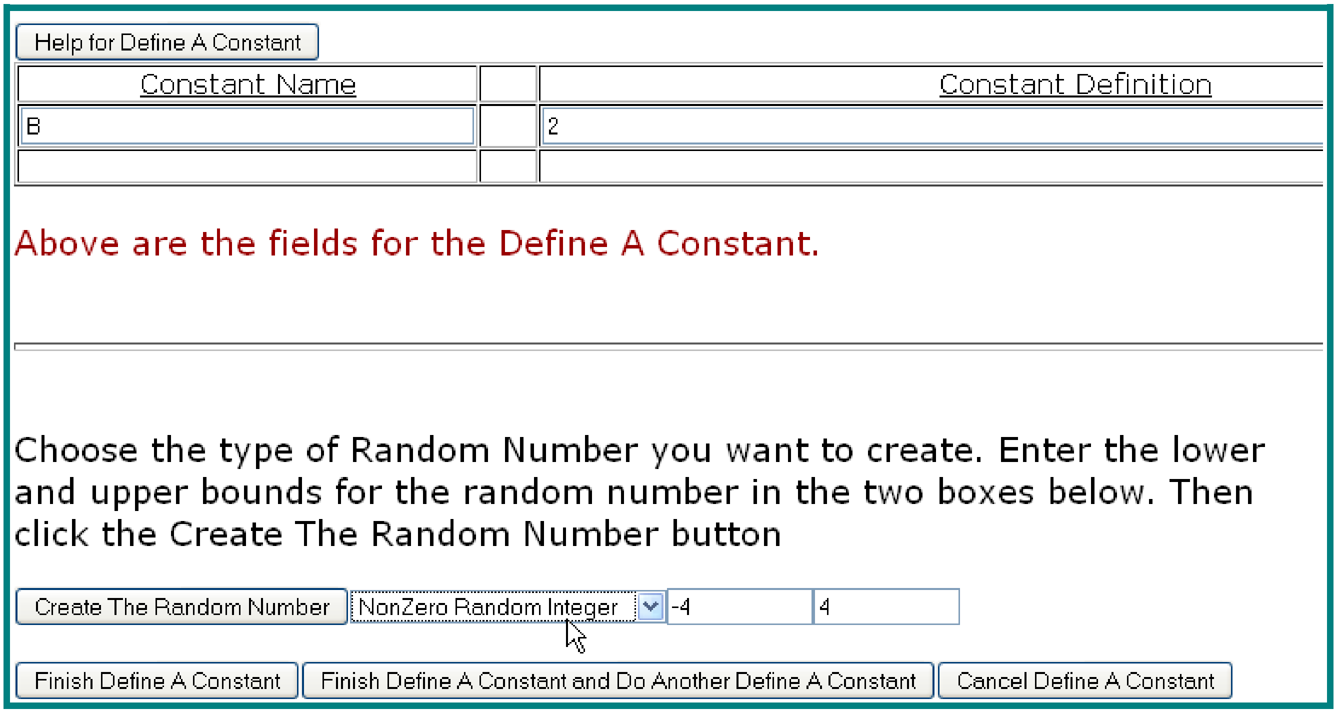
Update With Change In Parameters	Parameters	Function defined using points in these axes	Only Use For Lists
yes		n/a	no

Options (if any) are above, and can be ignored by beginning users. Required fields are below and must be filled in.

Constant Name	Constant Definition	Precision
A	2	5

Above are the fields for the Define A Constant.

Change the A* in the Constant Name field to B. Click the Create A Random Number button. This opens a new menu that is shown in the screen shot below. Select NonZero Random Integer and enter -4 and 4 for the range of the integer (both -4 and 4 will be included as possible values for the integer). **Click the Create the Random Number button**. This places the correct CaluMath code for the random number in the Constant Definition field and hides the Random Number menu. Click the Finish Define A Constant button to return to the CaluMath Page Maker Main window.


Help for Define A Constant
Create The Random Number
Finish Define A Constant
Finish Define A Constant and Do Another Define A Constant
Cancel Define A Constant

Constant Name	Constant Definition
B	2

Above are the fields for the Define A Constant.

Choose the type of Random Number you want to create. Enter the lower and upper bounds for the random number in the two boxes below. Then click the Create The Random Number button

NonZero Random Integer | -4 | 4

If we click the View item at the top of the CaluMath Page Maker, we only see the title of our page. Although we defined two constants, we have not done anything with them in the page. We will now create a paragraph that displays the values of A and B. This is done exactly the same way we created a title: Click on the Text and Html item in the Main Menu, and click the Text item in the sub-menu. In Text Type, select *Paragraph*. In the large text box, type the following:

A=cm_evalm(A), B=cm_evalm(B), and Unicode(pi)= cm_evalm(CM_Round(pi,3)).

Text and Number Fields: Let us discuss the above entry for a moment. In CaluMath, you should keep in mind whether you are typing something into a field that expects text, or a field that expects numbers and mathematical expressions yielding numbers. Fields that expect text do not evaluate your input in any way, they merely copy what you write. For example, if you type *This is pi*, a text field does not try to determine what you mean by *This* or *is* or *pi*. If you want a mathematical expression evaluated in a text field, you must enclose the expression in parentheses and enter cm_evalm in front of the parentheses. This will force the enclosed item to be mathematically evaluated. **You can only use this for mathematical expressions that evaluate to numbers**. Therefore the above text will be evaluated as follows:

1. A=cm_evalm(A) will produce the text *A*= and then display the value of A in place of cm_evalm(A).
2. B=cm_evalm(B) will produce the text *B*= and then display the value of B in place of cm_evalm(B).
3. Unicode(pi)= cm_evalm(CM_Round(pi,3)) does the following. Unicode(pi) tells CaluMath to replace pi by the Greek letter π. This can be done for any Greek letter, for example Unicode(delta) and Unicode(Delta) produce δ and Δ respectively. CM_Round is the CaluMath rounding command, you can use it to round an expression to a given number of decimal places (from 0 to 12 decimal places). Therefore CM_Round(pi,3) rounds π to 3 decimal places, in order to get the value 3.142 to appear in the page, you must enclose it inside the parentheses in cm_evalm(). Incidentally, instead of entering the number of decimal places, you can also enter either .25 or .5; these values round the expression to the nearest quarter or half respectively. These are particularly useful when dragging graphs (which will be discussed later). Therefore CM_Round(pi, .25) yields 3.25 (rounded to the nearest fourth) and CM_Round(pi, .50) yields 3 (rounded to the nearest half).

In CaluMath, you may also define objects that are not numbers, in this case you should use cm_eval() to evaluate them; the difference is that cm_eval() evaluates the object, while cm_evalm() evaluates the object mathematically to a number. For example, cm_evalm(pi) yields the number 3.14..., however cm_eval(pi) produces an error, since pi itself has no meaning in CaluMath.

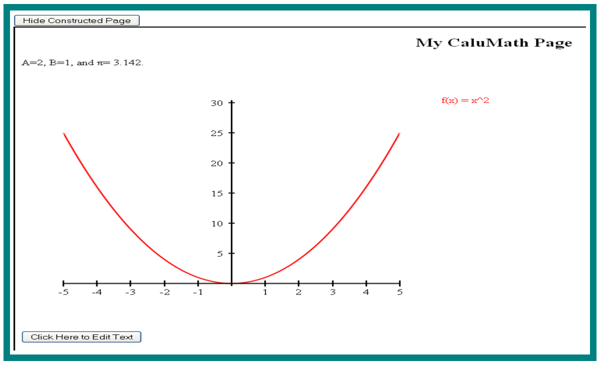
Note that cm_eval() and cm_evalm() can be used in any text field when you want something evaluated. You never need to use them in a field that expects numbers. CM_Round is one of several mathematical commands; these commands can be used in any field that expects numbers. **In general, all commands and special expressions in CaluMath have either the prefix *cm_* or the prefix *CM_*.**

Click the Finish Text button to save the paragraph and return to the CaluMath Page Maker main window. Click the View item at the top of the page to view the page we have created. You should see the title along with a paragraph that says *A=2, B=4, and π= 3.142,* although the value of B will depend on the random integer created by CaluMath. At the top left of the constructed page, you should see a Hide Constructed Page button. Click on it to return to the main window. If you click the View item again, a new page will open which will probably have a different value of B defined. You can click the View item and the Hide Constructed Page button several times to see the different values of B produced.

Functions: We now define the functions f(x) = x^2 and g(x) = B*x^2. In the Main Menu, click Functions and Constants and click Define a Function in the sub-menu. Again, two tables appear, the lower table contains required fields and the upper table contains options. The help button yields extensive help, since you can use the options table to create very powerful, and easy to use, functions (which we will see in a few minutes). For now, enter *f* in Function Name, *x* in Function Variable, and *x^2* in Function Definition. Click the Finish Define A Function And Do Another Define A Function button. This saves our definition of the function f and allows us to create another function. In the new screen that appears, replace the *f** in Function Name with *g*, and enter *B*x^2* in Function Definition. Note that you must use a * to represent multiplication. Although we do not need them here, keep in mind that you should liberally use parentheses to ensure your expressions are interpreted correctly. Click the Finish Define A Function Button.

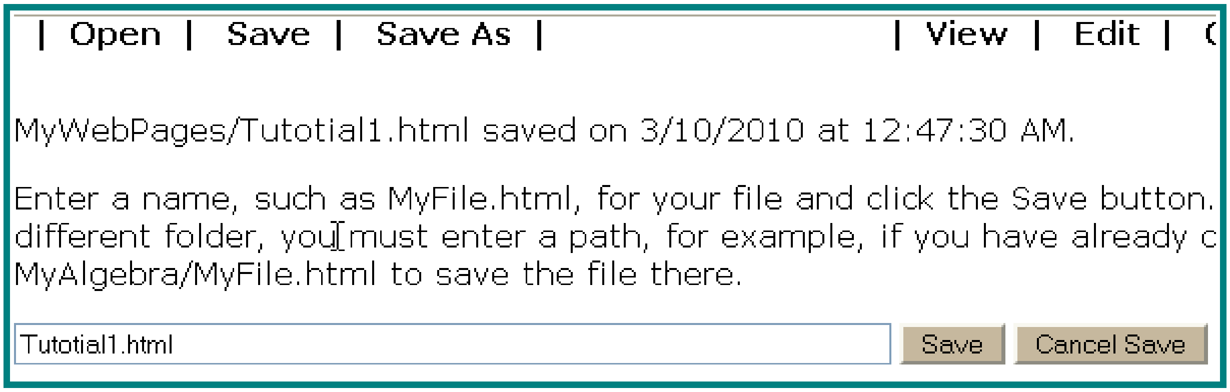
Axes: We now define a set of axes. In the Main Menu, click on Axes and Graphs and click on Draw Axes in the sub-menu. There are a large number of options for axes, as evidenced by the large table above the Help for Draw Axes button. For the moment, we will ignore all of the options. Almost every object in CaluMath must have a name; so enter *Axes1* in the Axes Name field. You can leave the default values of *6* and *5* for Plot Width and Plot Height, this corresponds to 6*72=432 and 5*72=360 pixels on your computer screen, or 6 inches by 5 inches if you print the axes on a piece of paper. Alternately, you can enter the number of pixels directly into the Plot Width and Plot Height fields (CaluMath will interpret any large number in these fields as being in pixels). Let us change the Beginning and Ending X values to *-5* and *5* respectively. We can leave the beginning and ending Y values with the word *default*; this will allow the axes to automatically adjust to the graphs we plot. Click the Finish Draw Axes button.

Graphs: We now graph the function f(x) = x^2. In the Main Menu, click Axes and Graphs and click Graph of a Function in the sub-menu. In the screen that opens, menus will list all of the axes you have constructed and all of the functions you have defined. In the Axes field, select *Axes1* (there will be no other choice, since we have not created other axes) and in the Function field, select *f*. Leave the word *default* in the Beginning and Ending X Value fields, this will cause the graph of f to begin and end at the beginning and ending x values of the axes (which we had defined to be from -5 to 5 above). In the options field, note that Plot Style defaults to *normal* (instead of thick or thin), and Display Equation defaults to *yes*. This latter value means that the equation f(x) = x^2 will be displayed to the right of the axes when the graph is constructed. Click the Finish Graph Of A Function button. Upon returning to the main window, click the View item to view our page. It should look like the screen shot below.


My CaluMath Page

A=2, B=1, and π= 3.142.

Saving Your Page: Click the Hide Constructed Page button to return to the main window. At the top left, click the Save item. Enter the name Tutorial.html in the text box. Note that all file names should have the extension .html to indicate that they are web pages. The file will be saved in the MyWebPages folder inside the CaluMath directory.

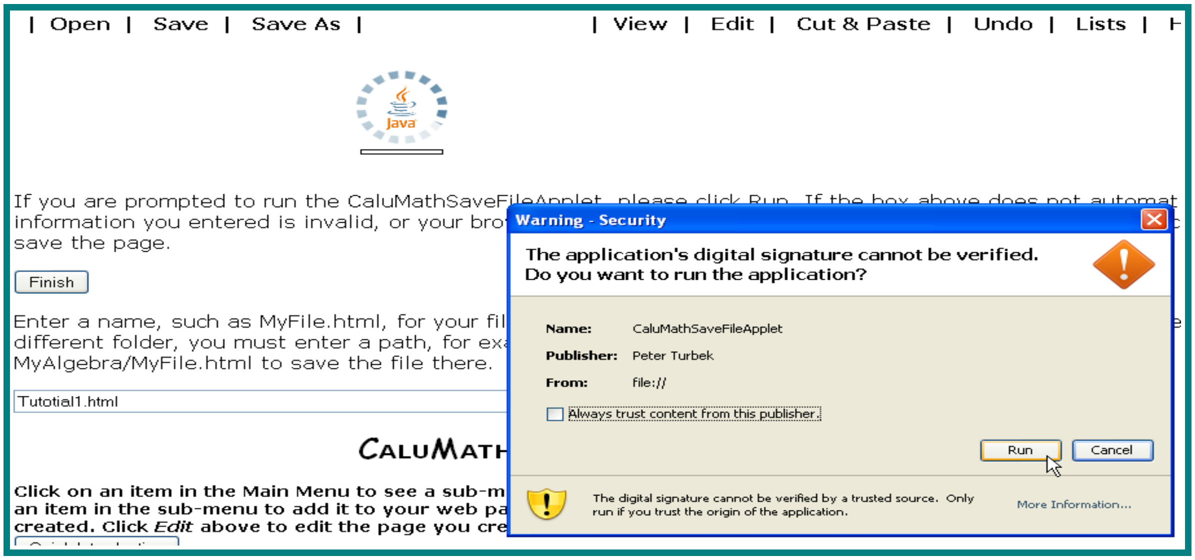


MyWebPages/Tutotial1.html saved on 3/10/2010 at 12:47:30 AM.

Enter a name, such as MyFile.html, for your file and click the Save button.
different folder, you must enter a path, for example, if you have already c
MyAlgebra/MyFile.html to save the file there.

Tutotial1.html

When you click Save, a Java applet will appear, asking for your permission for the web page to save the file to your hard drive. Click Run to run the applet and save the file.


| Open | Save | Save As | | View | Edit | Cut & Paste | Undo | Lists | H
Java
Finish
CALUMATH
Warning - Security
Run
Cancel
More Information...

If you are prompted to run the CaluMathSaveFileApplet, please click Run. If the box above does not automat
information you entered is invalid, or your bro
save the page.

Enter a name, such as MyFile.html, for your fil
different folder, you must enter a path, for exa
MyAlgebra/MyFile.html to save the file there.

Tutotial1.html

Click on an item in the Main Menu to see a sub-m
an item in the sub-menu to add it to your web pa
created. Click *Edit* above to edit the page you cre

The application's digital signature cannot be verified.
Do you want to run the application?

Name: CaluMathSaveFileApplet
Publisher: Peter Turbek
From: file://

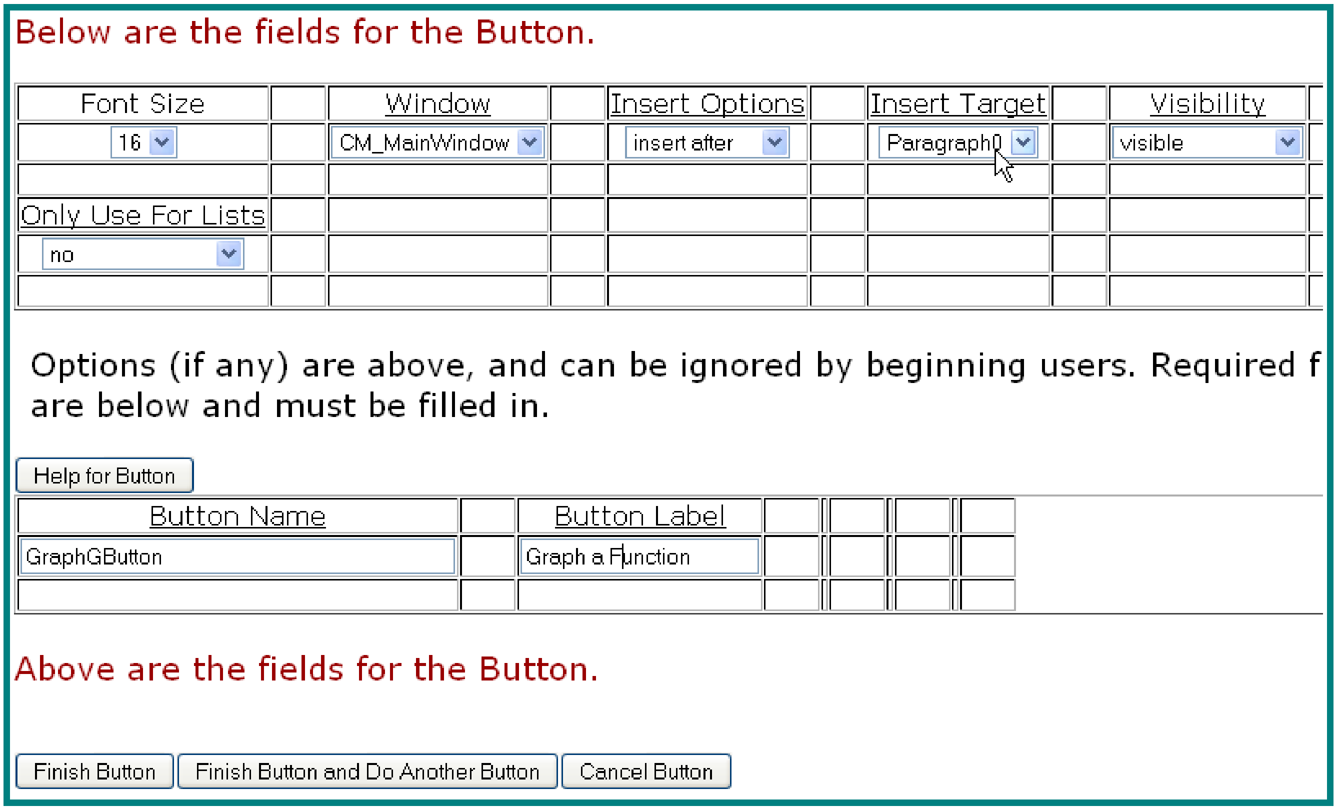
☐ Always trust content from this publisher.

The digital signature cannot be verified by a trusted source. Only run if you trust the origin of the application.

If you now minimize the CaluMath Page Maker and go to the MyWebPages folder inside the CaluMath directory, you should see the file Tutorial.html. If you open it in a web browser, you will see the page you have created. Note that the Click Here To Edit Text button, which was visible when the page was viewed in the CaluMath Page Maker, is no longer visible.

Buttons: We will now construct a button, and use this to highlight some general CaluMath features that apply to many Html objects in addition to buttons. When the user clicks on the button we construct, the graph of g(x) = B*x^2 will appear on the axes. In the Main Menu, click Buttons and Boxes and click Button in the sub-menu. Almost every object in CaluMath must have a name, it helps if the name for the button is descriptive of what the button does. In Button Name enter *GraphGButton*. **Note that**

names in CaluMath cannot contain spaces, must begin with a letter, and can only contain letters, numbers, and the underscore character. The Button Label is the text that appears on the button; enter *Graph a Function*. All Html items, such as paragraphs, buttons and boxes can be placed anywhere you choose. For practice in doing this, we will place the button so that it appears immediately after the paragraph we created (and before the axes we constructed). To accomplish this, we use the Window, Insert Options, and Insert Target fields. The only window we have is the page we are constructing, which automatically has the name CM_MainWindow (if the page we were constructing opened new windows, the names of these windows would appear in the Window field). In Insert Target, select *Paragraph0*, which is the name for the paragraph we constructed. In Insert Options, select *insert after* so that the button we create will appear after Paragraph0. When you are done, click the Finish Button button. Below is a screen shot for the entries in the Button screen.



Below are the fields for the Button.

Font Size	Window	Insert Options	Insert Target	Visibility
16	CM_MainWindow	insert after	Paragraph0	visible

Only Use For Lists
no

Options (if any) are above, and can be ignored by beginning users. Required f
are below and must be filled in.

Button Name	Button Label
GraphGButton	Graph a Function

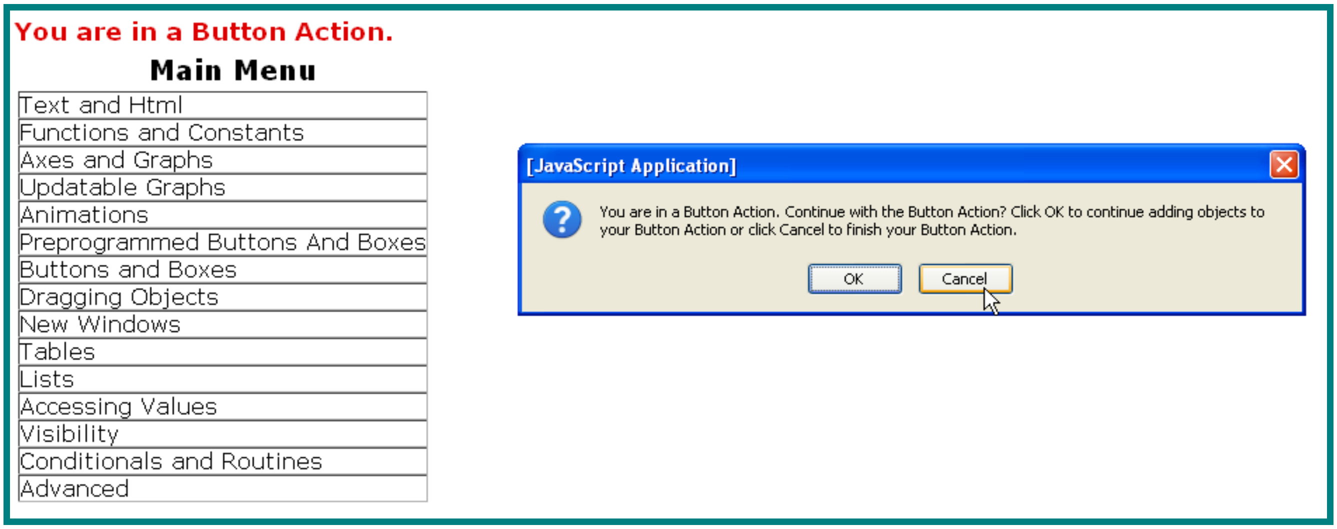
Above are the fields for the Button.

Click the View item to see the page. Note that the Graph a Function button appears between Paragraph0 and the axes. If you click on the button nothing happens. We now give the button instructions that it will execute when it is clicked.

Button Action: We now construct a Button Action. You can think of a Button Action as a container; inside the container you place all of the items that you want executed when the button is clicked. In the Main Menu, click Buttons and Boxes and click Button Action in the sub-menu.

We will use the creation of this Button Action as an opportunity to discuss naming conventions in CaluMath. Recall above that we defined the function Axes1 and we defined the function f. We also constructed the graph of f in Axes1. CaluMath automatically gives the graph the name Axes1.f. This is true for all graphical items we construct in Axes1. If we construct a point named Point1 in Axes1, then its true name is Axes1.Point1. It is important to understand this because that is the way it will be listed in some menus in the CaluMath Page Maker. The button GraphGButton that we constructed above does not belong to the axes Axes1 since it doesn't appear inside the axes. Instead the GraphGButton button belongs to an object called the CM_ParentObject and its true name is CM_ParentObject.GraphGButton. Most non-text items that do not belong to a set of axes automatically belong to the CM_ParentObject. The only reason you need to be aware of this is that some menus will display the true names of objects in the CaluMath Page Maker; in particular, that is the way the GraphGButton button appears in the Button menu in the Button Action screen. Select *CM_ParentObject.GraphGButton* (since we created no other buttons, this is the only choice). In the Timing of Event menu, select *When Button is Clicked*; you will not be able to select the *When Graph is Clicked* entry because of the type of button we are creating. Click the Finish Button Action button.

A pop up window appears which states: *Now construct the item you want to add to your Button Action*. You can close this window. Note that You are in a Button Action appears in red above the Main Menu. This reminds you that everything you now construct will be put in the Button Action container and executed when the user clicks the button. For now, the only thing we want to do is graph the function g(x) = B*x^2. So in the Main Menu, click Axes and Graphs and click Graph of a Function in the sub-menu. In the screen that opens, select *Axes1* and in the Function field, select *g*. Leave the word *default* in the Beginning and Ending X Value fields, this will cause the graph of f to begin and end at the beginning and ending x values of the axes. Click the Finish Graph Of A Function button. When you do, a dialog box opens, asking if you want to continue adding items to your Button Action container, or finish your button action. Click Cancel to stop adding items to your button action (if we wanted to continue adding more items we would click OK). This returns us to the main window. If you mistakenly click OK when you do not want to add more items to your Button Action container, you can click on Buttons and Boxes in the main menu and End Button Action, Conditional, Etc. in the sub-menu in order to exit the Button Action.



You are in a Button Action. Continue with the Button Action? Click OK to continue adding objects to your Button Action or click Cancel to finish your Button Action.

Click the View item to view your page. Click on the Graph a Function button to display the graph of g(x) = B*x^2. If you click the button repeatedly, you will see that the the page creates multiple copies of the graph of g (if B>0) which does not seem useful. This highlights one issue of CaluMath web page construction that you should keep in mind. If you only want a user to click on a button once, the Button Action for that button should include the instruction to hide the button after they click it so it will no

longer be visible (and so cannot be clicked again).
Instead of doing that, we will add more items to the Button Action to make the button more useful. We will do the following things:

1. Hide Paragraph0 to gain practice in hiding objects.
2. Redefine B to be a new Random Integer
3. Remove the old graph of g when we construct a new one.

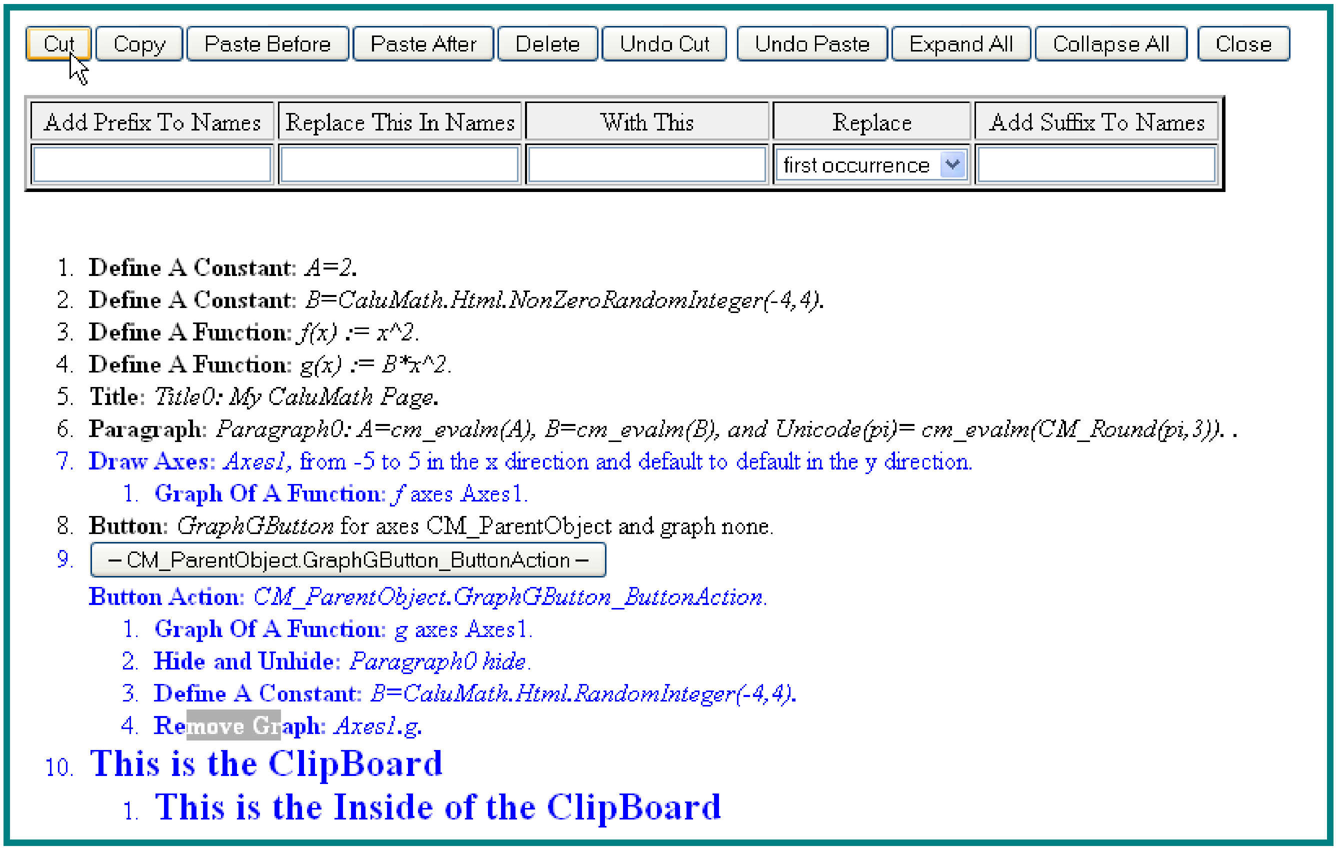
To do this, click Buttons and Boxes from the Main Menu and click Add To Button Action from the sub-menu. Select CM_ParentObject.GraphGButton_ButtonAction from the Button Action menu. A pop up window instructs you to construct the item you want to add to the Button Action. Again You are in a Button Action appears above the Main Menu.

1. To hide Paragraph0, click the Visibility item in the Main Menu and click Hide and Unhide in the sub-menu. In the Object field, select *Paragraph0* and in the Visibility field, select *invisible*. Click the Finish Hide and Unhide button. You will again be prompted with the dialog box asking if you want to add more items to the Button Action container. Click OK, since we want to construct items 2) and 3) above. You can close the pop up window that says *Add another item to your Button Action* when it appears.
2. To redefine B, click Functions and Constants from the Main Menu and click Define A Constant in the sub-menu. We repeat our construction of B from above: Enter *B* in the Constant Name field. Click the Create A Random Number button. Select *NonZero Random Integer* and enter *-4* and *4* for the range of the integer. **Click the Create the Random Number button,** which places the correct CaluMath code for the random number in the Constant Definition field, and click the Finish Define A Constant button. You will again be prompted with the dialog box asking if you want to add more items to the Button Action container. Click OK, since we want to construct item 3) above. You can close the pop up window that says *Add another item to your Button Action* when it appears.
3. To remove the old graph of g, click the Visibility item in the Main Menu. We should remove the old graphs of g, and not just hide them, so only one graph of g exists at a time. Therefore click Remove Graph in the sub-menu. In the Graph menu, select Axes1.g and then click the Finish Remove Graph button. Since we do not want to add anything else to the Button Action, click Cancel in the dialog box that opens. Close the pop up window that says *You have finished your Button Action*.

Cut and Paste: There is one problem with our Button Action. It contains four items that will be executed in following order:

1. Graph g.
2. Hide Paragraph0.
3. Redefine B to a new Random Integer.
4. Remove the graph of g.

Therefore, the graph of g will be created and immediately removed. To fix this, we must remove the old graph of g before we construct a new graph of g. To do this, we will Cut and Paste the remove the graph of g item and paste it at the beginning of the Button Action. At the top right of the CaluMath Page Maker main window, click the Cut & Paste item. A window opens that displays every item you have created. The Button Action contains several items, click the button corresponding to the Button Action item (which shou*ld be item 9)* to reveal the contents of the Button Action. We want to cut the *Remove Graph: Axes1.g* item and place it before the *Graph Of A Function: g axes Axes1* item. To do this, place your cursor anywhere on the *Remove Graph: Axes1.g* item and drag it a small way across the item as is shown in the screen shot below. It is better if you only drag it a small amount and do not try to start at the beginning of the line and go all the way across. After highlighting a small portion of the that line, go to the top of the menu and click the Cut button. You will see the Remove Graph: Axes1.g entry disappear. Now click the line that says *Graph Of A Function: g axes Axes1* which is where you want to place it, and go top the top and click the Paste Before button. This tells the CaluMath Page Maker to take the item you cut and paste it before the item on which you clicked.

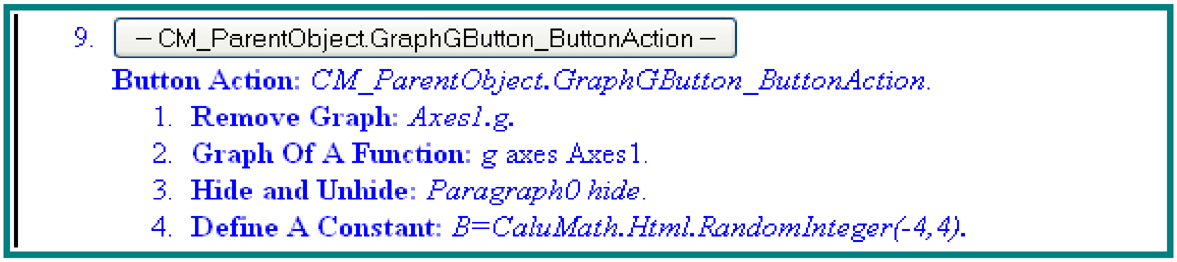


Add Prefix To Names	Replace This In Names	With This	Replace	Add Suffix To Names
			first occurrence	

1. **Define A Constant**: *A=2.*
2. **Define A Constant**: *B=CaluMath.Html.NonZeroRandomInteger(-4,4).*
3. **Define A Function**: *f(x) := x^2.*
4. **Define A Function**: *g(x) := B*x^2.*
5. **Title**: *Title0: My CaluMath Page.*
6. **Paragraph**: *Paragraph0: A=cm_evalm(A), B=cm_evalm(B), and Unicode(pi)= cm_evalm(CM_Round(pi,3)).* .
7. **Draw Axes**: *Axes1*, from -5 to 5 in the x direction and default to default in the y direction.
    1. **Graph Of A Function**: *f* axes Axes1.
8. **Button**: *GraphGButton* for axes CM_ParentObject and graph none.
9. – CM_ParentObject.GraphGButton_ButtonAction –

   Button Action: *CM_ParentObject.GraphGButton_ButtonAction.*
    1. **Graph Of A Function**: g axes Axes1.
    2. **Hide and Unhide**: *Paragraph0 hide.*
    3. **Define A Constant**: *B=CaluMath.Html.RandomInteger(-4,4).*
    4. **Remove Graph**: *Axes1.g.*
10. **This is the ClipBoard**
    1. **This is the Inside of the ClipBoard**

The Cut & Paste window will reload, and a dialog will ask if you want to paste the cut item again in a new position. Since we do not want to do this, click Cancel. When you click the button corresponding to the Button Action to reveal its contents, you will see them listed in the order we want.



9. – CM_ParentObject.GraphGButton_ButtonAction –

   Button Action: *CM_ParentObject.GraphGButton_ButtonAction.*
    1. **Remove Graph**: *Axes1.g.*
    2. **Graph Of A Function**: g axes Axes1.
    3. **Hide and Unhide**: *Paragraph0 hide.*
    4. **Define A Constant**: *B=CaluMath.Html.RandomInteger(-4,4).*

You may realize that there is a potential problem with the Button Action. The very first time the user clicks on the GraphGButton, the Button Action will try to remove the graph of g; however the first time the user clicks on the button, there is no graph of g to remove. Fortunately, CaluMath checks whether objects exist before it tries to remove them; if the object to be removed does not exist, CaluMath recognizes this fact and does nothing and does not cause an error to be produced.

Click the Hide Constructed Page button at the top left to exit from the Cut & Paste window and return to the CaluMath Page Maker main window. Click View to observe your page. As you click the Graph a Function button, you will see graphs drawn in different colors as the values of B changes. However, all is not well with the graph. Sometimes the graph of g is invisible, this is because the axes are scaled to the graph of f that originally appeared in the axes; negative y values (which occur when B is negative) do not appear on the graph. In addition, the equation to the right of the axes always says g(x) = B*x^2, which doesn't give us an idea of the value of B. To address these issues, we will discuss editing our page.

Editing the Page: There are two ways to edit the page, and we will illustrate both of them now. From the CaluMath Page Maker main window, click the Edit item at the top right. A window similar to the Cut & Paste window opens. As before, you can click on the button corresponding to the Button Action to see the items it contains. To edit an item, you simply click on the line corresponding to it; this will immediately take you back to the screen where it was created. Go to line 7 which begins with *Draw Axes: Axes1*, and click somewhere near the middle of the line. You will immediately be taken to the screen where the axes were constructed. In Beginning Y Value enter *-100* and in Ending Y Value enter *100*. This will give us a good view of the graph of g(x) = B*x^2 when B takes on values from -4 to 4. Click the Finish Draw Axes button to return to the main window.

We will now edit the graph of g by using the drop down menu below the Main Menu. Editing using this menu is generally faster than using the Edit item at the top of the main window, because the Edit item requires the Edit window to load, which can cause a small delay if the page is large. The items in the Edit drop down menu are organized in the same way as those in the Edit window, in fact, they have the same numbering system. Note that Axes1 is item 7, and you can see that the graph of f is a child of the axes. Similarly, the Button Action is item 9, and you can visually see the four items it contains. Select the graph of g (which should be item 9.2), and click the Edit Selected Item button. This returns us to the screen where we graphed g. We will use the Alternate Displayed Equation option to change how the graph of g is displayed. There is also an explanation of this option in the Help for Graph Of A Function button. Whatever we enter in this field will appear instead of the text that normally appears to the right of the axes for the graph of g. The normal text that appears is g(x) = B*x^2, and we consider the g(x) to be the left hand side of the equation and the B*(x^2) to be the right hand side. In Alternate Displayed Equation field, if we type cm_lhs, it will be replaced by the left hand side of the text that normally appears (meaning g(x)). If we type cm_rhs, it will be replaced by the right hand side of what normally appears (meaning B*x^2). If we type cm_color, it will be replaced by the color of the graph. In the Alternate Displayed Equation field, we will enter *cm_lhs = cm_evalm(B)*x^2*. When it is displayed, cm_lhs will be replaced by g(x) and the B will be mathematically evaluated. Therefore the result will be text such as g(x) = 2*x^2 if B were 2. If we entered *The cm_color graph is cm_evalm(B)*x^2*, as we repeatedly clicked the Graph a Function button, the text displayed would be similar to *The blue graph is 3*x^2*. Click the Finish Graph of A Function button to return to the main window. Click View to view your page and repeatedly click the Graph a Function button to view the graphs of g for random values of B.

Save Your Page: At this point you should save your page by clicking the Save item at the top left of the main window.

We now begin the construction of the web page concerning graphs and their derivatives that was described on page 4. Keep in mind that we are using this page as a vehicle to introduce you to the main objects in CaluMath, how they are constructed, and concepts to keep in mind during their construction.

Container Like Items: There are several other items that are similar to Button Actions in that they contain items that are to be constructed together. They are Routines, For Loops, and Conditionals. We will discuss them later in the tutorial.

The Clipboard: Since the GraphGButton has served its purpose of giving you experience with the construction of buttons and Button Actions, we will now remove the GraphGButton from the page using the Cut & Paste menu. Click on the Cut & Paste item at the top right of the main window.
The Cut & Paste screen will load. Notice that the last item listed (which is number 10) is the following:

10. This is the ClipBoard

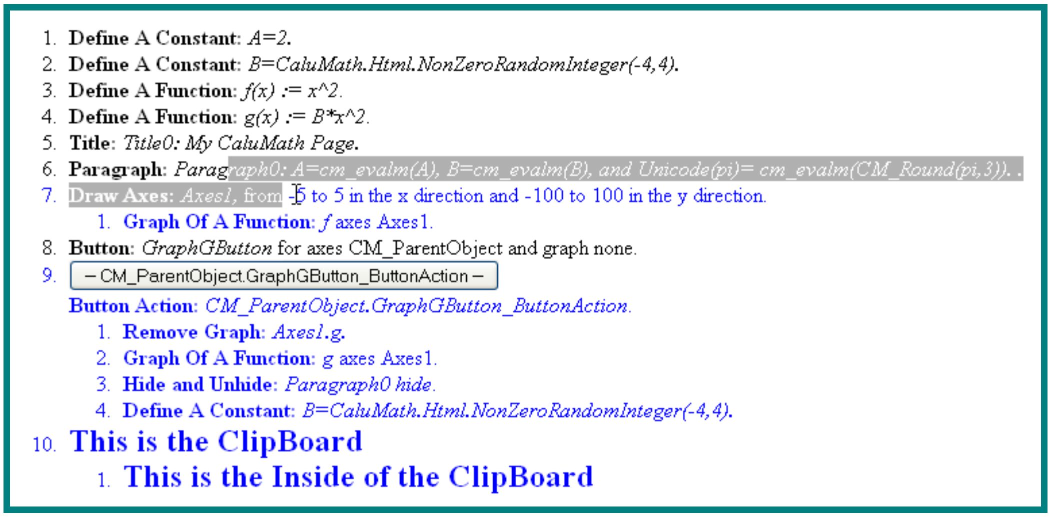
  1. This is the Inside of the ClipBoard

The Clipboard is an area where you can cut and paste items that you do not need, but which you may be interested in modifying and placing back in your web page at a later time. Items in the Clipboard do not appear in the web page, they instead are in a storage area and are invisible. If you save a page, the items in the Clipboard are saved with the page so they can be accessed the next time you load the page into the CaluMath Page Maker. We do not need the GraphGButton or the Button Action corresponding to it, since our sole purpose in creating it was to help you learn about buttons. We will cut and paste the button and the Button Action to the clipboard. There are several things to keep in mind when you are cutting and pasting. After stating these items we will illustrate them with examples below :

1. If you move a parent object, all of the children are automatically moved with it. For example, if we cut Axes1 and placed it in a different location in the document, then the graph of f would automatically be moved also. If you cut a Button Action and place it in the Clipboard, all of the items contained in the Button Action are automatically moved with it.
2. The beginning and ending items that you highlight during cutting and pasting must have the same parent, or they each must be top level elements in the web page. We will illustrate this below.
3. Items, such as Button Actions, that are actually containers containing objects have buttons associated with them in the cut and paste window. If you click on the button, the first line that is revealed says the type of item it is and below it are listed the children inside the container. To highlight a section of the document that begins with a Button Action (or other container-type item), you must click the button to reveal the items in the container. Highlighting the first line (that indicates the type of container it is) signifies that you are highlighting the entire container.

Here are some examples:

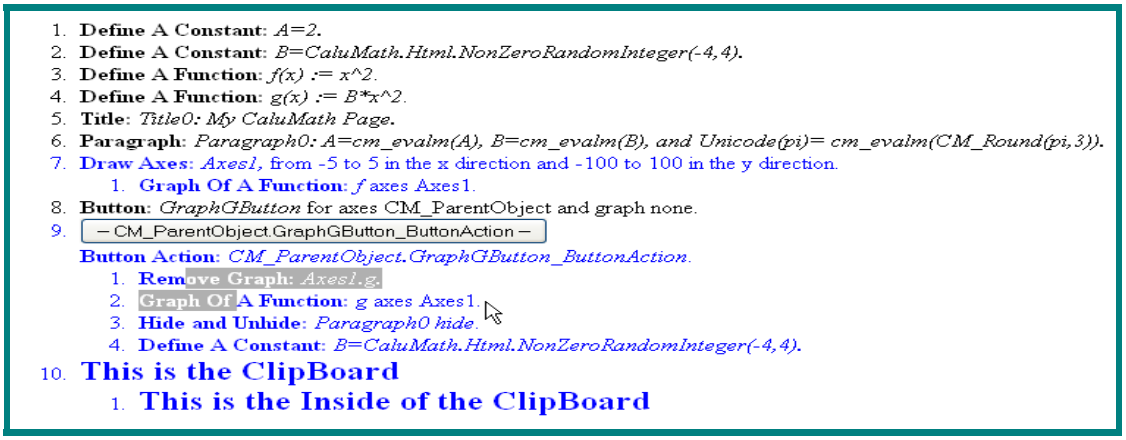
Below is an acceptable region to highlight for cutting and pasting, since both Paragraph0 and Axes1 are top level items in the page. Note that the graph g will automatically be included, since it is a child of Axes1.



1. **Define A Constant**: *A=2.*
2. **Define A Constant**: *B=CaluMath.Html.NonZeroRandomInteger(-4,4).*
3. **Define A Function**: *f(x) := x^2.*
4. **Define A Function**: *g(x) := B*x^2.*
5. **Title**: *Title0: My CaluMath Page.*
6. **Paragraph**: *Paragraph0: A=cm_evalm(A), B=cm_evalm(B), and Unicode(pi)= cm_evalm(CM_Round(pi,3)). .*
7. **Draw Axes**: *Axes1,* from -5 to 5 in the x direction and -100 to 100 in the y direction.
    1. **Graph Of A Function**: *f* axes Axes1.
8. **Button**: *GraphGButton* for axes CM_ParentObject and graph none.
9. [– CM_ParentObject.GraphGButton_ButtonAction –]

    Button Action: *CM_ParentObject.GraphGButton_ButtonAction.*
    1. **Remove Graph**: *Axes1.g.*
    2. **Graph Of A Function**: g axes Axes1.
    3. **Hide and Unhide**: *Paragraph0 hide.*
    4. **Define A Constant**: *B=CaluMath.Html.NonZeroRandomInteger(-4,4).*
10. **This is the ClipBoard**
    1. **This is the Inside of the ClipBoard**

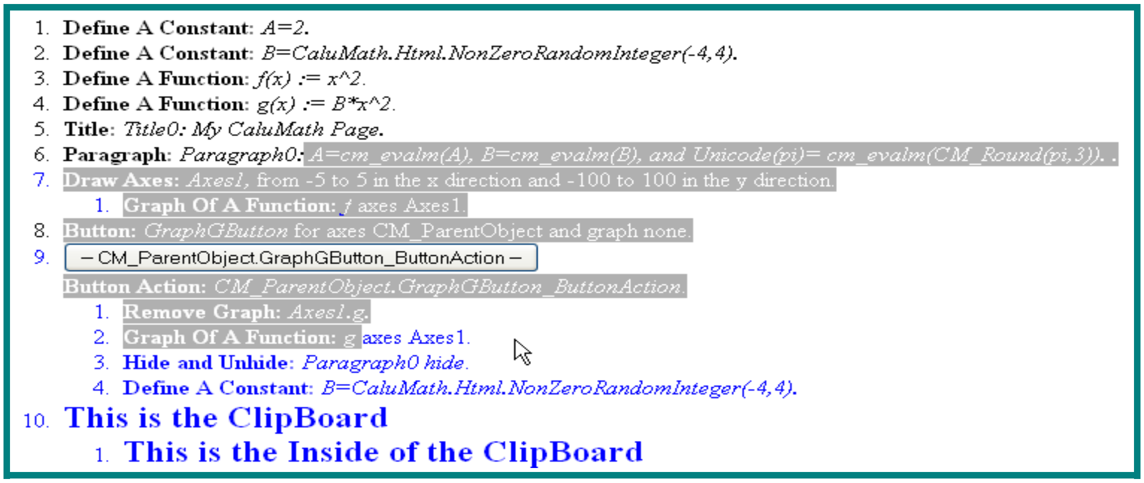
This is also an acceptable region to highlight for cutting and pasting, since both the Remove Graph g and the Graph of a Function items are children of the Button Action.



1. **Define A Constant**: *A=2.*
2. **Define A Constant**: *B=CaluMath.Html.NonZeroRandomInteger(-4,4).*
3. **Define A Function**: *f(x) := x^2.*
4. **Define A Function**: *g(x) := B*x^2.*
5. **Title**: *Title0: My CaluMath Page.*
6. **Paragraph**: *Paragraph0: A=cm_evalm(A), B=cm_evalm(B), and Unicode(pi)= cm_evalm(CM_Round(pi,3)).*
7. **Draw Axes**: *Axes1,* from -5 to 5 in the x direction and -100 to 100 in the y direction.
    1. **Graph Of A Function**: *f* axes Axes1.
8. **Button**: *GraphGButton* for axes CM_ParentObject and graph none.
9. [– CM_ParentObject.GraphGButton_ButtonAction –]

    Button Action: *CM_ParentObject.GraphGButton_ButtonAction.*
    1. **Remove Graph**: *Axes1.g.*
    2. **Graph Of A Function**: g axes Axes1.
    3. **Hide and Unhide**: *Paragraph0 hide.*
    4. **Define A Constant**: *B=CaluMath.Html.NonZeroRandomInteger(-4,4).*
10. **This is the ClipBoard**
    1. **This is the Inside of the ClipBoard**

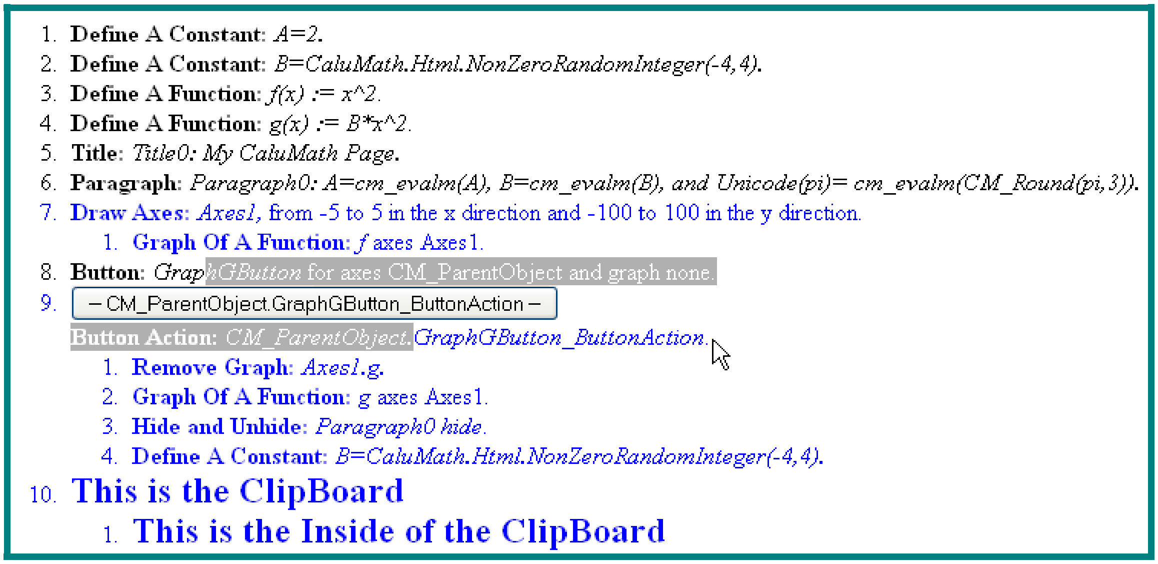
This is not an acceptable region to highlight, since Paragraph0 is a top-level item in the page and Graph of a Function is a child of a Button Action.



1. **Define A Constant**: *A=2.*
2. **Define A Constant**: *B=CaluMath.Html.NonZeroRandomInteger(-4,4).*
3. **Define A Function**: *f(x) := x^2.*
4. **Define A Function**: *g(x) := B*x^2.*
5. **Title**: *Title0: My CaluMath Page.*
6. **Paragraph**: *Paragraph0: A=cm_evalm(A), B=cm_evalm(B), and Unicode(pi)= cm_evalm(CM_Round(pi,3)). .*
7. **Draw Axes**: *Axes1,* from -5 to 5 in the x direction and -100 to 100 in the y direction.
    1. **Graph Of A Function**: *f* axes Axes1.
8. **Button**: *GraphGButton* for axes CM_ParentObject and graph none.
9. [– CM_ParentObject.GraphGButton_ButtonAction –]

    Button Action: *CM_ParentObject.GraphGButton_ButtonAction.*
    1. **Remove Graph**: *Axes1.g.*
    2. **Graph Of A Function**: g axes Axes1.
    3. **Hide and Unhide**: *Paragraph0 hide.*
    4. **Define A Constant**: *B=CaluMath.Html.NonZeroRandomInteger(-4,4).*
10. **This is the ClipBoard**
    1. **This is the Inside of the ClipBoard**

Finally, the screen shot below shows an acceptable region that we actually want to use to cut and paste the button and Button Action to the Clipboard. Please click the button associated to the Button Action to make sure the children are visible. Then place your cursor somewhere on the GraphGButton item and drag it down to the Button Action item.



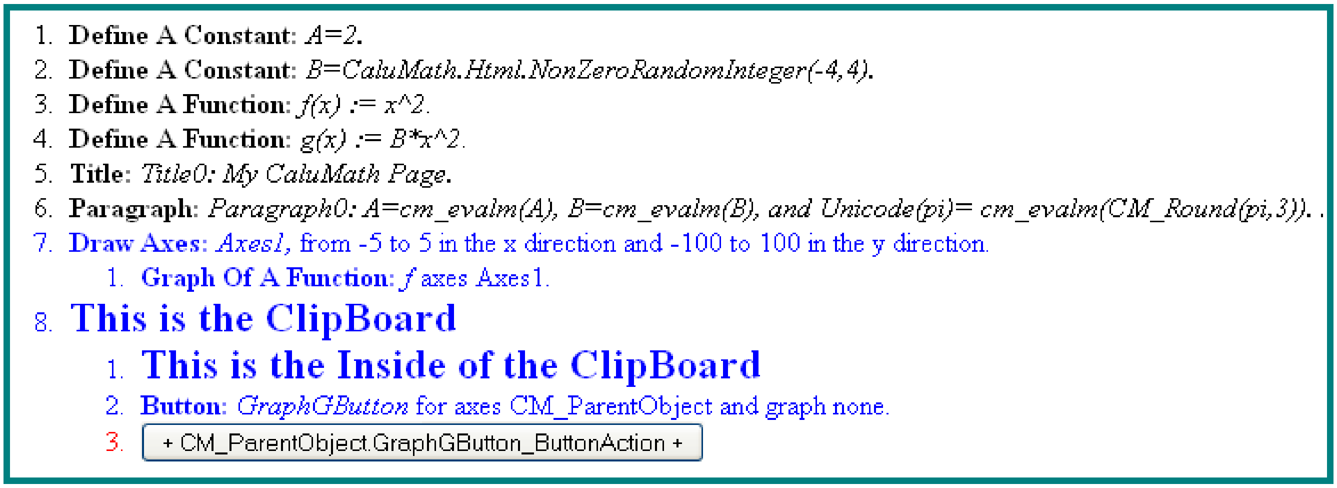
1. **Define A Constant**: *A=2.*
2. **Define A Constant**: *B=CaluMath.Html.NonZeroRandomInteger(-4,4).*
3. **Define A Function**: *f(x) := x^2.*
4. **Define A Function**: *g(x) := B*x^2.*
5. **Title**: *Title0: My CaluMath Page.*
6. **Paragraph**: *Paragraph0: A=cm_evalm(A), B=cm_evalm(B), and Unicode(pi)= cm_evalm(CM_Round(pi,3)).*
7. **Draw Axes**: *Axes1,* from -5 to 5 in the x direction and -100 to 100 in the y direction.
    1. **Graph Of A Function**: *f* axes Axes1.
8. **Button**: *GraphGButton* for axes CM_ParentObject and graph none.
9. – CM_ParentObject.GraphGButton_ButtonAction –

   Button Action: *CM_ParentObject.GraphGButton_ButtonAction.*
    1. **Remove Graph**: *Axes1.g.*
    2. **Graph Of A Function**: *g* axes Axes1.
    3. **Hide and Unhide**: *Paragraph0 hide.*
    4. **Define A Constant**: *B=CaluMath.Html.NonZeroRandomInteger(-4,4).*
10. **This is the ClipBoard**
    1. **This is the Inside of the ClipBoard**

After doing this, click the Cut button, then click the line that says

This is the Inside of the ClipBoard

and then click the Paste After button. The cut and paste window will reload with your changes. Click Cancel in the dialog box that asks if you want to paste the copied item again. When you are done, the cut and paste screen should look like the screen shot below. Note that both the Button and the Button Action are in the Clipboard.



1. **Define A Constant**: *A=2.*
2. **Define A Constant**: *B=CaluMath.Html.NonZeroRandomInteger(-4,4).*
3. **Define A Function**: *f(x) := x^2.*
4. **Define A Function**: *g(x) := B*x^2.*
5. **Title**: *Title0: My CaluMath Page.*
6. **Paragraph**: *Paragraph0: A=cm_evalm(A), B=cm_evalm(B), and Unicode(pi)= cm_evalm(CM_Round(pi,3)).* .
7. **Draw Axes**: *Axes1,* from -5 to 5 in the x direction and -100 to 100 in the y direction.
    1. **Graph Of A Function**: *f* axes Axes1.
8. **This is the ClipBoard**
    1. **This is the Inside of the ClipBoard**
    2. **Button**: *GraphGButton* for axes CM_ParentObject and graph none.
    3. + CM_ParentObject.GraphGButton_ButtonAction +

Cut and Paste Warnings: You can do a lot of damage to your document by cutting and pasting items without giving thought to what you are doing. These are principles to keep in mind when cutting and pasting items:

1. If you remove a button from your document, make sure you remove its associated Button Action. If you remove the button and leave the Button Action, you will encounter an error (and your page will not load properly), since the Button Action depends on the existence of its associated button. In general, if you remove an object, you must also remove associated objects that depend on it.
2. Do not place items in positions where they are not available when they are required. For example, our graph of f in Axes1 requires that the function f be defined. If we cut and paste f to the end of the web page, an error will occur, because the page will be trying to graph f before the function has been defined. In addition, if we cut and paste the constant A to the end of the document, errors will occur, since Paragraph0 evaluates the value of A, which means that the constant A must exist before Paragraph0 is created in the web page.

While we have the cut and paste window open, and thus, can see the structure of our web page, this gives us an opportunity to discuss the placement of constants and functions in CaluMath web pages.

Constants and Functions Placement: When you define constants and functions, CaluMath usually places them at the top of your web page in the order that you define them. This is done so that you can access them anywhere in your web page (because they will automatically appear before items such as paragraphs and axes). If you decide, as you are constructing your page, that you should have defined new constants and functions, then you can do this without worry; CaluMath will automatically place them at the top of your page so they will be available to objects you previously created. The only exceptions to this rule are the following:

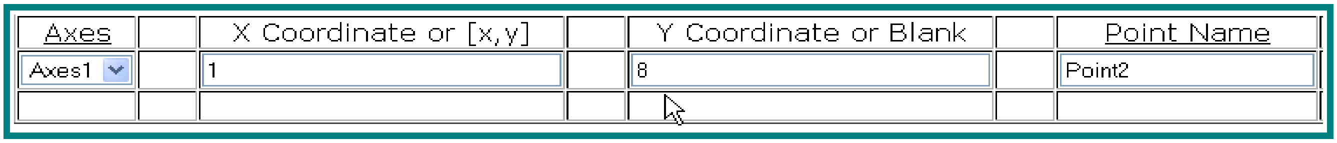
1. If you are in a Button Action, or a similar type of container item such as a Routine, For Loop or Conditional, then constants and functions you define while in that Button Action are placed in the Button Action instead of being moved to the top of the page. This makes sense, since you want the items in a Button Action to only be created when the user clicks a button, instead of being created as the web page loads.
2. Some constants and functions are defined in terms of points that appear in a set of axes. For example, if you construct two points, Point1 and Point2 in the axes Axes1, you can define a cubic function whose local maximums and minimums occur at Point1 and Point2. To define this function, the function needs to know the position of Point1 and Point2, and the function will not have this information if CaluMath moves the function to the top of the page before the axes and points are created. There is an option in the screen for creating functions that allows you select the name of the axes which the function requires in order to be defined correctly. This entry will make the function definition a child of the axes, instead of moving it to the top of the document. We will see an example of this in a minute.

In the next part of the tutorial, we will construct two points in Axes1, define the function h to be the cubic function whose local maximum and minimum occur at the two points. We will also define the derivative h′ and graph the function and its derivative. We will enable the user to drag the two points and thus change the definition of the functions and their graphs. Finally we will ask them to drag the graph so that h(3) = h′(3)+12 and check whether their answer is correct.

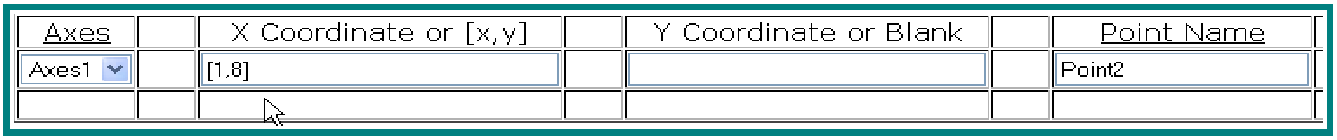
If you still have the cut and paste window open, click the Hide Constructed Page button to return to the main window.

Edit Axes: Since we are no longer going to graph g (which required a wide range of y values), let us edit Axes1 so that the y values only range between -5 and 25. This will give the user a clearer idea of the y values on the graphs we construct. To do this, go to the drop down menu below the Main Menu and select *Draw Axes named Axes1*. Click the Edit Selected Item button. In the screen that appears, enter *-5* in the Beginning Y Value field and enter *25* in the Ending Y Value field. Click the Finish Draw Axes button.

Points: We now plot points at (-2,20) and (1,8) in Axes1. In the Main Menu, click the Axes and Graphs item and click Plot A Point in the sub-menu. We will only concern ourselves with the required table at the bottom of the screen that appears. CaluMath suggests a name for the point, however type *Point1* in the Point Name field. We have two choices, we can enter the x and y coordinates of the point (-2,20) separately by entering *-2* in the X Coordinate or [x,y] field and entering *20* in the Y Coordinate or Blank field. Alternately, we can enter *[-2,20]* in the X Coordinate or [x,y] field and leave the Y Coordinate or Blank field blank. This feature is true for most CaluMath fields that require x and y coordinates, you can enter them separately, or you can enter the ordered pair, enclosed in brackets, in the x coordinate field. **Please note that coordinates are always enclosed in brackets, not parentheses.** The reason for this is that coordinates are technically arrays of numbers (meaning a list of numbers) and most programming languages denote lists by enclosing the entries in brackets. Click the Finish Plot a Point and Do Another Plot A Point button to create the second point. Enter the name *Point2* in the Point Name field, enter *[1,8]* in the X Coordinate or [x,y] field and click the Finish Plot A Point button to return to the main window. Recall that the true name for these points are Axes1.Point1 and Axes1.Point2 because they are plotted in Axes1. Below are screen shots of the two ways to correctly enter the coordinates for Point2.

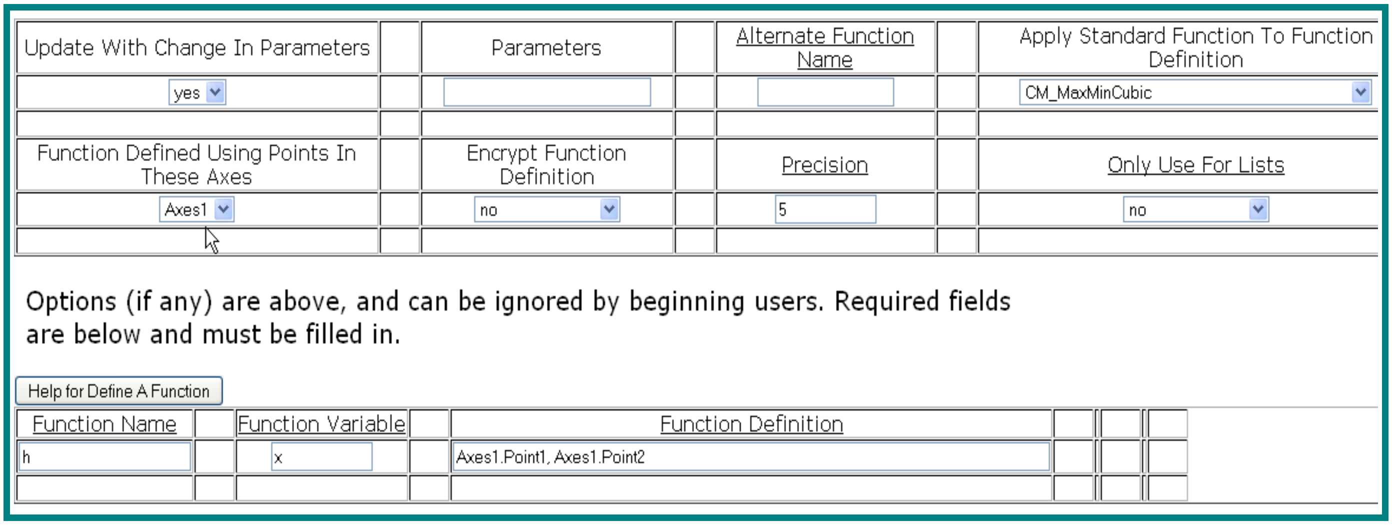


Axes	X Coordinate or [x,y]	Y Coordinate or Blank	Point Name
Axes1	1	8	Point2



Axes	X Coordinate or [x,y]	Y Coordinate or Blank	Point Name
Axes1	[1,8]		Point2

Standard Functions: We will now define h to be the cubic that has its local maximum and minimum at Point1 and Point2. In the Main Menu select Functions and Constants and click Define a Function in the sub-menu. Below is a screen shot of what you should enter. After the screen shot, we will discuss each entry individually.



Update With Change In Parameters	Parameters	Alternate Function Name	Apply Standard Function To Function Definition
yes			CM_MaxMinCubic
Function Defined Using Points In These Axes	Encrypt Function Definition	Precision	Only Use For Lists
Axes1	no	5	no

Options (if any) are above, and can be ignored by beginning users. Required fields are below and must be filled in.

Function Name	Function Variable	Function Definition
h	x	Axes1.Point1, Axes1.Point2

1. In Apply Standard Function to Function Definition, select *CM_MaxMinCubic*. This instructs the CaluMath Page Maker that you want to define a cubic whose maximum and minimum occur at prescribed values.
2. In The Function Definition, enter *Axes1.Point1, Axes1.Point2*. This indicates the two points through which the cubic should go. CaluMath will automatically define the correct function for us. There are a variety of ways you can enter this information; we list some of them here. All of the following produce a cubic whose maximum and minimum lies at the points (-2,20) and (1,8):

   a) Axes1.Point1, Axes1.Point2

   b) Axes1.Point1, [1,8]

   c) Axes1.Point1, 1,8

   d) [2,15],[1,8]

   e) 2,15,1,8

   In general, CM_MaxMinCubic expects an entry of the form x1,y1,x2,y2, however it recognizes that names of points or coordinates of points will account for two coordinates. Please note that the following should not be used: 2,[15,1],8, since [15,1] is not the coordinates of a point on the graph of h.
3. In Function Name enter *h* and in Function Variable enter *x*. This means that the function will be defined in terms of the variable x (you could chose a different variable, such as t if you desire).
4. In Function Defined Using Points In These Axes select *Axes1*, since the function will only make sense if it is placed as a child of Axes1. If this is not done, CaluMath will place the definition of the function at the top of the page and the page will not load correctly since Axes1.Point1 and Axes.Point2 will not have been constructed yet.
5. In *Update With Change In Parameters*, select *yes*. A complete discussion of parameters will take us too far afield at this juncture, however this item deserves some discussion at this point. The definition of h depends on the position of the points Axes1.Point1 and Axes1.Point2. We have two choices: if the points are moved we can redefine h to be the function whose graph

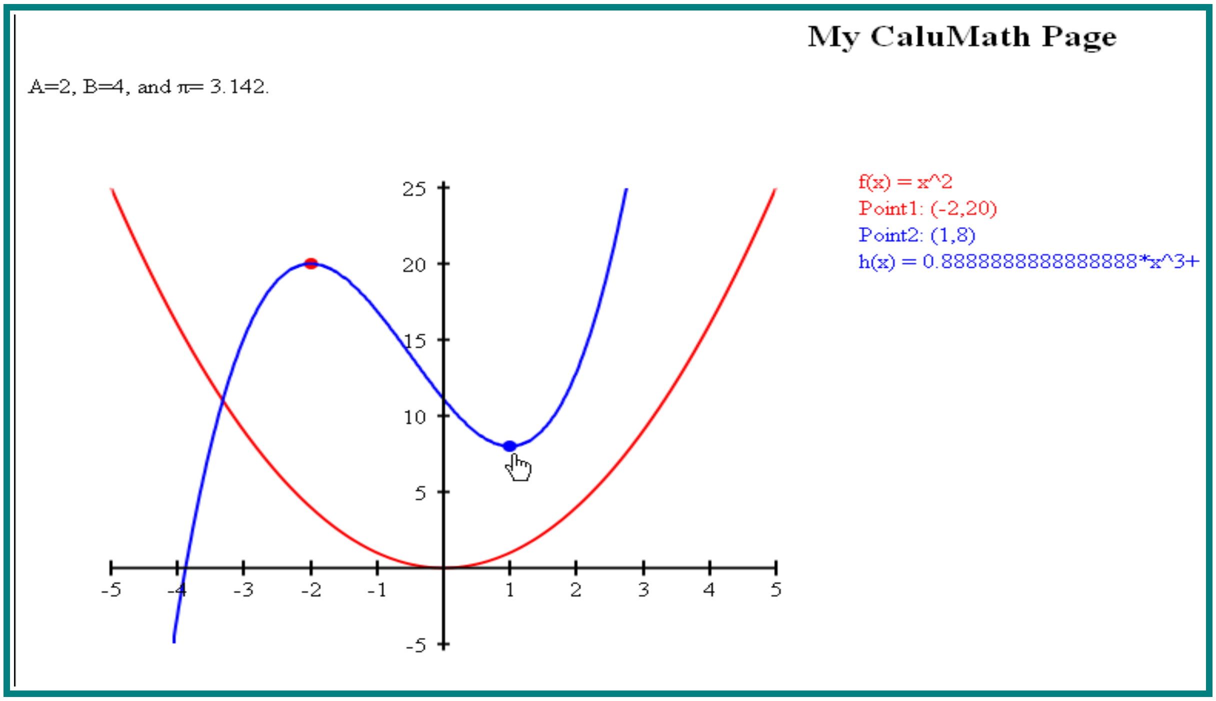
goes through the points' new positions, or we can decide to not change the definition of h, so that its definition is no longer tied to the positions of the points Point1 and Point2. In our case, we want the user to drag the points and update the graph, therefore we select *yes* in this field.

In this case, Axes1.Point1 and Axes1.Point2 are considered parameters for the function h, since the definition of h depends on them. Usually you have to enter the parameters (enclosed in brackets) on which the function depends in the Parameters field when you define the function. Therefore we could have entered *[Axes1.Point1, Axes1.Point2]* in the Parameters field. However when using standard functions (such as CM_MaxMinCubic) the CaluMath Page Maker will automatically recognize any entry in the Function Definition field that consists of a point or previously defined constant. It will not automatically recognize parameters in any other case, and it will not automatically recognize parameters for any function that is not a standard function. To be safe, you can always enter the parameters on which the function depends since CaluMath will automatically remove duplicate entries between the entered parameters and those that it calculates. Click the Finish Define A Function button to return to the main window.

Updatable Graphs: We now want to graph h, and do so in a ways that, as points Point1 and Point2 are dragged, the graph is automatically updated. Using the Axes and Graphs item in the Main Menu, you can graph functions, line segments, and circles and arcs. However these graphs are static; once they are created, they can be removed, however they cannot be modified. What we want is to create an Updatable graph; once these graphs are created, they can be updated to any type of graph as often as you need. For example, if you create an updatable graph called Updatable1, it can be updated to the graph of a function, then later updated to the graph of a circle, and later updated to the area between two functions.

To create an updatable graph, click the Updatable Graphs item in the Main Menu and then click Make Updatable Graph in the sub-menu. Enter the name *Updatable1* in the Updatable Graph Name field. Click the Finish Make Updatable Graph button to return to the main window. Creating the updatable graph reserves computer resources to that it can be graphed quickly; the graph remains invisible until you update the graph to the graph of a function, circle, etc.

Update To Function Graph: We now update the graph of Updatable1 to the graph of h. In the Main Menu, click Updatable Graphs and click Update To Function Graph in the sub-menu. In the Axes field select *Axes1*, in the Function field select *h* and in the This Updatable Graph field select *Updatable1*. In the Update Upon Function Redefinition field select *yes*; this ensures that as h is redefined, the graph will be automatically updated. Click the Finish Update To Function Graph to return to the main window. Click the View item to view your page. Below is a screen shot of the page we constructed.



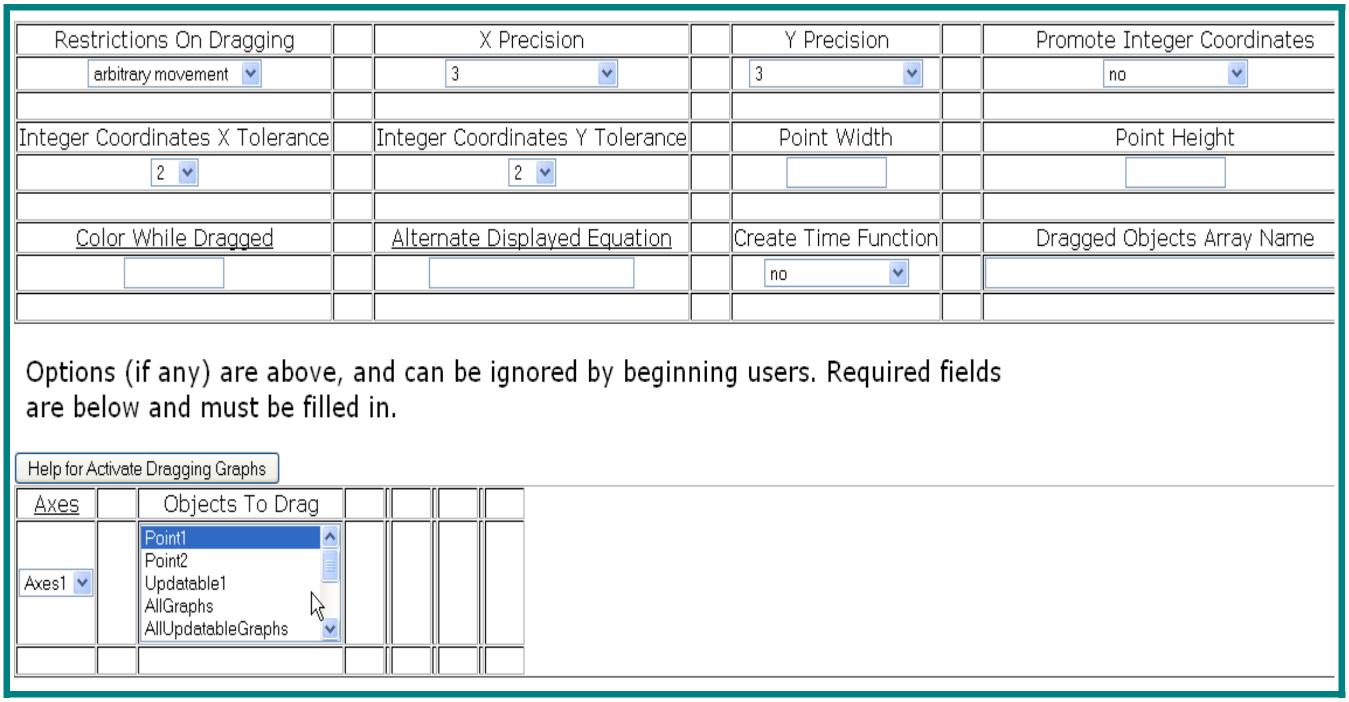
My CaluMath Page

A=2, B=4, and π= 3.142.

f(x) = x^2
Point1: (-2,20)
Point2: (1,8)
h(x) = 0.888888888888888*x^3+

Note that the defining equation of h displayed is awkward, there is really nothing that can be done about this, except for editing Updatable1 and choosing not to display the equation.

Dragging Graphs: We now activate Axes1 so the user can drag points. In the Main Menu, click Dragging Objects and click Activate Dragging Graphs in the sub-menu. Below is a screen shot; we will discuss many of the items individually.



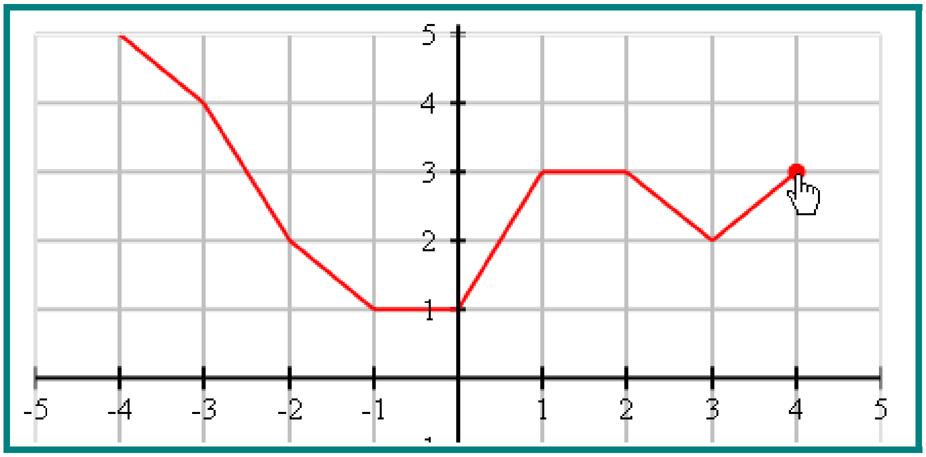
Restrictions On Dragging	X Precision	Y Precision	Promote Integer Coordinates
arbitrary movement	3	3	no
Integer Coordinates X Tolerance	Integer Coordinates Y Tolerance	Point Width	Point Height
2	2		
Color While Dragged	Alternate Displayed Equation	Create Time Function	Dragged Objects Array Name
		no	

Options (if any) are above, and can be ignored by beginning users. Required fields are below and must be filled in.

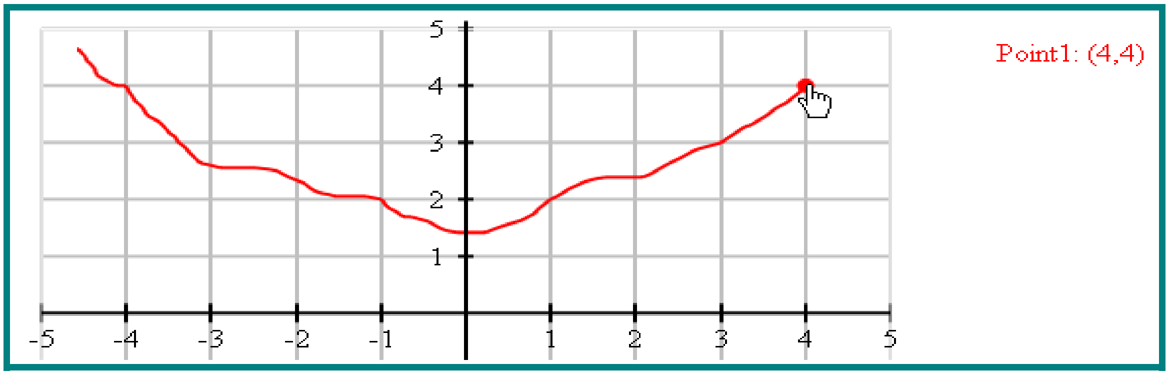
Axes	Objects To Drag
Axes1	Point1 Point2 Updatable1 AllGraphs AllUpdatableGraphs

1. **Objects to Drag:** There are a variety of choices for Objects to Drag, and you may select multiple items. The three items that are draggable in Axes1 are the two points and the graph Updatable1 (because the graph of f is static and cannot be dragged). You can also select categories of objects to be dragged, such as all points or all graphs. Select Point1 and Point2 by clicking down on Point1 and then holding the Ctrl key on your keyboard and clicking on Point2.
2. **Restrictions on Dragging:** select *arbitrary movement*. Selecting *x always increasing* is appropriate for when you want to capture the user's mouse movement and use it to define a function. In that case any motion of the mouse from right to left is ignored, so that a well defined function can be created.
3. **X Precision and Y Precision** rounds the coordinates of the mouse while dragging to a specified number of decimal places. You can also select .25 or .5 to round off the coordinates to the nearest fourth or half respectively. If you select CM_DragXPrecision or CM_DragYPrecision, it means that you have previously defined the constants CM_DragXPrecision and CM_DragYPrecision to be one of the numbers 0, 1, 2, … ,12, .25 or .5. In that case, the vales these constants define will be used in rounding.

   Rounding is a very important consideration when designing a page which includes graphs to be dragged by the user. For example, suppose you ask the user to drag a graph of a function h so that h(1) = 7. To do this, you would probably want the Y precision set to either .25, .5 or 0, so that the user has an easier time of obtaining integer y values. If you set Y Precision to 3, there is a good chance that the user would obtain h(1) = 7.012 or h(1) = 7.004, but will not achieve an integer value for h(1). Often selecting .25 for X Precision and either .25, .5 or 0 (depending on the scaling in the y direction) is appropriate for Y Precision. For your page, select *.25* for X Precision and *0* for Y Precision.
4. **Promote Integer Coordinates:** is another way CaluMath enables the user to be successful dragging their mouse over a desired point with integer coordinates. One might think that selecting X Precision and Y Precision to be 0 might be the best way to ensure that a graph will have integer coordinates. Unfortunately, this choice leads to graphs such as the one below, in which the page captured the user's mouse movements and used them to create a graph. Clearly this type of graph consists of straight segments, which may seem slightly unnatural for the user, since the path of their mouse probably will not consist straight paths. The graph does have the property, however, that it goes through many points with integer coordinates.



As an alternative, the graph below was obtained by setting the X and Y Precision to 3, however *yes* was selected in Promote Integer Coordinates and both **Integer Coordinates X Tolerance** and **Integer Coordinates Y Tolerance** were set to *4*. The effect of this is that a typical mouse coordinate is rounded to 3 decimal places, however when the mouse is within 4 pixels in the x and y direction of a point with integer coordinates, the mouse moves to the point with integer coordinates, and the rounded mouse position is captured. In the screen shot below, you can see that the graph, which was drawn freehand on the computer screen, is much more natural, however it goes through many points with integer coordinates (for example (4,4)), even though most coordinates were rounded to 3 decimal places.



For our page, we are not capturing the user's mouse motion, we are only interested in allowing them to move Point1 and Point2. Therefore select *no* in Promote Integer Coordinates. Because we selected *no*, any selection in Integer Coordinates X Tolerance and Integer Coordinates Y Tolerance is ignored.

5. **Create Time Function:** This allows you to capture the user's mouse movements and the time when the user's mouse was at that position. This can be used to construct an animation that replays the motion of the user's mouse. We have no need for this at this time, therefore leave the default value of *no* in Create Time Function.

Click the Finish Activate Dragging Graphs button to return to the main window. Click the View item to view your page and drag Point1 and Point2 to new positions and observe that the definition of h is updated and the graph reflects the new definition of h.

One of the powerful features of CaluMath is that you can design special routines that are executed either at the beginning of when the user starts to drag their mouse, while they are dragging their mouse, or after they drag their mouse. For example, suppose as the user drags their mouse, we want a paragraph to display some information about the graph, for example, the value h′(-1), meaning the value of the derivative of h at -1, and h(-1).

We emphasize that we are doing this to give you practice with Routines, and further experience creating graphs of functions rather than for pedagogical reasons. It should be emphasized however, that the information you display or do not display has important pedagogical implications, and CaluMath's power in this regard means it can be used to produce a wide variety of non-traditional problems.

Routines: A routine is a container that contains items you want constructed when the routine is executed. In this respect, it is very much like a Button Action: the items in a Button Action are constructed when the user clicks a button while the items in a routine are constructed when you instruct the page to execute the routine. Please note that there is a difference between defining a routine and executing a routine. When you define a routine, you tell CaluMath the items that you want it to construct sometime in the future. When you execute a routine, you are actually telling CaluMath to construct those items. As the user drags their mouse we want a paragraph displaying the above information to appear, and as their mouse position changes we will remove the old paragraph and replace it with a new one with the updated information. Since we want to display values of the derivative, we need to define h′ first.

Derivatives: Unfortunately, a CaluMath computer algebra system that calculates derivatives, factors polynomials, and does symbolic calculations is still in development and will not be part of release 2.0 of CaluMath. This means that, in general, you have to define the derivative of a function yourself in the same way you defined the original function. However, standard functions in CaluMath have derivatives that CaluMath can automatically calculate. To define the derivative of h, click on Functions and Constants in the Main Menu and click on Define a Function in the sub menu. In Apply Standard Function to Function Definition, select *CM_MaxMinCubicDerivative*, this means it will calculate the derivative of the function h (which was defined using the CM_MaxMinCubic selection). In The Function Definition, field, enter *Axes1.Point1, Axes1.Point2*. This means that the function defined will be the derivative of the function that is a cubic with a maximum and minimum at the points Point1 and Point2. In Function Name, enter *hprime*; this will automatically be displayed as h′ if you were to graph it. **We should note that all names in CaluMath must begin with a letter and can contain only letters, numbers, and the underscore character.** Enter *x* in Function Variable and select *yes* in the Update With Change In Parameters field, since we want h′ to be updated as the points move. Finally, **and very importantly, select *Axes1* in the Function Defined Using Points In These Axes menu**, so that the definition of hprime is placed as a child of the axes H and is not placed at the beginning of the web page.

Graph the Derivative: We now graph h′, and see how it changes as the graph of h is updated. Since we want the graph of h′ to be updated, we will construct a new Updatable Graph which will be updated to the graph of h′. In the Main Menu, click Updatable Graphs and click Make Updatable Graph in the sub-menu. Enter the name *Updatable2* in the Updatable Graph Name field. Click the Finish Make Updatable Graph button to return to the main window. We now update Updatable2 to the graph of h′ by clicking Updatable Graphs in the Main Menu and clicking Update To Function Graph in the sub-menu. In the Axes field select *Axes1*, in the Function field select *hprime* and in the This Updatable Graph field select *Updatable2*. In the Update Upon Function Redefinition field select *yes*; this ensures that as hprime is redefined, the graph will be automatically updated. Click the Finish Update To Function Graph to return to the main window. Click the View item at the top of the page to view the page. Note that as you drag Point1 and Point2, the graphs of h and h′ are updated.

If you have not saved your page, it would be wise to do this now. Click Save at the top left of the CaluMath Page Maker to do this and click Run if the Java applet prompt presents itself.

You may notice that the graph of f is no longer necessary and actually makes the page a little cluttered. Click the Cut & Paste item at the top of the CaluMath Page Maker so we can move the graph of f to the Clipboard. After the cut and paste window loads, click somewhere near the middle of the line corresponding to the graph of f, then click the Cut Button. After doing this, click on the This Is The Inside Of The Clipboard line and click the Paste After button. When the cut and paste window reloads, you will notice that the graph of f is now the first line in the Clipboard. Click the Hide Constructed Page button to return to the main window.

Edit Axes: Click the View item and look at your page. It would probably be improved by having the y values in the axes go down to about -15; to do this, edit Axes1. Select *Draw Axes named Axes1* from the drop down below the Main Menu and click the Edit Selected Item button. Enter *-15* in the Beginning Y Value field and click the Finish Draw Axes button.

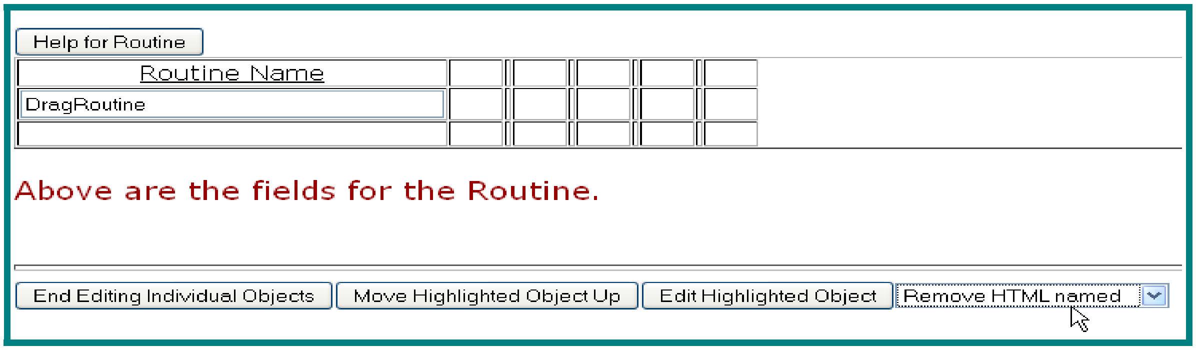
Drag Routine: We now create a routine that creates a new paragraph that displays h(-1) and h′(-1) and removes the old paragraph that displayed these values previously. We will then assign the routine to the Point1 and Point2 so that the routine is executed as the points are dragged.

In the Main Menu, click Conditionals and Routines and click Routine in the sub-menu. Enter *DragRoutine* in the Routine Name and click the Finish Routine button. A pop up window states *Now add the item you want to your Routine*. Close the pop up window. Note that You are in a routine is displayed in red above the main menu; this reminds you that every item you create will be added into the routine container. Since we want to create a paragraph, click Text and Html in the Main Menu and click Text in the sub-menu. Enter Paragraph2 in the Name field if it does not already appear there and select *x-large* for Text Size. Enter *Note that h(1)=cm_evalm(CM_Round(h(1),3)) and h′(1)= cm_evalm(CM_Round(hprime(-1),3))*.. In the large text box. Note that this will cause the values h(-1) and h′(-1) to be evaluated mathematically and rounded to 3 decimal places. Click the Finish Text button. In the dialog box that pops up, click OK, since we want to add more items to our Routine.

Remove Html: After closing the Add another item to your Routine window, click Visibility in the Main Menu and click Remove HTML from the sub-menu. Select *Paragraph2* in the Object field and click the Finish Remove HTML button. In the dialog box, click Cancel, since we do not want to add any more items to our routine. Close the *You have finished your Routine* window.

Note that we want to remove the old paragraph before creating the new one, however the order of the creation and removal is currently reversed We saw this problem earlier when we wanted to remove the old graph of g before creating a new graph of g. We could cut and paste the remove paragraph item before the create Paragraph2 item, however we will demonstrate a different technique that occasionally comes in handy. This will demonstrate that editing a container element, such as a Routine, Button Action, Conditional or For Loop (but not a set of axes), allows you to edit its children.

Editing A Routine: To edit the Routine, select the *Routine Named DragRoutine* entry from the drop down menu below the Main Menu and click the Edit Selected Item button. In the screen that opens, click the Edit The Individual Entries In The Routine button. Note that this causes the Finish, Cancel and Delete buttons for the Routine to be invisible and adds a set of buttons concerning the children of the Routine. In the drop down menu that appears, select *Remove HTML named*. There are two choices for what we can do, we can either click the Edit Highlighted Object to edit it, or we can click the Move Highlighted Object Up button, to make it appear earlier amongst the children. Click the Move Highlighted Object Up button. A screen shot is below.



Routine Name

DragRoutine

Above are the fields for the Routine.

After doing this, you will see that the Remove HTML element is before the Text Named Paragraph2 entry. Click the End Editing Individual Objects button. This reveals the Finish Routine button which you can now click. Close the *The changes to the Routine have been made* window.

Routine For Dragging: We have now defined the routine DragRoutine. We now need to indicate when the routine should be executed. For arbitrary routines, you would do this by clicking Conditionals and Routines from the Main Menu and clicking Execute A Routine from the sub-menu. However, we want this routine to be repeatedly executed as the user drags Point1 and Point2. To accomplish this, we need to do this a different way. Click Dragging Objects from the Main Menu and click Routine For Dragging from the sub-menu. In the Graph or Sliding Scale menu you will find a list of all the objects that can be dragged. In the Routine For Dragged Object menu you will find a list of all of the routines that have been created. In the When To Execute Routine menu you can select to execute the routine after the object has been dragged, while the object is being dragged, or at the beginning of the drag, when the user first clicks down on the object.

We want DragRoutine to be execute whenever Point1 or Point2 is moved; therefore we should associate it to these two objects. You could instead associate it to Axes1; in this case, it would be OK, however in many situations it is not a good idea to associate a routine to the axes. If we associate DragRoutine to the two points, it will only be executed when the points are dragged. If we associate it to the axes, DragRoutine will be executed anytime any object in Axes1 is dragged. In our particular case, it doesn't matter since we are only allowing the two points to be dragged. However if Axes1 contained many objects that could be dragged, associating the routine to the axes would cause the routine to be executed whenever any object is dragged, even though dragging those other objects would not update the definition of h nor require Paragraph2 to be removed and rewritten. In other words, we would be removing and reconstructing Paragraph2 for no reason. Dragging objects places a great demand on the computer, since calculations must be done quickly each time the mouse moves. It makes sense, therefore, to avoid unnecessary calculations.

Please do the following:

1. In the Graph or Sliding Scale menu select *Axes1.Point1* and in the When To Execute Routine menu select *After Dragging Object*. Click the Finish Routine For Dragging and Do Another Routine For Dragging button. Close the pop up window that appears.
2. In the Graph or Sliding Scale menu select *Axes1.Point1* and in the When To Execute Routine menu select *While Dragging Object*. Click the Finish Routine For Dragging and Do Another Routine For Dragging button. Close the pop up window that appears.
3. In the Graph or Sliding Scale menu select *Axes1.Point2* and in the When To Execute Routine menu select *After Dragging Object*. Click the Finish Routine For Dragging and Do Another Routine For Dragging button. Close the pop up window that appears.
4. In the Graph or Sliding Scale menu select *Axes1.Point2* and in the When To Execute Routine menu select *While Dragging Object*. **Click the Finish Routine For Dragging button**.

The above four steps mean that DragRoutine will be execute while Point1 and Point2 are being dragged and at the moment the user stops dragging the points. View your page and observe that Paragraph2 appears when the points are dragged and is constantly updated.

We conclude this page by presenting the following the problem for the user:Drag the two points so that $h(3) = h'(3) + 12$. To accomplish this we will need to

1. Construct a paragraph displaying the question.
2. Construct a button that the user clicks after they have completed the task and wants their answer checked.
3. Create a Button Action that contains the items to be constructed when the button is clicked. The Button Action needs to do the following:
    a) Check whether h(3) = h′(3) + 12 is true. This requires that CaluMath **Comparison** to be done.
    b) Provide a response if h(3) = h′(3) + 1 is true. This requires a CaluMath **Conditional**..
    c) Provide a different response if h(3) = h′(3) + 1 is not true. This requires a CaluMath **Conditional**..

We do the above items to complete the page.

Paragraph: Our page still contains Paragraph0, which is now serving no useful purpose, so we edit it to contain the information we want. Select *Text Named Paragraph0* from the drop down menu below the Main Menu and click the Edit Selected Item button. Select *large* in the Text Size menu and in the large text box, enter the following description (which you can cut and paste from this tutorial document):

Below are the graphs of two points, the graph of a function h which has a maximum or minimum at each of the two points, and the graph of h'. If you drag either of the points, the definition of h will change so that it becomes the function whose graph goes through the new positions of the points. The graph of h' will also change so that it is always the derivative of h.

Please drag the points so that the functions h and h' satisfy the following equation: h(3) = h'(3) + 12. When you have finished this task, click the Check Answer button.

Note that any line breaks you include in the text box are transferred to the web page. Click the Finish Text button.

Button: We now construct a Check Answer button. Click the Buttons and Boxes item in the Main Menu and click Button in the sub-menu. Enter *CheckAnswerButton* in the Button Name field and enter *Check Answer* in the Button Label field. Click the Finish Button button.

Button Action: We create a Button Action for the Check Answer button. Click Buttons and Boxes in the Main Menu and click Button Action in the sub-menu. Select *CM_ParentObject.CheckAnswerButton* in the Button menu. Click the Finish Button Action button. Close the *Now construct the item you want to add to your Button Action* window that pops up. We will now add items to our Button Action.

Conditional: We now add a conditional to our routine. The conditional will check whether h(3) equals h′(3) + 12, and if it is, it will display a paragraph saying that our answer is correct. To do this, click Conditionals and Routines in the Main Menu and click Conditional in the sub-menu. In the screen that opens, select *if* in the Branching Possibilities menu and enter *CorrectAnswerConditional* in Conditional Name (recall that names in CaluMath cannot contain spaces). You can think of a Conditional as a container, similar to a Button Action or a Routine. For a **Condition with Branching Type if**, the items in the container are executed if a certain condition is true. Click the Finish Conditional button to begin adding items to the Conditional container. Close the *Now do the comparison for the if* window. We are immediately taken to a screen in which we enter the condition that must be satisfied in order for the items in the Conditional container to be executed. Enter *h(3)* in the First Term field and enter *hprime(3)+12* in the Second Term field. Select = in the Comparison field and enter *.00001* in the Precision field. This means that the First Term and Second Term will be evaluated as numbers (since the comparison is =) and they will be considered to be equal if they are within .00001 of each other. Note that *equals* is a possible selection in the Comparison field; this selection should be used for entries that are not numbers. Incidentally the selection *in* should be made if you want to determine whether First Term is an element of an Array that is entered in Second Term.

It should be noted that you could enter three more conditions on the lines below and join them with OR or AND. We have no need to do this, so we click the Finish Comparison button. In the dialog box that appears, click OK since we want to add items to the Conditional container that are created if h(3) = h′(3) + 1. Close the Add another item to your Conditional window when it appears. Note that the text You are in a Button Action that contains a Conditional appears above the Main Menu to help you keep track of the placement of the items you are creating.

Paragraph: We now construct a paragraph to be displayed if the user's answer is correct. In the Main Menu, click Text and Html and click Text in the sub-menu. Select *large* in the Text Size field and enter *AnswerParagraph* in the Name field. In the large text box enter *h(3)=cm_evalm(h(3)) and h'(3)=cm_evalm(hprime(3)), therefore h(3)=h'(3)+12*. Click the Finish Text button. In the dialog box that appears, click Cancel, since we do not want to create any other items when the user's answer is correct. A new dialog box appears, asking if we want to add more items to the Button Action. We need to add another conditional, that is executed when the user's answer is wrong, however, let us not do this yet, so we can see if our first conditional is working properly. Therefore, click Cancel in this dialog box. Close the *You have finished your Button Action* window.

View the page and interact with it. The easiest way to obtain the correct answer is to move Point2 so that it has coordinates (3,12); since a maximum or minimum always occurs at Point2, h′ will always be 0 at the x coordinate of Point2. If you interact with the page and click the Check Answer button, you should see an AnswerParagraph created each time your answer is correct. Note, however, that the displayed text is probably not what you want. Due to the way JavaScript rounds numbers, a y coordinate of 12 may displayed as 11.9999999999999 and instead of 0, a very small number in scientific notation, such as 3.552713678800501e-15 is displayed. In addition, if you change the size of the window, you may find that mathematical text has inappropriate line breaks in the middle of it. We edit AnswerParagraph to take care of these problems.

Edit Paragraph: Select Text Named *AnswerParagraph* from the drop down menu below the Main Menu and click the Edit Selected Item button. In the large text box enter (or cut and paste from this document)

Your answer is correct. Note that <mi>h(3)=cm_evalm(CM_Round(h(3),3))</mi> and <mi>h '(3)=cm_evalm(CM_Round(hprime(3),3))</mi>, therefore <mi>h(3)=h'(3)+12</mi>.

Note that we have added CM_Round commands to round the input to 3 decimal places, this will ensure that all numbers will be displayed correctly. Note that we also enclosed each mathematical expression in a pair of <mi> </mi> tags. The tag <mi> signifies math-italics. Each opening <mi> tag must be paired with a closing </mi> tag (which contains a / symbol). Pairs of <mi> tags should never be nested, meaning that there is no reason to place pairs of these tags inside of another pair. The effect of these tags is to cause the text inside of them to be displayed in italics, and to never allow a line break inside the expression. **Always pair an opening <mi> tag with an ending </mi> tag; if you fail to do this your page will generate severe errors**. When you view your page, it should work correctly.

Conditional: We now need to create a conditional that is executed if the student's answer is incorrect. It will contain the following items:

1. An arrow drawn to the point with coordinates (3,h(3)) and an arrow drawn to the point with coordinates (3,h′(3)).
2. A paragraph stating that the user's answer is incorrect and says that they should be able to determine h(3) and h′(3) from the arrows drawn in the graph.

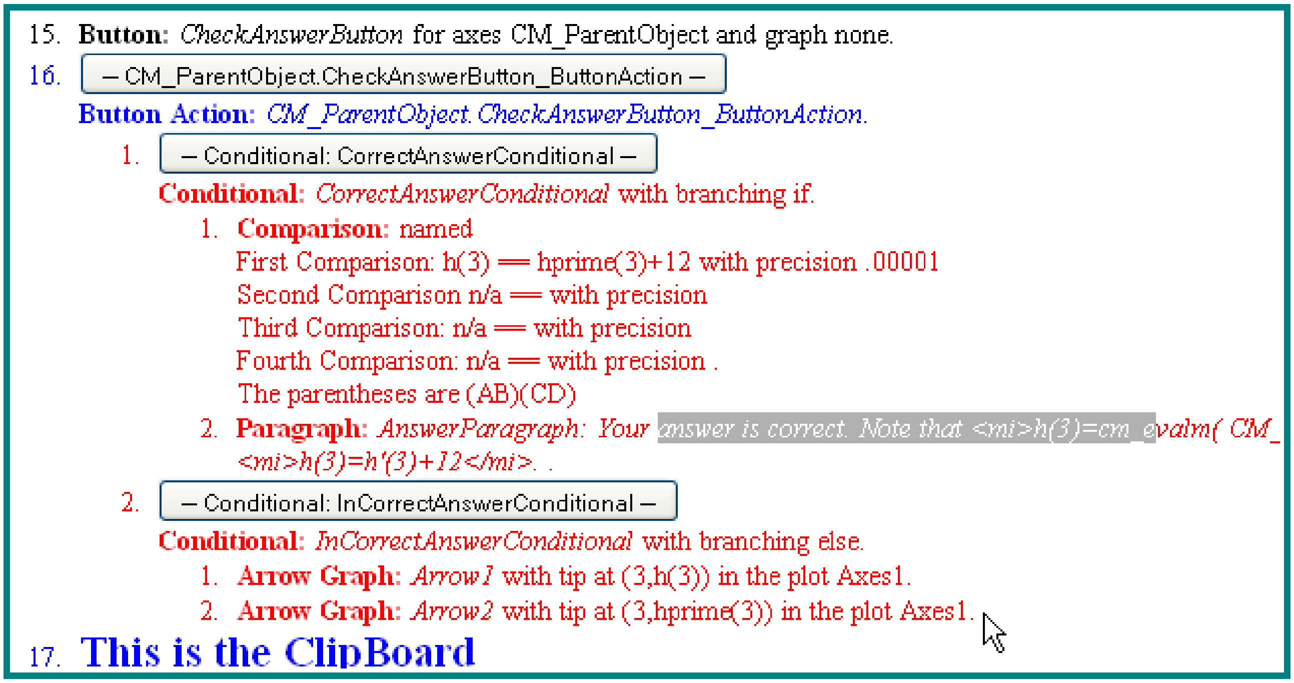
Recall that these conditionals are contained in a Button Action, so to make sure they are placed correctly, we need to select Buttons and Boxes from the Main Menu and click Add To Button Action from the sub-menu. Select *CM_ParentObject.CheckAnswerButton_ButtonAction* in the Button Action field and click the Finish Add To Button Action button. Close the *Now construct the item you want to add to your Add To Button Action* pop up window. We are now ready to create the conditional that will be executed if h(3) does not equal h′(3)+12. Click Conditionals and Routines in the Main Menu and click Conditional in the sub-menu. In the screen that opens select *else* in Branching Possibilities and enter *InCorrectAnswerConditional* in Conditional Name. **A Condition with Branching Type *else* means that the items in this Conditional container are executed if the condition in the previous conditional was not satisfied.** In particular, this means that this else-Conditional must be placed immediately after the previous conditional; no other items (such as paragraphs, buttons, etc) can be placed between an if-conditional and an else-conditional. Click the Finish Conditional button. Close the *Now add the item you want to your Conditional* window that pops up. Note that, in the if-Conditional, you were immediately taken to a Comparison screen which allowed you to check if h(3) and h′(3)+12 were equal. That does not happen in an else-Conditional because an else-Conditional is executed if the condition in the previous Conditional was not satisfied.

Arrows: We now create arrows pointing to the points with coordinates (3,h(3)) and (3,h′(3)). In the Main Menu, click Axes and Graphs and click Arrow Graph from the sub-menu. Enter *3* in Arrow X Coordinate and enter *h(3)* in Arrow Y Coordinate. Enter *50* in +/- X Length of Arrow Stem and *50* in +/- Y Length of Arrow Stem. This will create a stem on the arrow of length 50 pixels in the positive x and positive y directions, therefore the stem of the arrow will be in the positive x and y directions compared to the tip of the arrow. This actually means the arrow will point down and to the left. In the Optional Text field, enter the coordinates of the point *(3, cm_evalm(CM_Round(h(3),1)))*. Since this field expects text, we must use cm_evalm to evaluate h(3), and we use CM_Round to not have a coordinate displayed with a large number of decimal places. In Text Color enter *blue* to make sure the text stands out. Enter *Arrow1* in the Arrow Name field at the top left corner of the upper table. Click the Finish Arrow Graph button. In the dialog box click OK because we want to add more items to our conditional. Close the *Add another item to your Conditional* window that pops up.

Arrows: We create another arrow that points to (3,h′(3)). In the Main Menu, click Axes and Graphs and click Arrow Graph from the sub-menu. Enter *3* in Arrow X Coordinate and enter *hprime(3)* in Arrow Y Coordinate. Enter *-50* in +/- X Length of Arrow Stem and *50* in +/- Y Length of Arrow Stem. This will create a stem on the arrow of length 50 pixels in the negative x and positive y directions, therefore the stem of the arrow will be in the negative x and positive y directions compared to the tip of the arrow. This actually means the arrow will point down and to the right. The choice of this direction ensures that the two arrows will not intersect each other. In the Optional Text field, enter the coordinates of the point *(3, cm_evalm(CM_Round(hprime(3),1)))*. Enter *Arrow2* in the Arrow Name field at the top left corner of the upper table. In Text Color enter *red* to make sure the text stands out. Click the Finish Arrow Graph button. In the dialog box that appears click Cancel to stop adding items to the

Conditional. Although it is true that we need to add a paragraph informing the user that their answer is incorrect, it would be easier to cut and paste a copy of the AnswerParagraph we constructed above and modify the text. Also click Cancel in the next dialog box that appears, so that we finish the Button Action. Close the *You have finished your Button Action* window.

Cut & Paste: We will cut and paste a copy of AnswerParagraph. Click Cut & Paste to open the cut and paste window, click the *CM_ParentObject.CheckAnswerButton_ButtonAction* to reveal the contents of the Button Action and click both buttons corresponding to the conditionals to see the items contained in the conditionals. Highlight an area in the middle of the line corresponding to *AnswerParagraph* and click the Copy button. Click the line corresponding to Arrow2 and click the Paste After button. The screen shot below shows the screen before the Paste After button is clicked . Click Cancel in the dialog box that appears because we do not want to paste another copy of AnswerParagraph in our page. Click the Hide Constructed Page button to return to the main window.

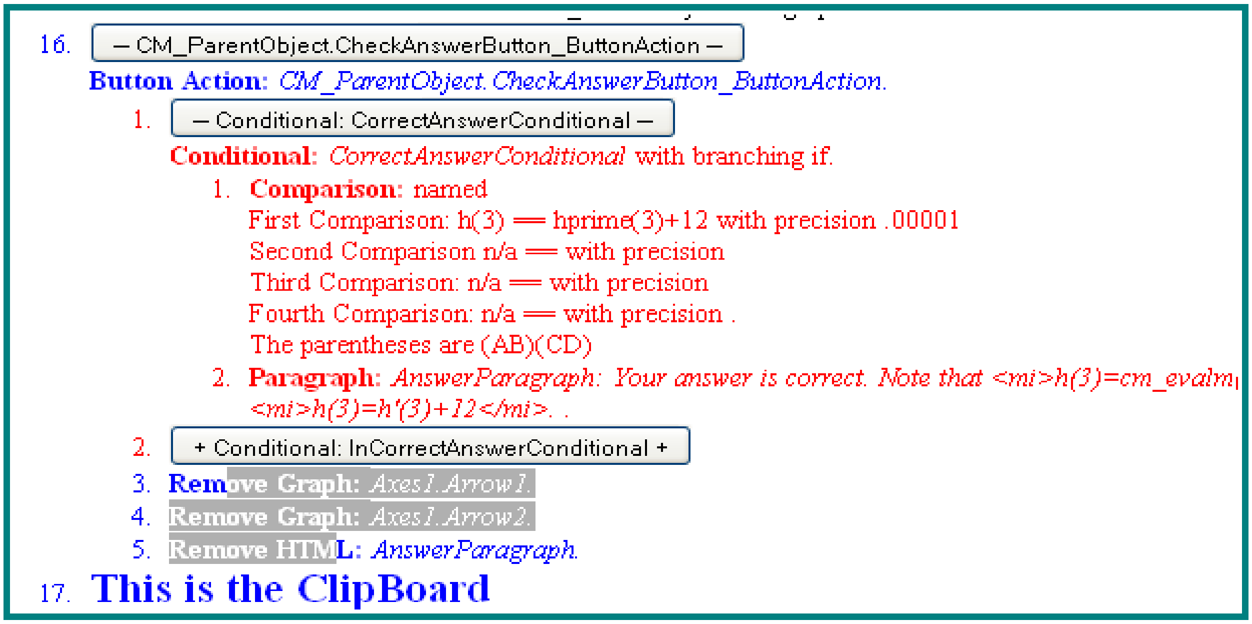

15. **Button:** *CheckAnswerButton* for axes CM_ParentObject and graph none.
16. – CM_ParentObject.CheckAnswerButton_ButtonAction –
Button Action: *CM_ParentObject.CheckAnswerButton_ButtonAction.*
    1. – Conditional: CorrectAnswerConditional –
    Conditional: *CorrectAnswerConditional* with branching if.
        1. **Comparison:** named
        First Comparison: h(3) == hprime(3)+12 with precision .00001
        Second Comparison n/a == with precision
        Third Comparison: n/a == with precision
        Fourth Comparison: n/a == with precision .
        The parentheses are (AB)(CD)
        2. **Paragraph:** *AnswerParagraph: Your answer is correct. Note that <mi>h(3)=cm_evalm(CM_*
        <mi>h(3)=h'(3)+12</mi>. .
    2. – Conditional: InCorrectAnswerConditional –
    Conditional: *InCorrectAnswerConditional* with branching else.
        1. **Arrow Graph:** *Arrow1* with tip at (3,h(3)) in the plot Axes1.
        2. **Arrow Graph:** *Arrow2* with tip at (3,hprime(3)) in the plot Axes1.
17. **This is the ClipBoard**

Edit Paragraph: Select *Text Named AnswerParagraph* **that is a child of the InCorrectAnswerConditional Conditional** from the drop down menu below the Main Menu and click the Edit Selected Item button. In the large text box enter (or cut and paste from this document) the text: *Your answer is not correct. Note that arrows have been drawn on the graph; these arrows indicate points that should allow you to determine h(3) and h'(3). You can drag the points to a new position and click the Check Answer button to try again.* Click the Finish Text button.

In general names in CaluMath should be unique, for example, if you have two paragraphs with the same name, and you want to remove one of them when the user clicks a button, the page will not know which paragraph to remove. However, a more precise statement is that, **at all times, there should only be one object in existence in the web page with a given name.** We are following this rule with our two paragraphs named AnswerParagraph above, since only one of them will appear at a time. The fact that these paragraphs have the same name can simplify your web page. For example, suppose we wanted to include another problem in this page for the user to solve. Setting up that problem would probably require removing anything that was created in response to the previous problem. In this case, we would want to remove AnswerParagraph, and when we do this, we are removing the paragraph that was created in response to a correct or incorrect answer. This is more simple than if we named the paragraphs CorrectAnswerParagraph and IncorrectAnswerParagraph and then had to remove both of them (because we would not know which one was present in the web page).

Remove Graphs and Remove Html: To complete the page, we should eliminate the previous AnswerParagraph and previously drawn Arrow1 and Arrow2 if they click the Check Answer button again. We now add these items to the Button Action for the Check Answer button. Click Buttons and Boxes in the Main Menu and click Add To Button Action in the sub-menu. Select *CM_ParentObject.CheckAnswerButton_ButtonAction* in the Button Action field and click the Finish Add To Button Action button. Close the *Now construct the item you want to add to your Add To Button Action* pop up window. Click Visibility in the Main Menu and click Remove Graph in the sub-menu. Select *Axes1.Arrow1* from the Graph menu and click the Finish Remove Graph And Do Another Remove Graph button. Now select *Axes1.Arrow2* from the Graph menu and click the Finish Remove Graph button. In the dialog box that appears, click OK since we still need to remove AnswerParagraph. Close the *Add another item to your Button Action* window that pops up. Click Visibility in the Main Menu and click Remove HTML. Select *AnswerParagraph* from the Object menu and click the Finish Remove HTML button. Click Cancel in the dialog box that appears since we do not want to add any more items to the Button Action and close the *You have finished your Button Action* window that pops up.

Cut & Paste: The final thing we must do is cut and paste the remove paragraph and remove arrows items we just created to the beginning of the Button Action, so the old items are removed before the new ones are created. Click Cut & Paste to open the cut and paste window, click the *CM_ParentObject.CheckAnswerButton_ButtonAction* to reveal the contents of the Button Action. You will see buttons corresponding to the two conditionals and buttons corresponding to each of the conditionals. We want to cut the three remove items and place them before the first conditional. To do this, you **must** click the Conditional: CorrectAnswerConditional button so we can see the first line that says *Conditional: CorrectAnswerConditional with branching if* . That line signifies the conditional, and we must place the three remove items in front of that line. Highlight the three remove items, as in the screen shot below and click the Cut button.



16. – CM_ParentObject.CheckAnswerButton_ButtonAction –

Button Action: *CM_ParentObject.CheckAnswerButton_ButtonAction.*

1. – Conditional: CorrectAnswerConditional –

Conditional: *CorrectAnswerConditional* with branching if.

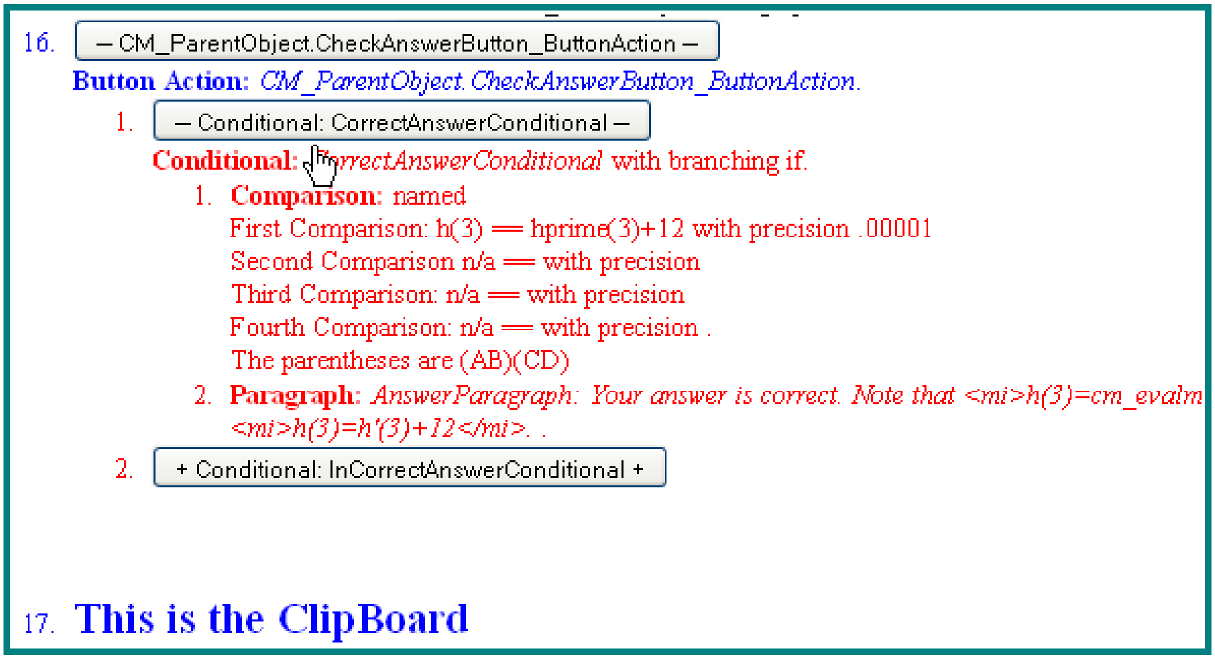
1. **Comparison:** named
First Comparison: h(3) == hprime(3)+12 with precision .00001
Second Comparison n/a == with precision
Third Comparison: n/a == with precision
Fourth Comparison: n/a == with precision .
The parentheses are (AB)(CD)

2. **Paragraph:** *AnswerParagraph: Your answer is correct. Note that <mi>h(3)=cm_evalm*
<mi>h(3)=h'(3)+12</mi>. .

2. + Conditional: InCorrectAnswerConditional +
3. **Remove Graph:** *Axes1.Arrow1.*
4. **Remove Graph:** *Axes1.Arrow2.*
5. **Remove HTML:** *AnswerParagraph.*

17. **This is the ClipBoard**

Now place your cursor on the conditional line, as in the screen shot below, click the line, and then click the Paste Before button.



16. – CM_ParentObject.CheckAnswerButton_ButtonAction –

Button Action: *CM_ParentObject.CheckAnswerButton_ButtonAction.*

1. – Conditional: CorrectAnswerConditional –

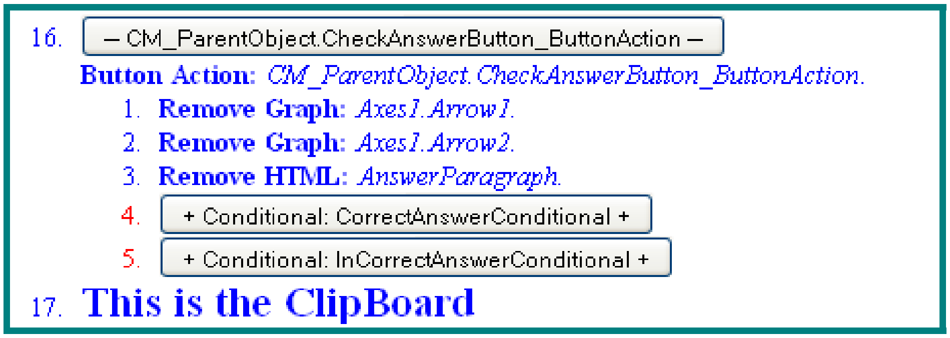
Conditional: *CorrectAnswerConditional* with branching if.

1. **Comparison:** named
First Comparison: h(3) == hprime(3)+12 with precision .00001
Second Comparison n/a == with precision
Third Comparison: n/a == with precision
Fourth Comparison: n/a == with precision .
The parentheses are (AB)(CD)
2. **Paragraph:** *AnswerParagraph: Your answer is correct. Note that <mi>h(3)=cm_evalm*
<mi>h(3)=h'(3)+12</mi>. .

2. + Conditional: InCorrectAnswerConditional +

17. **This is the ClipBoard**

The cut and paste window will reload. Close the dialog box that appears. If you click on the button corresponding to the Button Action, the items should be placed as in the screen shot below.



16. – CM_ParentObject.CheckAnswerButton_ButtonAction –

Button Action: *CM_ParentObject.CheckAnswerButton_ButtonAction.*

1. **Remove Graph:** *Axes1.Arrow1.*
2. **Remove Graph:** *Axes1.Arrow2.*
3. **Remove HTML:** *AnswerParagraph.*
4. + Conditional: CorrectAnswerConditional +
5. + Conditional: InCorrectAnswerConditional +

17. **This is the ClipBoard**

Click the Hide Constructed Page button to return to the main window. Click view to view your page.

The page should work quite well, however we still have DragRoutine executing (which constantly updates Paragraph2) as the points are dragged. We should cut and paste *DragRoutine* and the four *Routine for Dragging* items below it to the Clipboard. This is left as an exercise for you. You may also notice that sometimes the points (3,h(3) and (3,h′(3)) do not appear on the graph because the values h(3) and h′(3) are outside the y values that are displayed on the graph. In this case, the arrows will be missing or incompletely drawn. We could fix this by creating a conditional that checks whether h(3) and h′(3) are between -15 and 25 and placing the commands to draw the arrows inside this conditional. That is also left to you for an exercise.

Congratulations on completing this tutorial! Save your page!