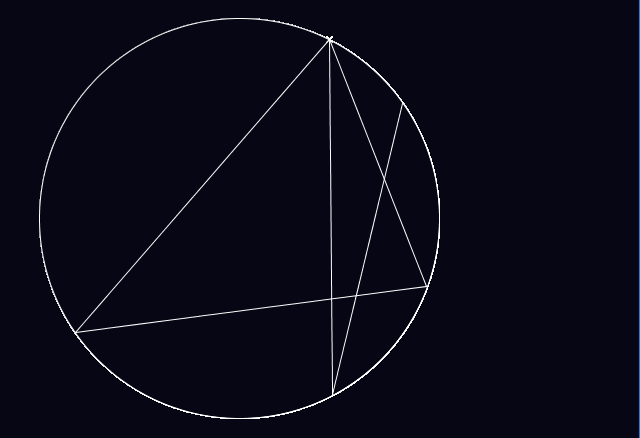
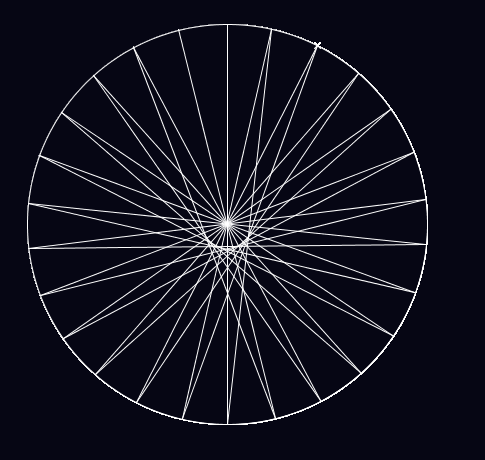


Circle Language



Introduction – Learning Circle Language

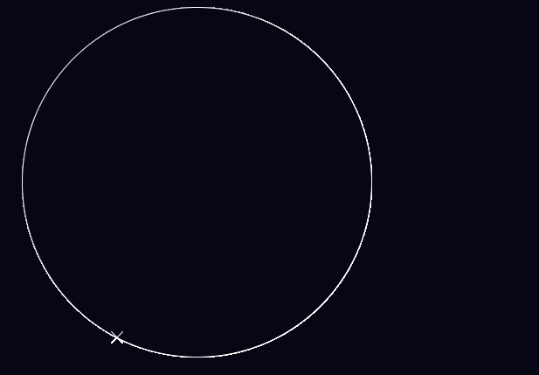
Introduction



Circle Language is based on a very simple principle.

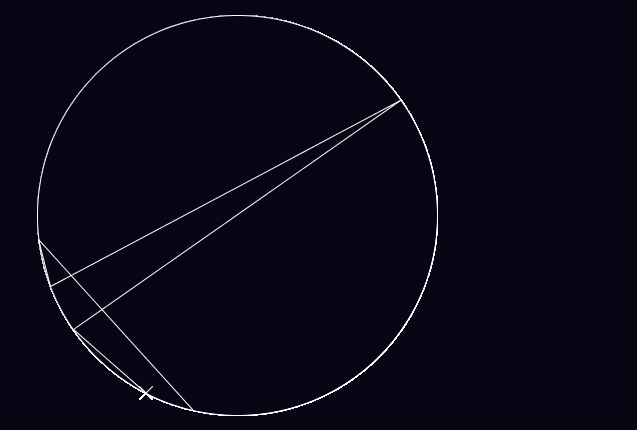
See every spoke on the wheel above? Each one represents a letter, starting with A at the top, and moving around clockwise. So, for instance, P is at a bearing of 221°, and B at a bearing of 014°.

To write a word, you put a cross at the starting letter:



P

Then join all of the letters together in the order they appear in the word:



Presto!

Drawing Circles

Prerequisites:

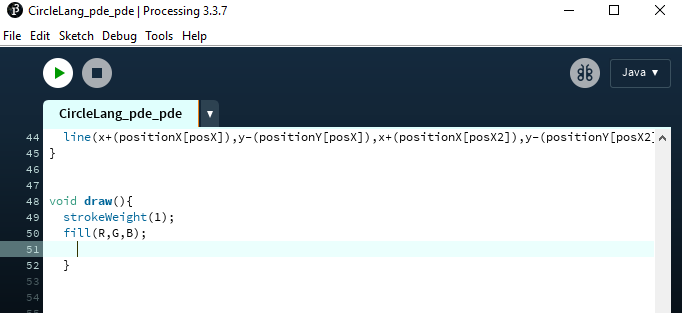
**Processing (latest version is currently 3.3.7, free)**

**YOU DO NOT NEED ANY KNOWLEDGE OF CODING**

OK, so I’m going to anticipate that a lot of the people here don’t really care about the code, so I’m not going to spend any time on it. In any case, the code itself is pretty simple so you could probably figure it out. I coded this after just an *hour* of learning Processing.

Yeah, *an hour*. That’s how intuitive Processing is.

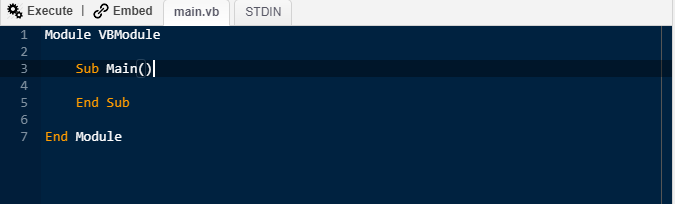
Once you’ve downloaded Processing, open up CircleLang.pde. There should be about 50 lines of poorly written code there.



Scroll down until you reach Line 51 as shown in the image.

Now, Processing code can get awfully repetitive. It’s just lines, lines, lines. So I outsourced, and used Visual Basic to help me write the code.

You can use Visual Studio for the next step, but since you can’t copy and paste out of the console, I think the link is better. A free online version of BASIC can be found [here](https://www.tutorialspoint.com/compile_vb.net_online.php).



The compiler should look something like this. Delete everything there, and copy and paste EVERYTHING in the text box below (expand it to see full contents) into the compiler.

Module VBModule

Sub Main()

Dim str as String ="enter text here"

Dim x as String = "1.2"

Dim y as String = "1.6"

str = str.ToLower()

Console.WriteLine("stroke(R,G,B);")

Console.WriteLine("ellipse(radius\*" & x & ",radius\*" & y & ",radius\*2,radius\*2);")

Dim i as Integer = 0

Console.WriteLine("cross(radius\*" & x & "+(positionX[" & CStr((Asc(str.Substring(i,1))-97)) & "]),radius\*" & y & "-(positionY[" & CStr((Asc(str.Substring(i,1))-97)) & "]));")

Console.WriteLine("stroke(255,255,255);")

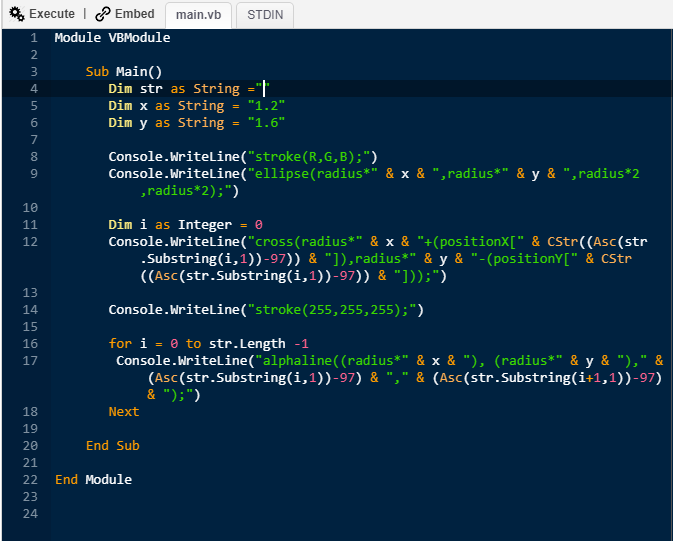
for i = 0 to str.Length -1

Console.WriteLine("alphaline((radius\*" & x & "), (radius\*" & y & ")," & (Asc(str.Substring(i,1))-97) & "," & (Asc(str.Substring(i+1,1))-97) & ");")

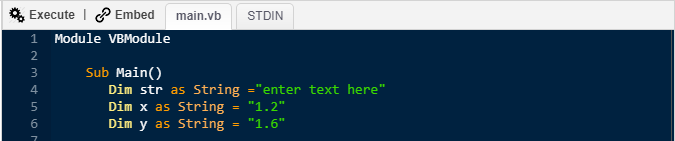
Next

End Sub

End Module

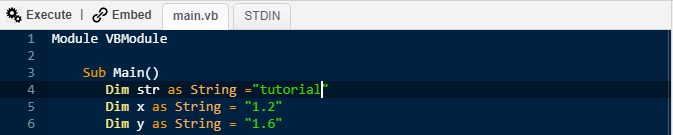


Since this tutorial is as beginner-friendly as possible, I’ve created the program so you can ignore all of it except for this part right at the top:

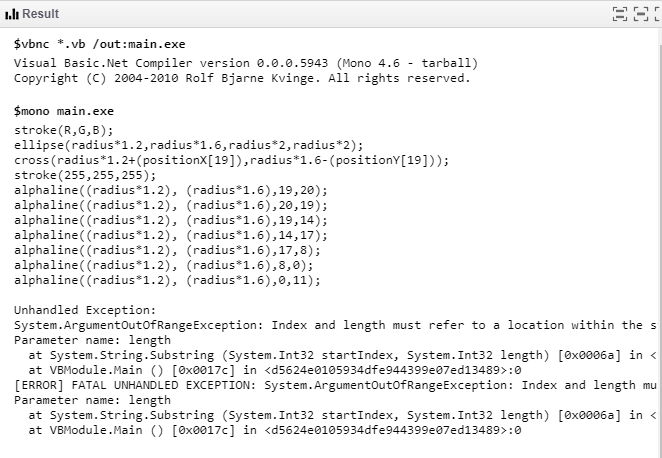


Right where it says “enter text here”, enter your text in between the quotation marks. Do not enter any punctuation or spaces. This only works for **letters.**

The *x* and *y* are important when drawing multiple words, but we’re not quite there yet so I’ll come to that later.

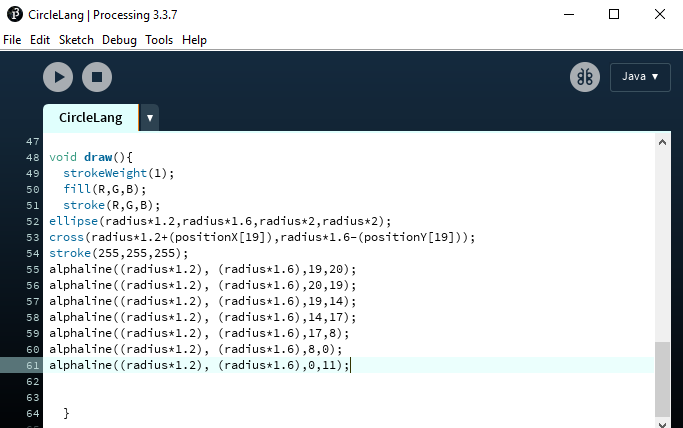


Once you’ve entered your text, click Execute. It may take a few tries for the program to run. It will result in something like this:



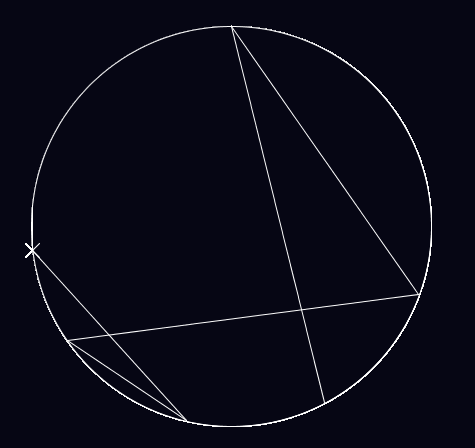
Copy all of the text from the first ‘stroke’ to the last line beginning with ‘alphaline’. Do not copy anything above, or below.

Now paste this text into the Processing file (at line 51, as you may recall):



If you want to see the circle, remove this line! [stroke(R,G,B);]

…click the Run button and there you have it. Your word written in Circle Language.



Size and Colour

The size of the circle can be changed very simply. Just change the value for the radius [line 10]. **By default, it’s set to 50 pixels, but looks much cleaner at larger values.**



If you change the size of the circles, you may also want to change the size of the window [line 17].



You can change the colour of the background by changing the RGB values [line 4-7]



and the colour of the lines by finding the stroke() and changing the RGB values.

I recommend:

**R: 6, G: 6, B: 20 (default)**

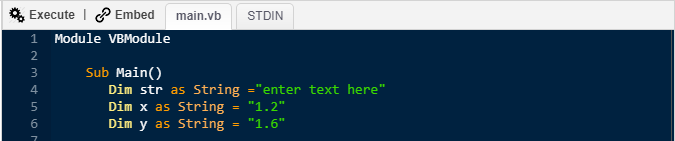
stroke (255,255,255)

**R: 170, G: 0, B: 0**

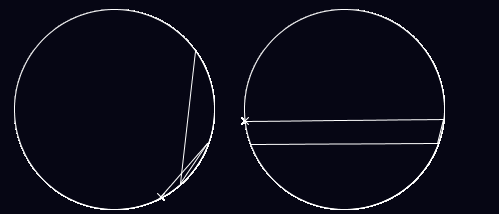
stroke (0,0,0)

But feel free to find your own preferred colour combinations.

Formatting for multiple circles



Now this is where the *x* and *y* come in. To draw multiple circles, you have to manually change the *x* and *y* so that the circles don’t overlap. Increasing the *x* value moves the circle to the right, increasing the *y* value moves it downwards.



X = 1.2, Y= 1.6

X = 3.5, Y= 1.6

It does take a bit of trial and error. Unfortunately, I haven’t found a better way than this that accounts for changing radii – but it works. And I guess it gives you more freedom on how you format your circles.

Writing Guidelines

If you’re going to use Circle Language as a formal language, there are three punctuation marks you should be aware of. This is going to require looking at the code a little more.

* the cross
* the doublestroke
* the chevron (incomplete)

We’ve already covered the cross, so I’ll skip that. The doublestroke is used when marking a double letter, such as the **L** in **HELLO,** or the **S** in **PASS**.

The doublestroke has to be inputted manually, so it’s a bit of a pain.

You need to know the *x* and *y* of the target circle, and the number of your letter, and input them in the designated spaces:

doublestroke(radius\* +(positionX[ ]), radius\* -(positionY[ ]));

number of letter **minus one**

*y* of target circle

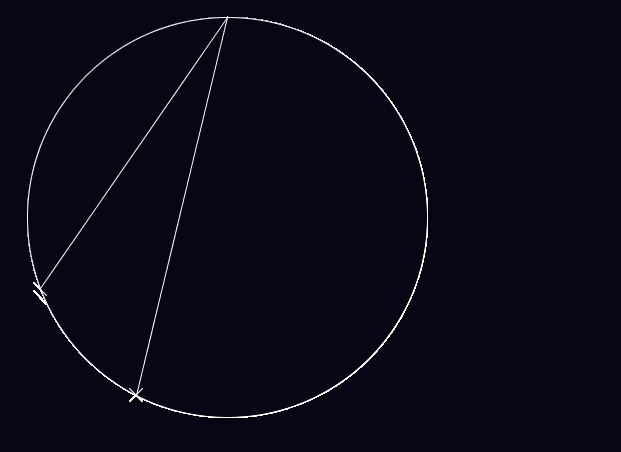
e.g. 1.6

number of letter **minus one**

e.g. C = **2**

*x* of target circle

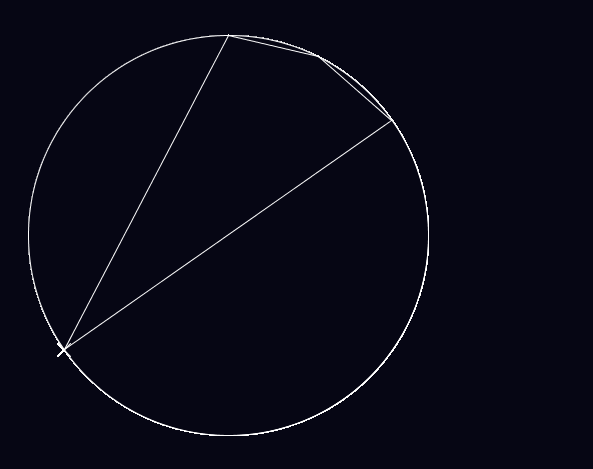
e.g. 1.2



That’s the doublestroke ->

The Chevron

Unfortunately, it’s not possible to determine the direction of a line in Circle Language. A line spelling **o-l**, could just as easily be spelling **l-o**. To reduce errors like these, a chevron is necessary to point the line in the right direction.

The problem is, I’m not good enough to code it.

For instance, this is ‘racer’ but it could be ‘recar’.

Or an Imperial Star Destroyer.

The chevron should look something like what I’ve drawn in Word on the image – a directional arrow.

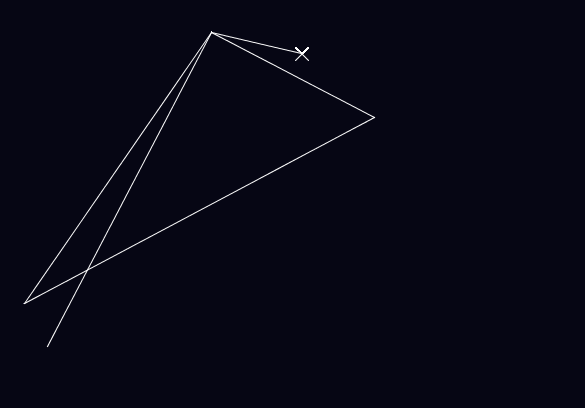
If anyone is good enough to code this, take a look at Chevron\_Draft.pde. I think I’ve made some progress there, but on the other hand, I can hardly read my own code so you might wish to start from scratch.

Oh, and last but not least, **you don’t need to draw the circle stencil with every word**. In fact, it often looks better without. The instruction on how to remove the circle is on **page 6**.

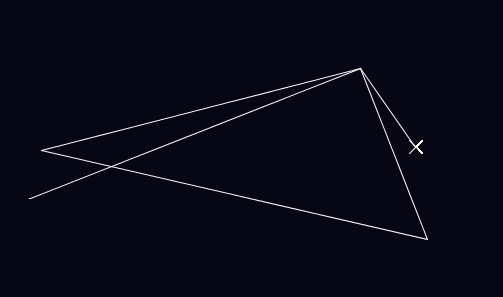
Conclusion

Circle Language is quite unique, I think, if I do say so myself. Although I know it would be diabolical as an actual written language, the patterns it yields are sometimes quite beautiful.

I originally thought of Circle Language as a means of steganography, a secret language for passing notes, and it absolutely could be used for that. I mean, take a simple Caesar shift. It can change a word from this:



To this:



It’s the same shape, just rotated. Maybe Circle Language is the best way to break a Caesar cipher. And even then, bear in mind, that’s just a simple shift. You could jumble the letter order completely and find all sorts of new shapes. Common prefixes and suffixes all take the same route.

Whatever you use Circle Language for – pattern finding, steganography, communication – I hope you enjoy it.

*You can contact me at legionofwhales@gmail.com*

