**Make Your Own Distance Sensor!**

**Created by: Adam Poetzel, July 25, 2013**

**Challenge:**

Create a “distance sensor” that will graph the horizontal distance between two objects over time. Users can move one of the objects to create a “real-time” graphing story of the situation. Make a reset button to start a new graph when desired. (See image below!)

**Things You’ll Need to Know:**

* Graphing on the Coordinate Plane 🡪 (x,y) relationship
* Using “Pen Trails” in Etoys

**Things to Think About:**

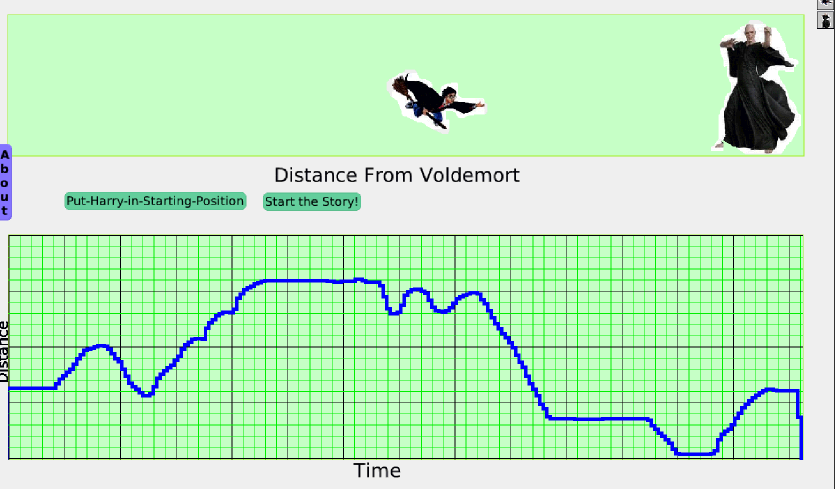
* How do you calculate the horizontal distance between two objects on a coordinate Grid?
* How can I make a “graphing point” that will move according to my two objects’ distance apart over time? What will it’s X and Y coordinates be associated with?
* How can I scale the actual distance apart data so it fits nicely on my graph?

**How to Do It:**

* There are many ways! See the attached lesson plan for one possibility.

**Extensions:**

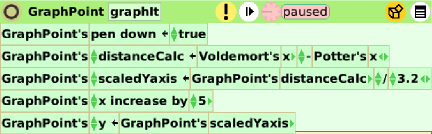
* Create a graph that will differentiate between whether the “moving” object is to the right or left of the “stationary” object. (Ex: 5 feet to the right would be shown differently than 5 feet to the left)
* Create a motion sensor that is not limited to horizontal distance. Have it create a graph that will show the actual distance apart over time between two objects, no matter where they are moved on the screen



**“Make Your Own Distance Sensor” Directions**

(Remember, there are many ways to do this. This is just one method. If you have a different idea you want to try, go for it!)

1. I suggest finding 2 images you want to use for your story. I chose Harry Potter and Voldemort. Save these images as .jpg files and drag them into your Etoys screen. Using the “Lasso” tool in the Supplies Box at the top of your Etoys screen, you can enclose the part of the image you want to use in your story. (You can also skip finding images and just use Etoys paint tools to draw 2 of your own)
2. Use a “PlayField” from the Supplies Box to create a long rectangular area that you want your “story” to take place within. Use the black “Pick-Up” tool in each object’s Halo to place each of your images in the playfield (make sure the black “pick-up” tool is within the playfield when you let go). Now, when you move the objects using the “move” handle in the Halo, they will stay in the playfield.
3. Add a new playfield (of similar length) beneath the first. This will be used for your graph. The height of this playfield will be adjusted based on your needs. Choose the white/black “Menu” Handle in the Halo, Choose “Playfield Options”, and “make Graph Paper”. Set as desired.
4. Paint a small circle, keep it, and call it “GraphPoint”. This will be the point we use to construct the distance-time graph. Move it to the left side of your graph paper.
5. Now we need to a make a script that tells GraphPoint how to move to make a Distance-Time Graph for the objects in your top playfield. The X value of GraphPoint will need to be associated with the time that has passed in the story. The Y value will be associated with the distance between the two objects in your top playfield.
6. Drag out an empty Script of GraphPoint and call it “GraphIt”. Drag “GraphPoint’s X” tiles by the green and white arrow into the new script. Every time Etoys reads this script, we want this x-value to increase to simulate a passing of time. Click on the small black arrow facing left after “GraphPoints X” in your script and choose “increase by”. Then choose a number. This number will control the rate at which time passes. I suggest starting with the number 5.
7. Now we need to associate GraphPoint’s Y-value with the horizontal distance between your two objects. We can calculate this by subtracting the x values of your two objects. I clicked the orange circle with the “v” in the top of GraphPoint’s viewer window to create a new variable called “DistanceCalc”. I dragged this new variable into my GraphIt script and set it to be the difference of my object’s x values (see image a few steps below). Get these tiles from the viewers of each object.
8. We have a small problem. The distance between the objects can be much greater than the actual distance we want to show on our graph. We need to “scale” this distance to fit on our graph. I created a new variable called “ScaledYAxis” and set this value to be the “DistanceCalc” Value divided by 3.2, so it is shrunk about three times smaller for my graph. You can play with this number later to change as needed.
9. Now, drag “GraphPoint’s Y” tiles by the green and white arrow into the script. Set the Value of this to be equal to “ScaledYaxis”
10. Finally, in the “pen use” menu within GraphPoint’s Viewer, add a “pen down” tile to your script and set it to “true”. The image below shows a sample of my finished “GraphIt” Script.



1. At this point, it will be helpful to make a “Restart Button” that lets you start a new graph. Pull out a new script for GraphPoint and Call it “Reset”. In this script, you need to clear all pen trails, and tell GraphPoint to go back to a good starting position. See a sample script below. You will need to adjust your x-value accordingly. Once done, click the black and white menu circle seen in the top right of the image below and choose “button to fire this script”. Place this button on your screen.



1. You can also create a start button to begin the “Graphit” Script. See below for an example. Then click the menu icon and choose “button to fire this script”. Place this button on your screen.



1. Try it out! (and trouble shoot accordingly). Press the “Start Button”, then move one of your objects in the top playfield back and forth from the other. When the graph is done, press the “reset” Button to immediately start a new graph.
2. Adjust as you want! Consider adding labels to your axes!