Creating HTML5 Apps with Geometry Expressions

A Guide to Making Interactive Math Apps

Marissa Miller
6/9/2014

CONTENTS

INTRODUCTION

WHY HTML5/JAVASCRIPT APPS?

First, there were Java Applets. In the 1990's at Saltire Software, we had an experimental geometry tool which created applets and inserted them in web pages. There are two main problems with Java applets. First, users of the applet must have the Java runtime environment on their computers. This is not a problem for most computers, but smart phones and tablets typically do not have Java enabled. The second problem is that creators of the app must have a Java compiler. These problems were sufficient to persuade us not to commercialize our Java Applet generation technology.

The advent of JavaScript removed both of these obstacles. While Java applets would have required additional plug-ins, HTML5/JavaScript apps run inside the browser and without any additional plug-ins, allowing them to work well on computers, smart phones, and tablets. Further, JavaScript source code is embedded directly in the web page, so no compilation is necessary.

All this means that by using Geometry Expressions, you can create a web page with an interactive JavaScript app with the click of a button!

WHAT IS THIS GUIDE GOING TO COVER?

Essentially, this is a detailed guide on how to use Geometry Expressions software to create HTML5/JavaScript Apps. This guide will begin by giving you a very basic explanation of how to make an app. Then, we will look at the how-to of making an HTML5/JavaScript app in greater detail, including explanations of the various features that can be used in your app. This guide will also feature a section on our app sharing website, Euclid's Muse.

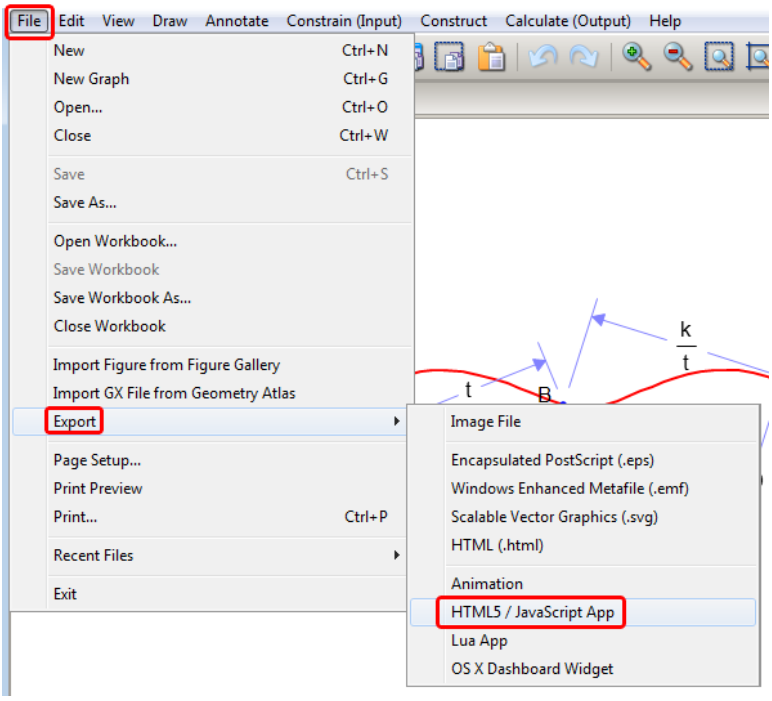
Following that is a section containing a number of example apps that you can explore to get a better understanding of the types apps you can create with the various features that Geometry Expressions offers. Further, you will find a section containing an assortment of Tips and Tricks that you may find useful when creating your apps.

The last two sections are the More Resources section and the Index. In More Resources you can learn about various other sources that you may find helpful while using Geometry Expressions, while creating apps, or while using Euclid's Muse.

Now that you know what this guide is about, let's start learning about apps!

THE BASICS

The first step in making your interactive math app is to acquire the Geometry Expressions software (GX). If you do not already have the software, you can purchase it from geometryexpressions.com. Once you have GX you can start creating your app using the drawing, constraint, construction, and calculation tools. We will talk about many of these tools in more detail later in this guide.



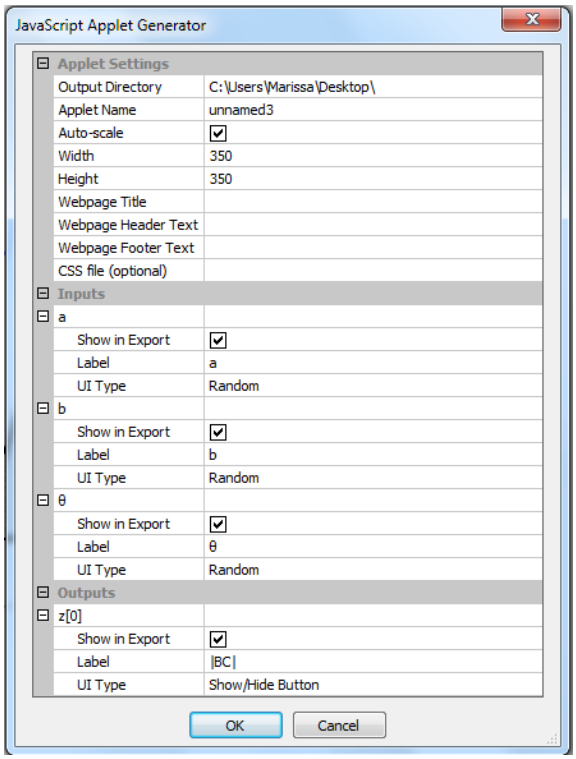
When you have a GX file ready that you would like to turn into an app, simply go to File > Export > HTML5/JavaScript App. This will bring up the JavaScript Applet Generator panel. The JavaScript Applet Generator dialog lets you specify a variety of options for your app, including:

- Whether the app should rescale when the user changes an input
- Which inputs the user should be able to modify, and the UI Type that the user will use to control the inputs
- Which outputs to display, and what text label to display with the output
- The title of the web page
- Text for the web page, both before and after the app

Once you have finished chosing your options in the JavaScript Applet Generator, you can press “OK” and your app will be created! You can open your app in any web browser to see it in action.

JAVASCRIPT APPLET GENERATOR

Now let's take a closer look at the JavaScript Applet Generator dialog. This dialog is where you can specify the various features of your HTML5 app. Listed below are brief explanations of the options found in the JavaScript Applet Generator dialog:



- *Output Directory* - tells GX where to put the files. It will create an html file in the specified directory. You can bring up the file in a browser to see the app.
- *Applet Name* - the name of the html file.
- *Auto-scale* - when the box is checked, the JavaScript applet automatically rescales the drawing when the user changes the values of the inputs. When unchecked, you can click and drag a rectangle around the drawing after you click *OK* to choose the area to be displayed in your app.
- *Width and Height* - specify the size of the drawing on the html page.
- *Webpage Title, Webpage Header Text, and Webpage Footer Text* - enter title text, text above the applet, and text below the applet.
- *CSS File* - attach a CSS file to format your app's webpage.
- *Inputs* - lets you choose which variables the user will be able to change, what text label identifies each variable, and what type of control to use for each variable. You can use any variable from your GX model as an input variable.
- *Outputs* - lets you choose which outputs to display, what text label identifies each output, and how the output is displayed. You can use any output defined in your GX model.

TITLES, HEADERS, AND FOOTERS

The webpage title will appear at the top of the webpage, above the app. By default, it will be large and bold text. The webpage header will allow you to add text above the app drawing. The webpage footer allows you to add text below the app drawing.

For both the header and the footer, you have the option to open a separate dialog box to type in your text. To do this, first select the text entry box. You will then see the ... button. Clicking this button will bring up the dialog box.

Note: you can use html code in your header and footer text for additional formatting options.

LABELS AND UI TYPES

For both inputs and outputs, you are able to change the label and the way the element is controlled using the Label and UI Type fields in the JavaScript Applet Generator dialog.

UI TYPES: MAKING YOUR APP INTERACTIVE

In the JavaScript Applet Generator, you can choose which inputs and outputs to include in your app's display. You have a variety of options that you can use to change the way these elements are displayed and controlled. These methods of control are called UI Types.

There are seven UI Types that can be used to control the input variables. These UI Types include the following: Slider, Text box, Advance Button, Media Controls, Timer, Random, and Draggable. Note that not all UI Types are compatible with all the variable types. For functions, you only have the option of using a Text Box or a Multiline Text Box.

There are two UI Types that can be used to control the outputs. These UI Types are Plain Text and Show/Hide Button.

Most of the UI Types get their range from the Animation values specified in the Variables panel of your GX model. For instance, if you choose to move a point with the Slider, the point will not be able to move outside of the range specified by the Animation values.

SLIDER

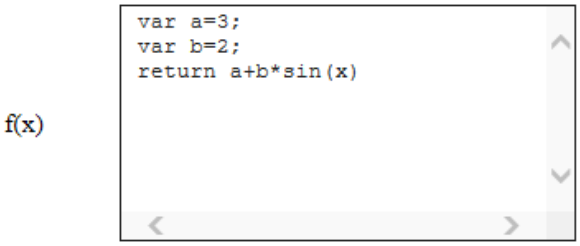
The Slider has several unique features. The first is that you can move the Slider to different positions to manually change the value of the variable. The range of the values will be displayed on either end of the Slider, and the current value will be in the middle of the Slider.

Another useful feature of the Slider is the Start/Stop Button. This button will allow you to see your app in motion. Pressing the button once will start the animation of that variable, starting at the beginning of the range and moving towards the end of the range. Pressing the button again while the animation is occurring will stop the animation.

TEXT BOX AND MULTILINE TEXT BOX

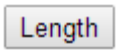
The Text Box will allow the user type in any numeric value for their variable or function.

For a function, there is also the option to use a Multiline Text Box. If you use the Multiline Text Box, remember to use the JavaScript return statement to display the function.



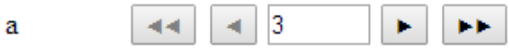
ADVANCE BUTTON

The Advance Button displays the variable label on a button. Clicking the button will increment the variable. If you used decimals in GX for your range, they will be rounded to whole numbers. Once the end of the range is reached, pressing the button again will return to the beginning of the range.



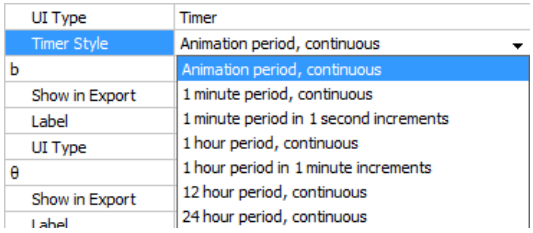
MEDIA CONTROLS

The Media Controls option is similar to the Advance Button, but offers a couple more features. For values within the specified range, you can increment or decrement, skip to the beginning or the end of the range, or type in a specific value, (as in the Text Box mode).



TIMER

The Timer is useful for making Clock Apps or apps with continuous animation. There are seven timer styles, pictured below.



The hours, minutes, and seconds are taken from your computer's internal clock.

RANDOM

This UI Type gives the specified variable a random value within the range set by the Animation values in the Variables toolbox. To change the variable's value to a different random value, press the reload button  , or reload the page.

DRAGGABLE

The Draggable UI Type can be used on any point that has been constrained using the Coordinates constraint or the Point Proportional constraint. These two constraint types each allow different types of movement for the constrained points.

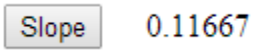
If the Draggable point is constrained by the Coordinates constraint, then the point can be moved freely about the entire plane. If the point is constrained by the Point Proportional constraint, then the point can only be moved along the curve on which it is constrained.

PLAIN TEXT

The Plain Text UI Type is one of the two ways to display an output measurement. Selecting this UI Type will display the value next to its label.

SHOW/HIDE BOX

The Show/Hide Box is another way to display an output measurement. Selecting this UI Type creates a toggle button that when pressed will either hide or show the value.

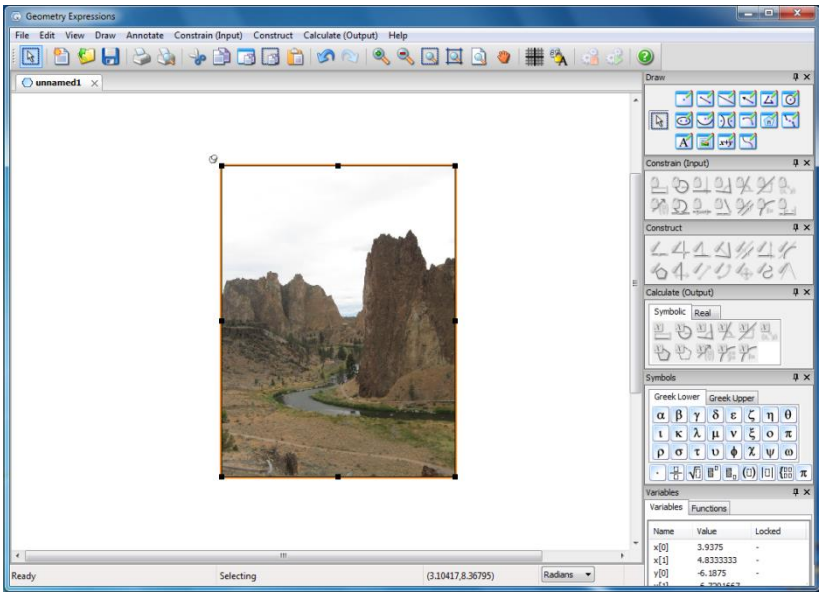


ADDING A PICTURE TO YOUR APP

The Picture tool in GX allows you to add an image to your GX model. This picture will then be displayed in your HTML5 app when exported. Having a picture in your app can make it stand out a little more, or even allow the user of your app to interact with the picture.

Adding a picture to your app is easy. First, select the Picture tool in the Draw toolbox. Then, using the Picture tool, drag a rectangle that is the size you want your picture to be. This will bring up the Select Image File dialog. This is where you can look through your computer's directories and find the image that you would like to insert into your app.

Once you have selected your file and pressed "Open", your picture will be added to GX. From there, you can move the picture around, change its size, and even add other objects to interact with the picture.



To move the picture around, just click and drag somewhere on the picture. To change the size of the picture in the GX file, first click on the picture. Then click and drag on one of the points on the corners or edges of the picture. When resizing the picture, if you want to keep the size of the picture proportional to its current size, simply hold down the shift key and resize the image as above.

If you would like to see an example of an app that uses pictures, check out the section entitled Example: Pictures. There we will make an app that allows the user to measure the slope of various items in the picture.

EUCLID'S MUSE

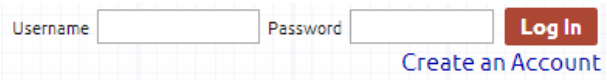
WHAT IS EUCLID'S MUSE?

Euclid's Muse is to Geometry Expressions as YouTube is to a video camera. Euclid's Muse is a website that allows you to upload your own apps, check out other people's apps, and share apps to your favorite social media site. In addition to uploading your interactive HTML5 apps, you can even upload animations and static images that you have created in Geometry Expressions.

Check out what's already on Euclid's Muse by going to euclidsmuse.com, share your favorites, then create a free account and upload some apps of your own.

HOW DO I UPLOAD AN APP?

Before you upload you app, you need an account on Euclid's Muse. To make an account, you can press the "Create an Account" link in the top right corner. Follow the instructions and fill in the appropriate information.



Now that you have made your account, you can start uploading you apps! To upload an app, make sure you are logged in. Then you can either press the "Upload" button found in the navigation bar, or you can press the "Upload" button that is on the Home page. Both buttons will bring you to the first Upload screen. This is where you select the file of the app that you want to upload, as well as choosing the license type and visibility. You can also choose to include the GX file for your app, or even include a TI-Nspire™ version of your app. Once you're ready, you can press "Submit" to move on to the next step.

The second Upload screen allows you to change the details of the app, such as its name, the description, and the tags that will help people find your app when they search Euclid's Muse. You also have the option to append your app to a collection. Once you are ready, press "Save" and your app should be uploaded.

WHAT IS A COLLECTION?

A collection is a way to group multiple apps together on Euclid's Muse. To create a collection, mouse over Collections in the navigation bar, then click the "Create New" button. This will bring you to the "Create New Collection" screen where you will be able to give a name, description, and tags to your collection. You can also choose the visibility of the collection, as well as what apps to include in the collection, (you can always add more apps to the collection later). Once you are ready, press save and your collection will be created.

HOW DO I ACCESS MY UPLOADS AND COLLECTIONS?

If you would like to access your uploads or collections, click the "My Uploads" or "My Collections" link in the top right corner. "My Uploads" will bring you to a page that shows thumbnails of all of your uploaded apps. Clicking one of them will bring you to that app. "My Collections" will bring you to a list of all of your collections. Clicking one of your collections will bring you to that collection.

HOW DO I EDIT MY UPLOADS AND COLLECTIONS?

You can edit your uploads by clicking “My Uploads,” then selecting the app you would like to edit. Then in the description box on the right, click the “Edit App” button. To edit a collection, click “My Collections,” and select the collection you want to edit.

Then click the “Edit Collection” button in the upper right. Note that editing your collection will allow you to add more apps to your collection. Adding an app to your collection is as easy as copy and pasting the app’s link into the “Apps in Collection” box.

EXAMPLE APPS

There are many options that GX gives for making your apps interactive. We want you to get an idea of some of the types of apps you can create and how you can use the features that GX offers. This section contains a number of examples that demonstrate how each of the UI Types can be used in various apps. Some of these apps also showcase some tricks that you may find useful while creating apps.

The following examples give a detailed explanation of how the apps were created so that you can try to recreate them if you would like. If you would like to see the examples in action or to download the GX files, make your way to euclidsmuse.com to see the finished apps, (see More Resources).

PREVIEW OF EXAMPLES

Below is a table that will give you a brief preview of each of the example apps and what topics they focus on so that you can decide if there is a specific example you would like to skip to.

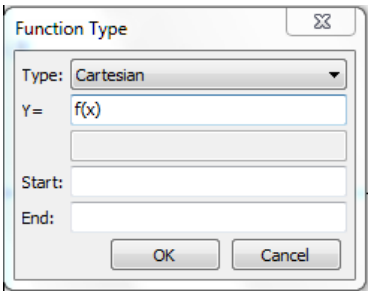
Example Name and Page Number	Description of App	UI Types Used	Other Features Used
Graphical Calculus Practice App, **13**	Test how well you know your derivatives with a graphical representation	Slider, Text Box	Generic Functions
Altitude of a Triangle App, **14**	See how different side lengths of a triangle affect its altitude	Text Box, Plain Text	Symbolic Calculations, Distance/Length Constraints
Euclid's Equilateral Triangle, **15**	Step through the construction of an equilateral triangle from a line segment	Advance Button	Visibility Conditions
A Second Look at Euclid's Equilateral Triangle, **16**	Explore the steps to construct an equilateral triangle from a line segment	Media Controls	Visibility Conditions
Clock App, **17**	A basic analog-type clock	Timer	Point Proportional Constraints
Cosine Rule Assessment App, **18**	Test how well you know the Cosine Rule by calculating side lengths of random triangles	Random, Show/Hide Button, Plain Text	Expressions, Distance/Length Constraints, Angle Constraints
Area of a Triangle App, **19**	Explore how the shape of a triangle affects it area	Draggable, Plain Text	Symbolic Calculations, Coordinate Constraints
Pictures, **20**	Calculate the slope of various features in a landscape picture	Draggable, Show/Hide Button	Pictures, Coordinate Constraints
Draggable Point on a Locus, **21**	See how a locus can be used to avoid Constraint Conflicts	Draggable	Locus Construction, Point Proportional Constraint
Drawing an Ellipse, **22**	Creating an ellipse from a locus	Draggable	Locus Construction, Point Proportional Constraint, Arc Tool

EXAMPLE: GRAPHICAL CALCULUS PRACTICE APP

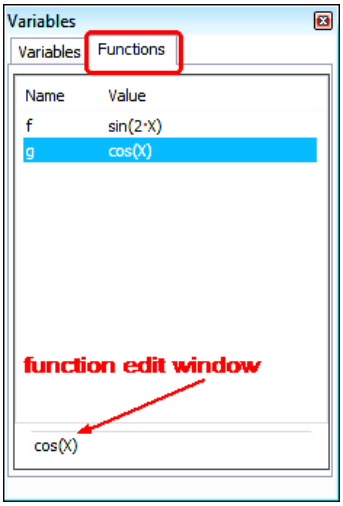
Generic functions are a powerful feature of Geometry Expressions. We will explore how to use this feature by creating a graphical calculus practice application. We will also see a unique way that the Slider UI Type can be used in our app.

The app that we are going to create will present two functions, f(x) and g(x). The user will be prompted to enter a function in for f(x) and the derivative of the function for g(x). Then using the animation feature of the Slider, the user will be able to check that their derivative is correct.

First, we turn on the axes  . Then we want to create our first function f(x). We use the Function tool in the Draw toolbox  . This will bring up a dialog box that will allow you to enter a function. Our function will be kept as the generic function f(x) so that its value can be changed by the user.

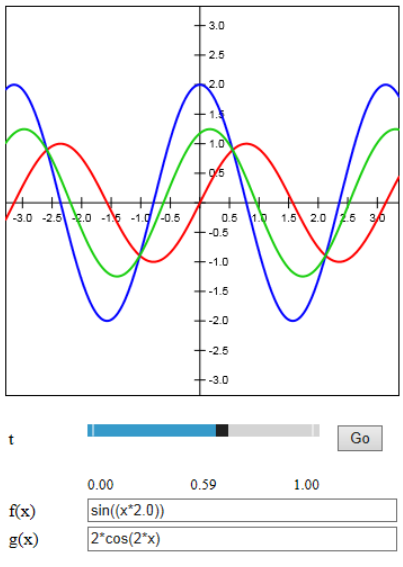


We also want to create a second generic function that the user will be able to set as the derivative of f(x). Again we use the Function tool, and this time we set the function to g(x).



Since f(x) and g(x) are generic functions, the user will be able to change them in the app. But we need them to have starting values. To do this, go to the Variables toolbox and open the Functions tab. Then select f(x) and replace the default function with one of your own, for example sin(2*x). Then select g(x) and replace it with a different function.

Next, we are going to create our answer function. Again, we use the Function tool, and we set the function in the Y= window to (1-t)*f(x)+t*f'(x). This will create a new variable *t* that the user will be able to animate with the Slider to see f(x) morph into its derivative. Make sure to set the Animation Values of *t* to go from 0 to 1 in the Variables toolbox. This way, when *t*=0 the third function evaluates to f(x), when *t*=1 the function evaluates to f'(x).

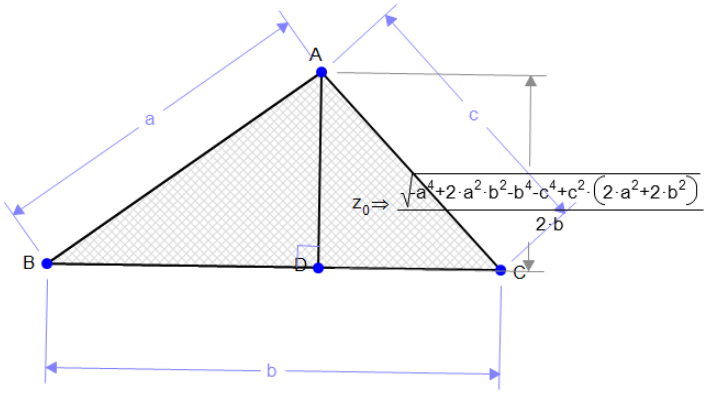


We can now export the app by going to File > Export > HTML5/JavaScript App. The user should be able to see the function animate, so we use the Slider UI Type for *t*. We keep the UI Type for f(x) and g(x) as a Text Box. Press "OK" when you're ready to finish the export.

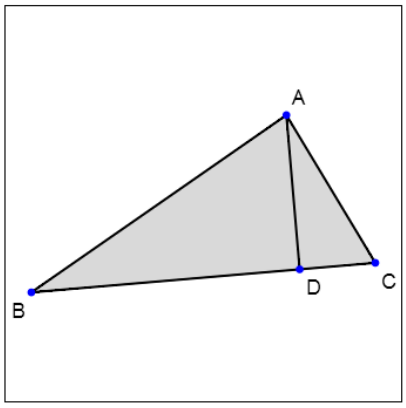
You also might find it helpful to change the colors of the functions so that they are easier to distinguish from one another. To learn more about changing colors, see the Tips and Tricks section of this guide.

EXAMPLE: ALTITUDE OF A TRIANGLE APP

In this app, we will see how different lengths of the sides of a triangle affect the altitude of the triangle. This app will make use of the Text Box UI Type so that the user can enter whatever values they want for the sides of the triangle.



To begin making this app, use the Polygon tool in the Draw toolbox to draw a triangle. Then use the Distance/Length constraint to give each side of the triangle a unique variable for its side length.



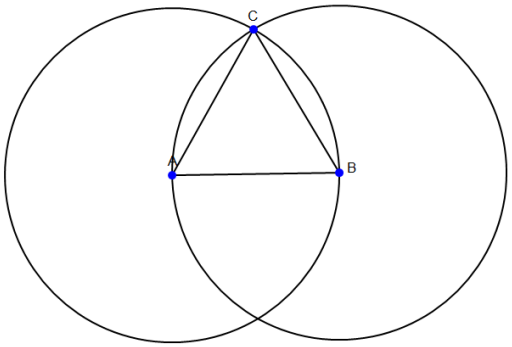
a	9
b	10
c	5
z[0]	4.4900

Next, using the Line segment tool, draw the altitude of the triangle. Make the line segment perpendicular to the edge of the triangle by using the Perpendicular constraint. Then use the Symbolic Length calculation on the altitude line segment so that we can show the length of the altitude as an output in the app.

Now we can export the app by going to File > Export > HTML5/JavaScript App. Set the UI Type for each side length to Text Box. Make sure that the UI Type for the length of the altitude is set as Plain Text. Press “OK” when you are ready to export the app. The user can now enter various values for the lengths of the triangle’s sides and will see the resulting length of the altitude.

EXAMPLE: EUCLID'S EQUILATERAL TRIANGLE

In this example, we will use the Advance Button UI Type and Visibility Condition to demonstrate Proposition 1 from Book 1 of Euclid's Elements. This proposition describes how to draw an equilateral triangle from a single, straight line segment.

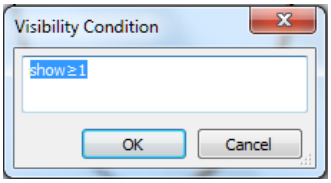


We start by using the Line Segment tool from the Draw toolbox to draw a line segment that is close to horizontal.

Then, using the Circle tool from the Draw toolbox  , we draw a circle that is centered on one end of the line segment and has the line segment as its radius. Draw another circle centered on the other end of the line segment, and also with the line segment as its radius, (see image above).

Next, use the Line Segment tool again to connect each end of the original line segment to one of the intersection points of the two circles, (you will be drawing two more line segments in total).

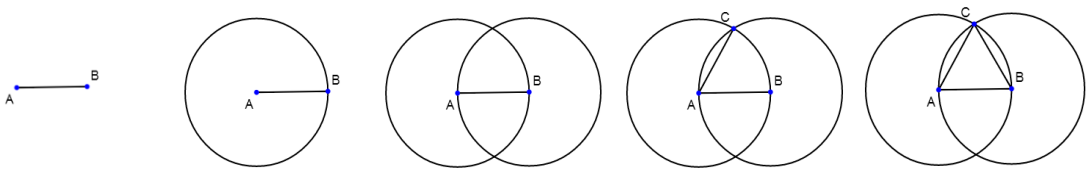
Your model is almost complete. The next step is to set the Visibility Condition so that we can step through each of the steps of creating the equilateral triangle. We want the original line segment to always be there, so we won't change its Visibility Condition. However, we do want the circles and the other two line segments to appear one by one to simulate the construction process.



First we will change the Visibility Condition of one of the circles. To do this, click on the circle to select it, then right click to bring up the Selection Context menu. Select Visibility Condition from the menu. This will bring up a dialog box that allows you to enter your desired conditions. For this circle, we are going to set the Visibility Condition to "show >=1". This will create a new variable called *show,* which you should be able to see in the Variables toolbox. If you change the value of *show* to be less than 1, you will see the circle disappear. If you make *show* greater than or equal to 1, the circle will appear again.

You can now set the visibility condition for the other circle and the two other line segments to "show >= 2", "show >= 3", and "show >=4", respectively. Make sure that you set the range of *show* in the GX Animation panel to go from 0 to 4. Also, before exporting, make sure the current value of *show* is 0 so the construction will begin at the first step.

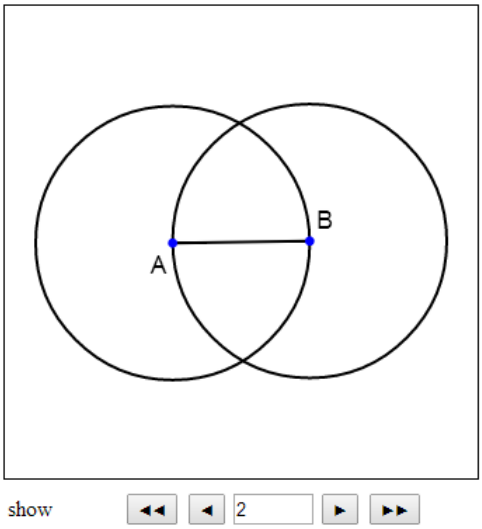
The last step is to open the JavaScript Applet Generator by going to File > Export > HTML/JavaScript App. Then change the UI Type of *show* to Advance Button. You're now done, and can export your app when you're ready. In your app, each time the user presses the advance button, they will see another step of the triangle's construction.



EXAMPLE: A SECOND LOOK AT EUCLID'S EQUILATERAL TRIANGLE

In the previous section, we looked at how to create an app that shows the steps of constructing an equilateral triangle, based on Proposition 1 from Book 1 of Euclid's Elements. The app created in that section used the Advance Button UI Type to move through each step of the construction process.

In this example, we will show the same construction of the equilateral triangle. However, this version will use the Media Controls UI Type instead of the Advance Button.



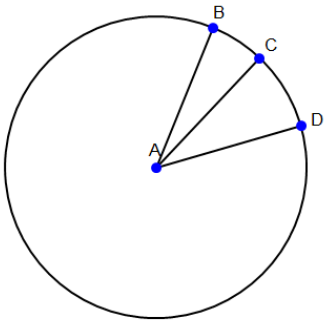
The Media Controls are similar to the Advance Button in that they increment a specific variable. But the Media Controls also allow you to decrement the variable, skip to the beginning or end of the range, or even enter a specific value, similarly to the Text Box UI Type.

For this type of app, the Media Controls can be useful because it can allow the user to move forwards or backwards in the construction process, start over, see the final construction, or go to a specific step of the construction process.

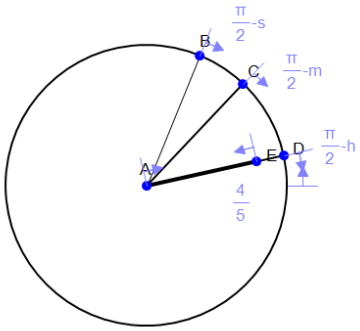
To make this app, follow the instructions in the previous section, entitled Example: Euclid's Equilateral Triangle, except instead of the Advance Button, set the UI Type of the variable *show* to Media Controls.

EXAMPLE: CLOCK APP

The Timer UI Type in GX makes it very easy to create a Clock App. Here we will walk through how to create a simple Clock App using GX.



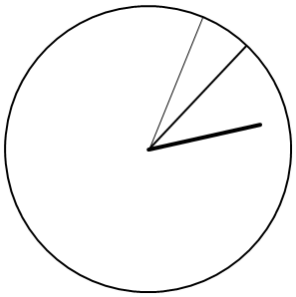
First, we use the Circle tool and the Line Segment tool from the Draw Toolbox to construct the basics of the clock. We start by drawing a circle for the basic clock shape and three lines that will represent the three hands of the clock.



We have also made another line segment on top of line AD that is 4/5 the length of AD, (using the Point Proportional tool). This will be our hour hand.

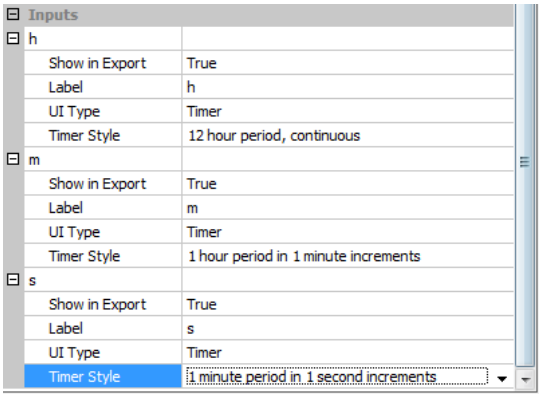
To make it a little more obvious which hands are which, you can change the thickness of each line segment from the Selection Context menu > Line Properties. Here we have changed the hour hand to thickness 3, the second hand to thickness 1, and left the minute hand at thickness 2.

Now we want the clock hands to be able to move clockwise around the circle. To do this, we want to use the Point Proportional tool  to constrain the three endpoints of the hands to the circle. We introduce three new variables: *s*, *m*, and *h*, (seconds, minutes, and hours). We constrain the hour hand endpoint with the proportion π/2-*h*; the minute hand endpoint with the proportion π/2-*m*; and the second hand endpoint with the proportion π/2-*s*. These constraints will allow the clock hands to move in a clockwise direction starting from the top of the clock. This is because the Point Proportional command for a circle is defined in radians with 0 starting at 3:00, proceeding counter clockwise.



Be sure to check that the Animation Values for *s, m,* and *h* each range from 0 to 6.283, (2π), so that the hands will travel all the way around the clock.

Most of the work on your GX model is now done. All that's left is to clean it up a little and then export it to an HTML5/JavaScript App. To clean it up, we will hide the objects and labels that we don't want to appear on our final app. Such as the points and the guide for the hour hand. You can also change to Arrowheads on the end of the hands, all from the (right click) Selection Context menu.

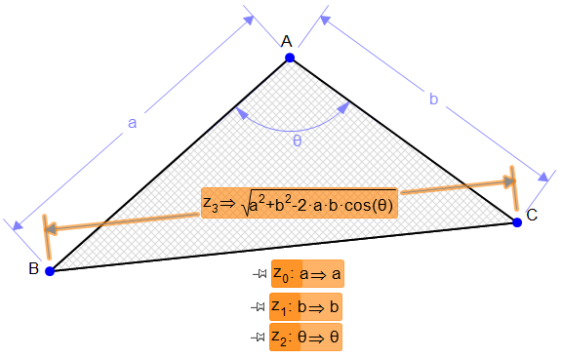


The last step is to set the timing for the clock. This is done in the JavaScript Applet Generator dialog. For each of the Input variables, (*s, m,* and *h*), we change the UI Type to Timer. Then, for the hour hand, we change the Timer Style of *h* to "12 hour period, continuous," as we are making a 12 hour clock. For the minute hand, we change the Timer Style of *m* to "1hour period in 1 minute increments." For the second hand, we change the Timer Style to "1 minute period in 1 second increments."

When you've changed the rest of your desired settings, you can press "OK" to make your Clock App!

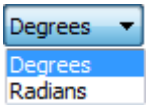
EXAMPLE: COSINE RULE ASSESSMENT APP

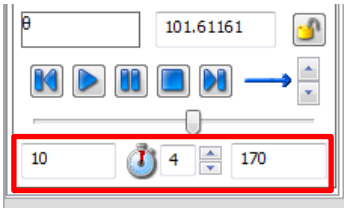
In this example, we will look at how you can use the Random UI Type, as well as the Show/Hide Button, to create an assessment style app. This app will generate a triangle with two random side lengths and a random angle between those sides, with the ability to generate new values using the reload button. Then the user can practice using the Cosine Rule and check their answer.



The first step is to create some arbitrary triangle using the Polygon tool from the Draw toolbox.

Next, use the Distance/Length constraint on two of the sides of the triangle. Make sure to assign them both variable names so that the variables can be randomized when we create the app. You will then want to use the Angle constraint on the angle between the two constrained sides. Note that for this app we have put GX in degree mode (the window at the bottom right of your screen.





We also want to set the range for the constraints because we don't want the sides to be too long or too short, and we don't want the angle to be less than 1 or more than 180 degrees. To set the ranges, we go to the Variables toolbox and select one of the variables. You can enter a start and a stop value in the Animation values boxes. For the two variables representing the lengths of the sides, we will put the range from 3 to 15. For the angle, we will go from 10 to 170 degrees.

Now that all the necessary constraints have been added, we can add our outputs. We want the user to know the lengths of the two constrained sides as well as the angle between them, but the Random UI type does not display the values of the variables. Instead, we will add three Expressions from the Draw toolbox. Set each of these expressions equal to one of these three variables.

Inputs	
a	
Show in Export	☑
Label	|AB|
UI Type	Random
b	
Show in Export	☑
Label	|AC|
UI Type	Random
θ	
Show in Export	☑
Label	Angle
UI Type	Random
Outputs	
z[3]	
Show in Export	☑
Label	|BC|
UI Type	Show/Hide Button
z[0]	
Show in Export	☑
Label	|AB|
UI Type	Plain Text
z[1]	
Show in Export	☑
Label	|AC|
UI Type	Plain Text
z[2]	
Show in Export	☑
Label	Angle
UI Type	Plain Text

We also want the user to be able to toggle the length of the third side, so we want to use the Symbolic Distance/Length calculation from the Calculate toolbox on the third side.

Now we can export our app by going to File > Export > HTML5/JavaScript App. Since we want the triangle to be random each time, we want to set the angle variable and the side length variables to Random UI Type. We will leave the output variables all as Plain Text except for the third side length, which we will set to the Show/Hide Button UI Type.

Don't forget that you can change any of the labels or the other Applet Settings. Once you're done, you can press OK and your app will be done! Now a user can generate random triangles and practice using the Cosine Rule.

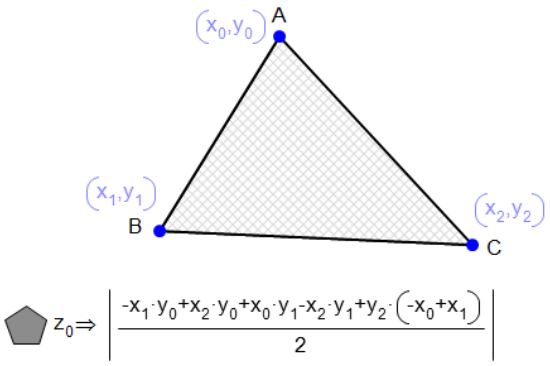
EXAMPLE: AREA OF A TRIANGLE APP

In this example we will make a simple app that shows the area of a triangle with adjustable side lengths. This app will use the Draggable UI Type to allow the user to move the corners of the triangle.

To begin, use the Polygon tool in the Draw toolbox to draw a triangle.

Since we want the vertices of the triangle to be Draggable across the entire coordinate plane, we will use the Coordinate constraint on each of the vertices. By leaving the coordinates as arbitrary variables, we can make the points draggable in the JavaScript Applet Generator.

We also want to output the area of the triangle, so we click inside the triangle to select it and use the Symbolic Area calculation in the Calculate toolbox.



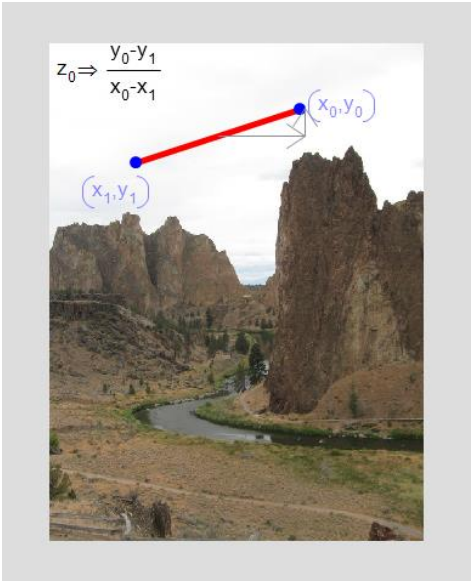
Inputs	
x[0], y[0]	
Draggable	☑
x[1], y[1]	
Draggable	☑
x[2], y[2]	
Draggable	☑
Outputs	
z[0]	
Show in Export	☑
Label	Area
UI Type	Plain Text

The GX model is now finished. Now we can open the JavaScript Applet Generator dialog by going to File > Export > HTML5/JavaScript App. We will keep the UI Type of the vertices as Draggable. We will make sure to keep the area output as visible (check the Show in Export box) during export, and we will leave its UI Type as Plain Text.

Once you have changed any other desired Applet Settings, (such as the title, header, or footer), then you can press “OK” to export your app.

EXAMPLE: PICTURES

Here we are going to take a look at how to create an app that uses pictures. This app is going to allow the user to measure the slope of different objects in a picture. To begin making this app, follow the instructions in the Adding a Picture to Your App section of this guide, and add your picture to the GX model. For our app, we chose a picture of some hills and cliffs.



Next, we want to create a way for the user to measure slopes. To do this, we can use the Line Segment tool in the Draw toolbox to add a Line Segment to our model. This segment will be what the user uses to measure their slopes.

We want the user to be able move the two endpoints of the line, so we want these points to be Draggable. To do this, we will constrain points A and B using the Coordinate Constraint, and we will leave the coordinates as arbitrary variables. This will allow the user to move the points freely across the coordinate plane when we create our app.

Then, we can make some changes to make the app look a little nicer. First, we can change the color of the line segment and make it thicker to help it show up a little better over the darker parts of the picture. Also, we can hide the labels for A and B, as they are not necessary. Consult the Tips and Tricks for more information on these style changes.

We also want our user to be able to see the value of the slope. To output this value, select the line segment, then go to the Calculate toolbox, and select the Symbolic Slope calculation  , (only Symbolic measurements can be displayed in the app). Now you have created a value for the slope to display in our app.

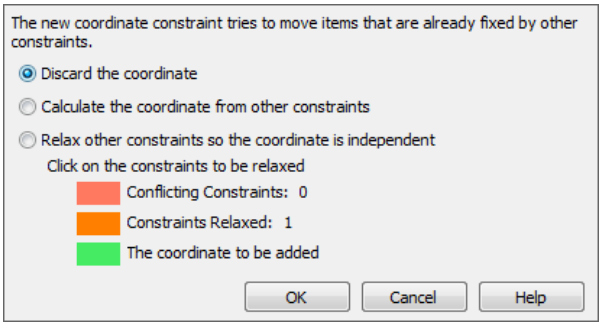
Finally, we need to export our app. Open the JavaScript Applet Generator dialog by selecting File > Export > JavaScript/HTML5 App. You want to leave the UI Type for the coordinate inputs as Draggable.

You then have the option of whether you want to display the slope as Plain Text or as a Show/Hide Button, (see the section entitled UI Types: Making Your App Interactive). In our case, we chose to display the slope as a Hide/Show Button so that the user could hide the slope if they want to try and guess the value first. Also, remember that you can change the labels on each variable!

Once you are ready, you can press OK to create your app. Note that the app pictured to the left also uses a CSS file to alter the layout, (see Tips and Tricks).

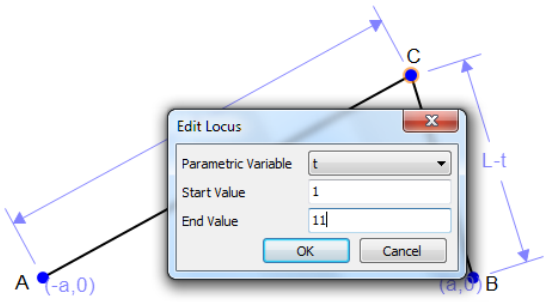
EXAMPLE: DRAGGABLE POINT ON A LOCUS

You may have noticed that sometimes you may want to make a point Draggable, (by adding either the Coordinate constraint or the Point Proportional constraint), but end up getting a Constraint Conflict. It may seem like you just cannot add a Draggable point where you want it. In this example, we will see how using Loci and the Point Proportional constraint we can create a Draggable point even when it seems like we would have a Constraint Conflict.



To start off this model, we are going to create two line segments that both share one endpoint. Then we are going to add a few constraints so that the two line segments are completely constrained. The first constraint we will add is a Distance/Length constraint on one of the line segments; we will constrain it to a variable *t*. Then we will use the Distance/Length constraint on the other line segment and constrain it to *L-t,* which will create another variable *L*.

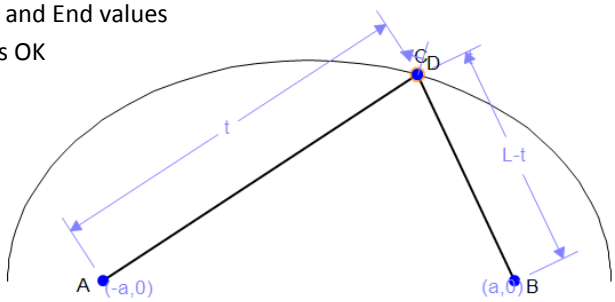
Next we will constrain the independent endpoint of each line segment using the Coordinate constraint. One will be constrained to *(-a,0)* and the other will be constrained to *(a,0)*. This will create another variable *a*.



Note that you will want to lock the variables *L* and *a* so that you can move the GX model around without all of the geometry changing, (see Tips and Tricks for more on the Lock Tool).

Now we want the user to be able to drag the shared endpoint of the line segments and move it. So it seems like we would want to use the Coordinate constraint on this point as well. However, trying to add that constraint will result in a Constraint Conflict as the model is already fully constrained.

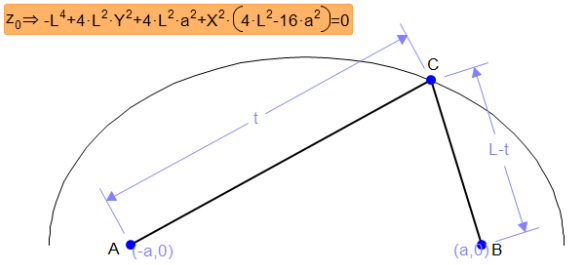
Instead, we can use the Locus tool  from the Construct toolbox. First, select the shared endpoint, and then select the Locus tool. This will bring up the Edit Locus dialog. Make sure that *t* is selected as the parametric variable, and then change the Start Value and End Value to your desired values. We set our Start Value to 1 and our End value to that same value as *L*. Note that the Start and End values must be numeric, *i.e.* you can't use a variable value. Press OK when you're done. This should create your locus.



Now create another point and use the Point Proportional constraint to constrain it to the locus, proportional to *t*. You are now ready to export. Select File > Export > HTML5/JavaScript App to open the JavaScript Applet Generator. Change *t* to Draggable. Now when you open the app in a browser, you will be able to drag the point along the Locus.

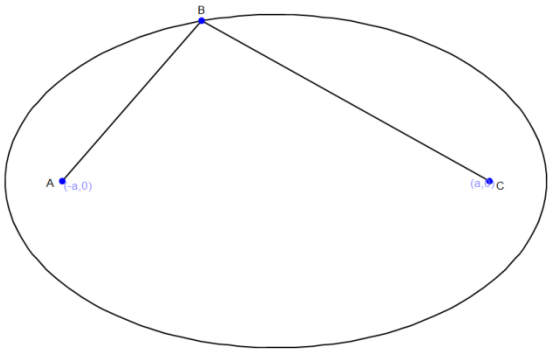
EXAMPLE: DRAWING AN ELLIPSE

This example is an extension of the previous example, "Draggable Point on a Locus." You may have noticed when creating the previous app that the Draggable point can only be dragged a certain distance, (the range of *t*). It also seems like the locus that was drawn is part of an ellipse, so it would be nice if the draggable point could move all the way around the ellipse.



In this app, we are going to see how we can simulate the "two pins and a piece of string" construction of an ellipse, using what we made in the previous app.

To start this app, do everything you did in the previous app, up to adding the last Draggable point. Instead of constraining a point to the locus, we are going to make a new ellipse and have the endpoint move along the ellipse instead of the locus.

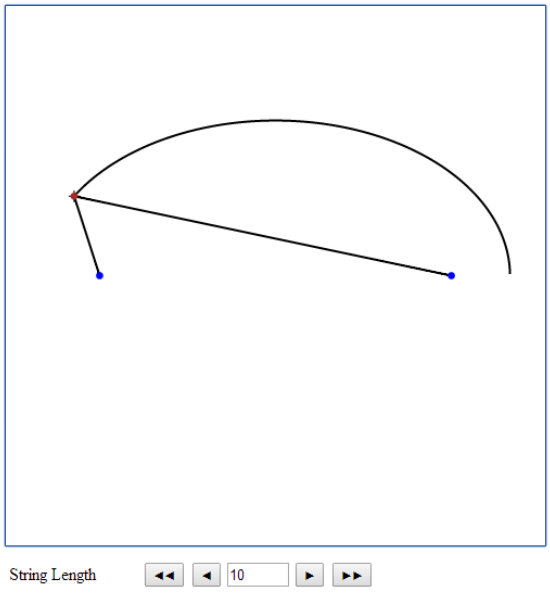


First, create an ellipse using the Ellipse tool from the Draw toolbox  . This ellipse should have the two independent endpoints as its foci and the shared endpoint on the curve. You have now created an ellipse that follows the same path as the locus.

Now we don't even need parts of our original model. We can delete the length constraints for the two line segments. Instead, we can proportionally constrain that shared endpoint to the ellipse with some new variable, for instance *s*. Don't forget to change your Animation Values for *s* so that it can go around the whole ellipse.

To make this look even more like you are drawing the ellipse, we are going to use the Arc tool from the Draw toolbox. We are going to start on some point on the ellipse and end the arc on the shared endpoint of the line segments. Then, we are going to hide the ellipse using the Hide option.

Now when you export the app, make *s* Draggable, (we also set *L* to have Media Controls so the user can change the length of the string). As you drag the point around you will see the ellipse appear as the arc is made longer. Your "two pins and a piece of string" simulation is complete!

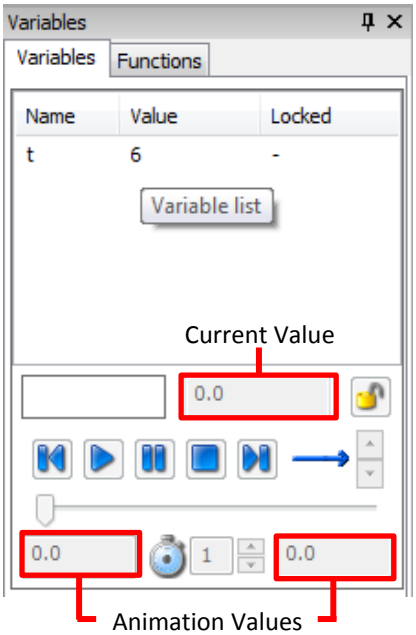


TIPS AND TRICKS

There are some additional techniques and features in GX that are useful to know when making HTML5/JavaScript Apps. The following are a compilation of such techniques that we think you may find helpful.

CURRENT VALUE AND ANIMATION VALUES

The Variables tab of the Variables toolbox has a number of features that you can learn about in more detail by reading the GX Manual, (see More Resources). However, there are two features of the Variables toolbox that we are going to discuss here: the Current Value and the Animation Values.



The Current Value box is the easiest way to change the value of a variable to a specific value. When you export your GX file as an HTML5/JavaScript App, the app the model will be exported in its current position, so changing the Current Value allows you to easily set the variable's starting position in the app.

Additionally, the Animation Values of a specific variable will act as the range of that variable, for most UI Types. For instance, if you were to use the Point Proportional constraint to allow a point to move along some curve, you could make that point Draggable when exported. However, in the app, the point would not be allowed to drag past whatever values you set in the Animation Values. Essentially, changing the Animation Values will define the range of a variable in your app.

LOCKING VALUES

Another feature in the Variables tab of the Variables toolbox is the Lock Tool. By default, when you drag points in a GX model, it will adjust the numerical sample values used in the various parameters of the model to accommodate the drag, as best it can.

However, you may want certain variables and constraints not to change when you change other parts of the model. This is where the Lock Tool comes in. If a variable is locked (+), its value will not change when you move the geometry or add additional constraints. An unlocked (-) variable is free to change as the geometry moves or changes.

DRAGGABLE POINTS

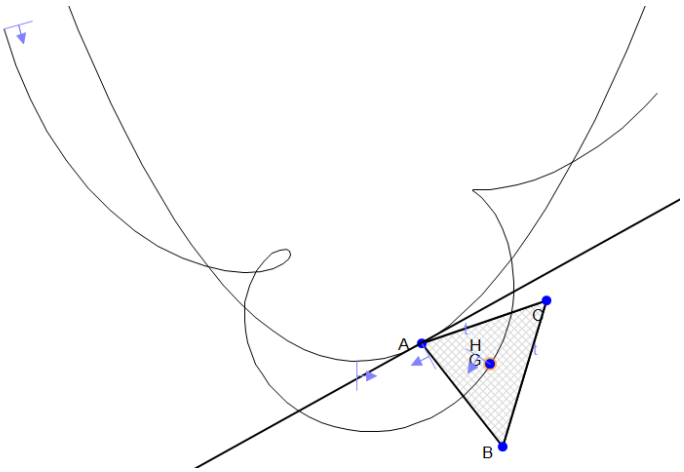
There are two ways to make points Draggable. The first is to use the Coordinate constraint. This constraint allows you to constrain a point to a specific coordinate on the coordinate plane. If you leave the constraint values as variables, you can make the point Draggable when you export the app. This point will be able to move anywhere in the coordinate plane.

The other way to make a point Draggable is to use the Point Proportional constraint. This constraint allows you to proportionally constrain a point to a specific line, curve or conic. If you leave the constraint value as a variable, you

will be able to make the point Draggable when you export the app. This point will be able to move only along the curve where it is constrained and within the range values set in the Point Proportional dialog.

LOCI

Loci can be very useful in a couple of situations when making apps. One such situation is when you may want to make a point Draggable, but can't because the model is already completely constrained. Instead, you could make a locus of the path that you want the point to travel, and then constrain a different point to the locus that is proportional in a way such that it moves the original point, (see Example: Draggable Point on a Locus).

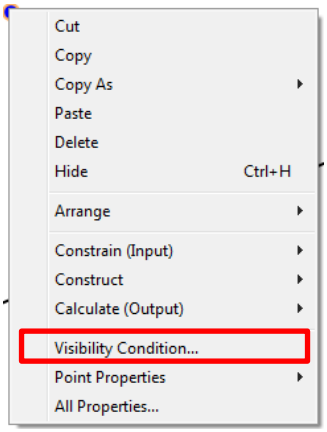


Another way that you could use a locus when making apps is if you want a Draggable point to be at a different location. Let's say you want to trace the movement of a triangle's center rather as a vertex of the triangle moves along another curve. You could find the center point and create a locus of that point in relation to your driving parameter.

Further, one could use a locus along with the implicit equation tool to create a new object with the same shape as the locus. For instance, you may create a locus of a point and find that it looks like a circle, but perhaps the locus only generates part of the circle. You could use the implicit equation tool to get the equation of the locus, verify that it is a circle, and then create a circle that is constrained using the implicit equation of the locus.

VISIBILITY CONDITIONS

The Visibility Conditions are a useful way to dynamically show or hide objects in your app. They essentially allow the visibility of and object to be dependent on the values of some variable(s). An example of how Visibility Conditions could be used in an app can be found in the Examples section of this guide, in both Euclid's Equilateral Triangle and A Second Look at Euclid's Triangle, so we won't go into those details here.



Instead, we will see how exactly to change the Visibility Conditions. To change the Visibility Conditions of a certain object, simply click that object to select it, and then right click to open a pop-up menu. Select Visibility Conditions to open the dialog box. This dialog is where you will enter your conditions, *e.g. |t| >=0 AND |t| <.3* . You can use the Symbols toolbox or your keyboard for absolute values and inequalities.

HIDE/SHOW

You may want some objects or labels in your app to be permanently hidden. To do this, first select the object or label that you want to hide. Then you can either right click and select Hide from the Context menu, or you can go to View > Hide, (in the top menu bar). To show an item that has been hidden, right-click and select Toggle Hidden

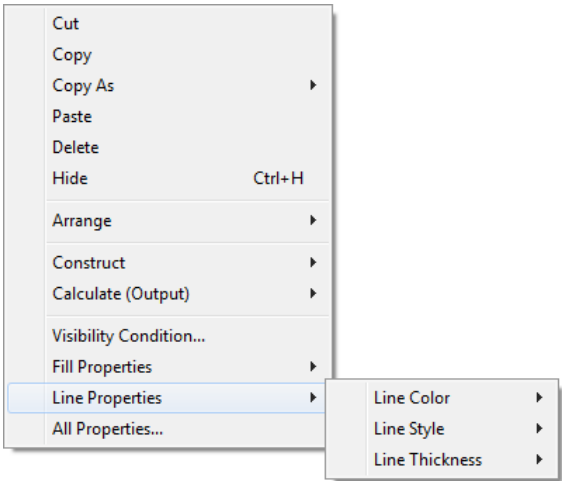
from the general Context menu. The magic wand cursor appears, and any hidden objects appear faintly in the drawing window. A click of the wand will make them show. You can click again to re-hide them. Return to the Selection arrow to exit the Toggle Hidden mode and see your modifications. Alternatively, if you have only hidden on object in your drawing, you can choose Show All from the Context menu or the View menu.

EXPRESSIONS

Expressions can be useful when creating apps in several ways. If you want to display some calculation based on values from the model, you could use the Expression tool in the Draw toolbox to write this equation. Then when exporting your app, you can choose to show that Expression in the JavaScript Applet Generator.

Another example of when you may want to use Expressions in an app can be found in the Examples section of this guide, (see Examples: Cosine Rule Assessment App).

PROPERTIES



For some apps, you may want to change the colors, thickness, or fill properties of objects or properties axes. The easiest way to do these things is to first select the object that you want to change, and then right click. This will bring up a menu where you will find a number of options, depending on the object selected. The options can include Fill Properties, Line Properties, Point Properties, Text Properties (for apps, this would only apply to point labels) and All Properties. All Properties will bring up a dialog box where you can access all the properties of that specific object including additional color selections. You can also go to Edit > Properties to open the dialog box.

AUTO-SCALE

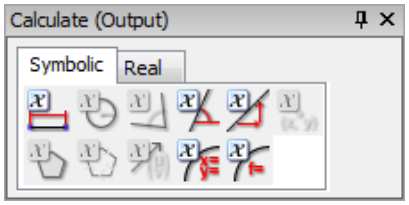
In the JavaScript Applet Generator you will find an option that says Auto-Scale, which can be toggled on or off with a check box. Using Auto-Scale and turning it off are both useful in different situations. When Auto-Scale is turned on, the JavaScript applet automatically rescales the drawing when the user changes the value of one of the inputs. When turned off, you click-and-drag a rectangle around the drawing to select the part of the model that you want to include in the app.

Using Auto-Scale is useful when your model is going to be able to change size because Auto-Scale will automatically change the scale to fit the model. Turning Auto-Scale off is useful when your model is not going to change much, if at all, so that you can keep that model in view. There are other reasons that you may want to turn Auto-Scale on or off, but these are some of the usual reasons.

SYMBOLIC AND REAL CALCULATIONS

The Real Calculation tools are useful if you need to know the exact value of some measurement. Note that these values are only internally consistent. They do not represent any specific units. So the scale of your drawing is set by the first numerical value you enter.

The Symbolic Calculation tools can be used when making your model if you need to know how a certain measurement relates to the rest of the model. These calculations will be given with respect to other variables in the model. If you haven't supplied all of the necessary input constraints for an output calculation , the system inserts any missing variables automatically.



Also it should be noted that only Symbolic calculations can be used to display calculations in your apps. However, the app will only display the numeric value of the measurement.

CSS FILES

When you export a JavaScript app from Geometry Expressions, you get a particular default layout of the elements of the app. You can alter this layout, however, by specifying a CSS file (Cascading Style Sheet) in the JavaScript applet Generator dialog.

For example, a CSS which positions every element in the center of the window, rather than left justified would look like this:

```
#gxInputOutput {
   margin-left: auto;
   margin-right: auto;
}
#gxCanvas {
   padding: 5px;
   text-align: center;
}
#gxHeader {
   text-align: center;
}
#gxFooter {
   text-align: center;
}
```

Try out different CSS files to give your app a unique look!

MORE RESOURCES

There are several places that you can go to get more information regarding Geometry Expressions in general, making HTML5 Apps, and Euclid's Muse.

GENERAL RESOURCES FOR GEOMETRY EXPRESSIONS

If you need more information on the features of GX, the Help button in GX would be a good place to start. This will bring up the GX Manual. The Manual can also be found on geometryexpressions.com under Download > Documentation and in your installation under the Doc subdirectory.

The Geometry Expressions website provides a number of other useful resources. Also under Download > Documentation you will find several other documents that will give more information on various aspects of the GX software, as well as some applications of the software.

Another resource that can be found on the Geometry Expressions website is under the Explore heading. Here you can find several more documents about GX and its applications including activities, projects, and examples; our newsletters; a geometry atlas; the Journal of Symbolic Geometry; and a link to our YouTube channel.

Our newsletters give updates on our products, but also feature various examples, challenges, and projects using GX. There are even some newsletters that focus only on creating HTML5/JavaScript apps.

Our YouTube channel includes a number of Gx playlists. These playlists include tutorials for the GX software, tutorials for the creation of HTML5/JavaScript apps, and interesting math problems illuminated with GX.

Also on the Geometry Expressions website is a link that allows you to e-mail us for support. You can find this under Support > E-mail.

RESOURCES ON EUCLID'S MUSE

There are several resources on Euclid's Muse that you can go to if you need help or inspiration when creating your apps.

The first place to look if you need help is the Help section on euclidsmuse.com. There you can find some basic information on app creation and what can be done with apps on Euclid's Muse.

You can also go to the Forums portion of Euclid's Muse. There you can ask questions, find answers to questions, or get inspiration for different apps.

Another way to get inspiration is to look through the apps that have been created by other users on Euclid's Muse. You can browse through random apps, apps with GX files attached, apps with TI-Nspire™ versions, or use the search bar to find specific types of apps.

Lastly, we have created a collection on Euclid's Muse that contains all of the example apps that have been explored in this guide. They can be found by going to euclidsmuse.com, clicking "Collections" on the Navigation bar, and scrolling until you find the collection entitles "Creating HTML5 apps with Geometry Expressions." All the apps contained in the collection include the GX files so that you can look further at how the apps were made.

INDEX

U

V

W