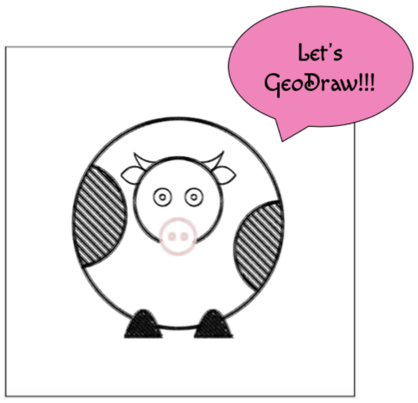
GeoDraw Tutorial

Meredith Wolf and Isabel Grondin



GeoDraw is a language that lets you use basic math equations to construct cartoon drawings. Using GeoDraw, you can draw any equation of the form y = *Operation*. By default your equations will have the edges of the canvas as bounds, but you can also specify your own bounds. Furthermore, you can choose to use one of our handcrafted brushes to add artistic style to your drawing. You can even create brushes of your own!

Let's get coding 😊😊😊!

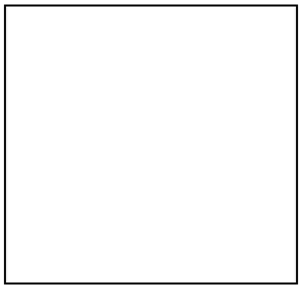
To start off, the first line of every GeoDraw program must be a canvas specificification. A canvas has two required positional arguments, the height and the width, followed by an optional third positional argument, a background color. The height and the width of the Canvas must be positive, and the maximum canvas size that can be specified is 600 by 600. If a background color is not specified, then the default background color of white will be used. Let's look at some examples - Create a file tutorial.geo and add the following line of code:

```
canvas(300, 300)
```

Now you can run this file at the command line using:

```
$./geo tutorial.geo
```

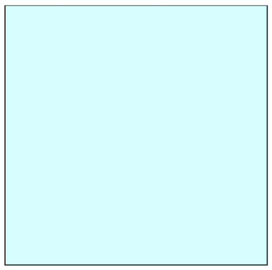
If you have followed the instructions, you should see the message "`Success! tutorial.geo.svg created.`" As you can see, your program will also output an SVG file "`tutorial.geo.svg`". You can open this SVG file in your favorite browser to produce the following result:


* Not shown to scale

If we want to get fancy, we can edit our code to add a background color! Colors follow the RGB format and are passed in using the following format (R,G,B). R, G, and B are integers ranging from 0 to 255. Lets change our code to:

```
canvas(300, 300, (204, 255, 255))
```

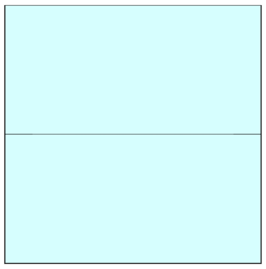
When we run it, we will now get the following:



Now let's draw some lines! The draw function takes four parameters: (1) the equation, (2) a list of bounds, (3) a color for the line, and (4) a brush type. All of these parameters are positional and must always be specified! Let's start by drawing a horizontal line across the middle of the canvas. Add the following line to you GeoDraw file:

```
draw(y = 150, [], (0,0,0), 'simple')
```

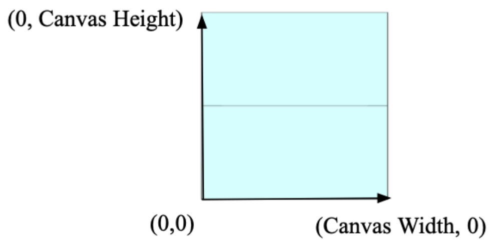
The output will be:



Let’s break this down piece by piece!

The Equation:

You may have noticed that our equation y = 150 puts a line in the middle of the canvas. You can imagine the canvas on an xy coordinate plane, with (0,0) being the bottom left corner. Look at the visual below for reference.



It is important to note that all equations must take on the form Y = OPERATION. A complete list of the operations available to you is given in the table below. Here is a comprehensive list of operations in GeoDraw and their syntax:

Subtraction	(o1 - o2)
Multiplication	(o1 * o2)
Division	(o1 / o2)
Addition	(o1 + o2)
Exponentiation	(o1 ^ o2)
Sin	sin(o1)

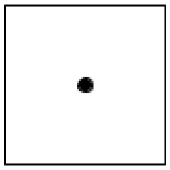
Cos	cos(o1)
Square Root	sqrt(o1)
Absolute Value	abs(o1)

You can replace o1 and o2 with any other operation, x, or a number, which can either be an integer or a float. Note that parentheses are very important in GeoDraw! For example, `sin((x + 4))` is correct syntax, but `sin(x+4)` is not.

Bounds:
The second argument of the draw function is the bounds. Bounds must be provided in a list format and have the following form: x Eq Num or y Eq Num, where Eq can be < or > and Num can be a float or an integer that is within the canvas size.

Color:
Color corresponds to the color of the line and it is given in the same RGB format as it is for the background color of the Canvas.

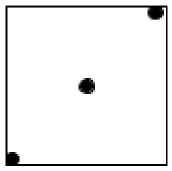
Brush:
Geodraw has five predefined brushes, Funky, Whispy, Simple, Thick and Sparse. A brush simply consists of a list of points within a 5 by 5 frame that is dragged across the canvas in the shape of the given equation. For example the simple brush is defined as follows:



Geodraw also gives you the option to design your own brush! It is important to note that all the predefined brush names are protected and can not be redefined. Let's try to create our own brush! Type the following line of code in your tutorial file between the canvas and draw statements.

```
brush myBrush =  [(0,0) (2.5,2.5) (5.5)]
```

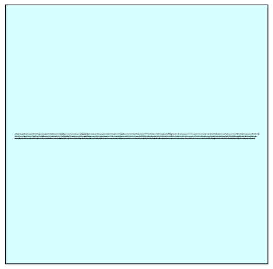
You can imagine our newly defined brush myBrush is now defined as follows:



Now that you know all the features in the draw function, let’s play around a little to see how these features work! Alter the last line of your code to utilize our new brush. And while we are at it, let’s add some bounds to make things interesting!

```
draw(y = 150, [x >10, x < 290], (0,0,0), 'myBrush')
```

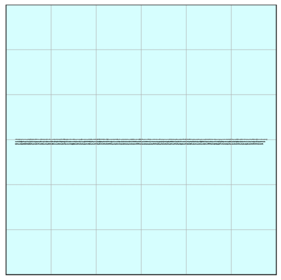
Now the output of the program should look like this:



Gridlines:
GeoDraw also provides the users with one more feature, gridlines! Gridlines can be helpful for fine tuning your equations. Let’s add some gridLines - Add this line of code to the bottom of your program:

```
gridlines(50)
```

The output should be as follows:



As you can see, there are gridlines placed every 50 units on the X and Y axis. Gridlines must be passed a **positive integer**!

If you would like to add comments, you simply just need to add the # sign before your comment. Incase you got lost along the way, here is the full, commented, program to get the output above:

```
# Create a blue canvas
canvas(300, 300, (204, 255, 255))

# Generate a new brush
brush myBrush = [(0,0), (2.5,2.5), (5,5)]

# Draw a black horizontal line in the middle of the canvas
draw(y = 150, [x > 10, x < 290], (0,0,0), 'myBrush')

# Add gridlines for future work :)
gridlines(50)
```

You are now on your way to becoming a professional GeoDraw coder! Happy Drawing!

(p.s. If you want any more examples or inspiration, check out `example-1.geo`, `example-2.geo`, `example-3.geo`, and `cow.geo`!)