CCPROG1 MP Specifications – Part 3 of 3 (for S13A + S14A + S15B + S16B)
Circles & Hypotrochoids – Application of cosine() and sine() Library Functions
Part 3 is 25% of the MP Grade

I. INTRODUCTION

Continuing on the concept of divide-and-conquer, and separate compilation, we will apply the **`cosine()`** and **`sine()`** library functions that were created for the last part of the MP. These functions will be called by other programs that will compute and output (x, y) coordinates of parametric equations to produce 2D shapes, specifically, that of a circle, and hypotrochoid.

II. GLE

We will NOT write C codes for plotting or drawing as this would entail learning graphics libraries which is totally outside of the scope of CCPROG1. Instead, we will use the Graphics Layout Engine (GLE) which is an easy-to-learn open software for plotting curves and graphs.

TASK #1: Learn How to Use GLE.
As a preparatory step, please download and install GLE in your own computer. Installers for Windows and Mac platforms are available. The links are:

Homepage: http://glx.sourceforge.net/

Download: http://glx.sourceforge.net/downloads/downloads.html

A reference manual, along with several GLE codes and plots will be included in the installation. You do not need to know everything about GLE – only those that pertain to plotting a series of (x, y) coordinates are the ones that are important.

Please learn how to use GLE (it's easy!) by understanding and editing/experimenting on the sample GLE scripts attached with this MP specs.

II. CIRCLE

/* NOTE: we already did this part in the lab using Excel to plot the circle. Please try to re-do the task on your own, and use GLE for plotting. */

As a practice (that is, this part is not graded yet), Let us deal with a simple curve shape first – a circle. We will write a program that will compute and print the (x, y) coordinates of points that when connected together will form a circular shape. The parametric equations of a circle are:

$$x = r * cosine(theta)$$
$$y = r * sine(theta)$$

where *r* is the radius, *theta* is an angle that ranges from 0 to 360 degrees.

TASK #2: Compute points on a circle.
Use the skeleton file **`CODENAME_circle.c`** and implement the function that will generate the list of (x, y) coordinates. Input the values of radius *r* via **scanf()**. You'll need to use the library functions **`cosine()`** and **`sine()`**.

Let us assume that your codename is SHAZAM. Do a separate compilation of your **`SHAZAM_circle.c`** file. You should know how to do this by now. Write the command below:

```
D:\CCPROG\MP>
```

The compilation step would produce the object file **`SHAZAM_circle.o`** if there is no syntax error.

Next, link your object file with the math object file as follows:

```
D:\CCPROG\MP> gcc -Wall SHAZAM_circle.o SHAZAM_math.o -o SHAZAM_circle.exe
```

If there are no linking errors, you should have the exe file. Run your exe file as follows:

```
D:\CCPROG\MP> SHAZAM_circle
```

Run the exe file again, but this time, use output redirection as follows:

```
D:\CCPROG\MP> SHAZAM_circle > circle.txt
```

A successful run would produce the **`circle.txt`** file which you can view with any text editor like Notepad. Please note that you can use any filename (not just circle.txt) for the text file. Just be careful not to overwrite an important existing text file with the same filename.

For your final step, the contents of **`circle.txt`** will be plotted (visualized) using GLE. An example GLE script is provided to you for this purpose; please see the file **`plot_circle.gle`**. You should see a plot corresponding to, of course, a circle.

Try the following experiments:

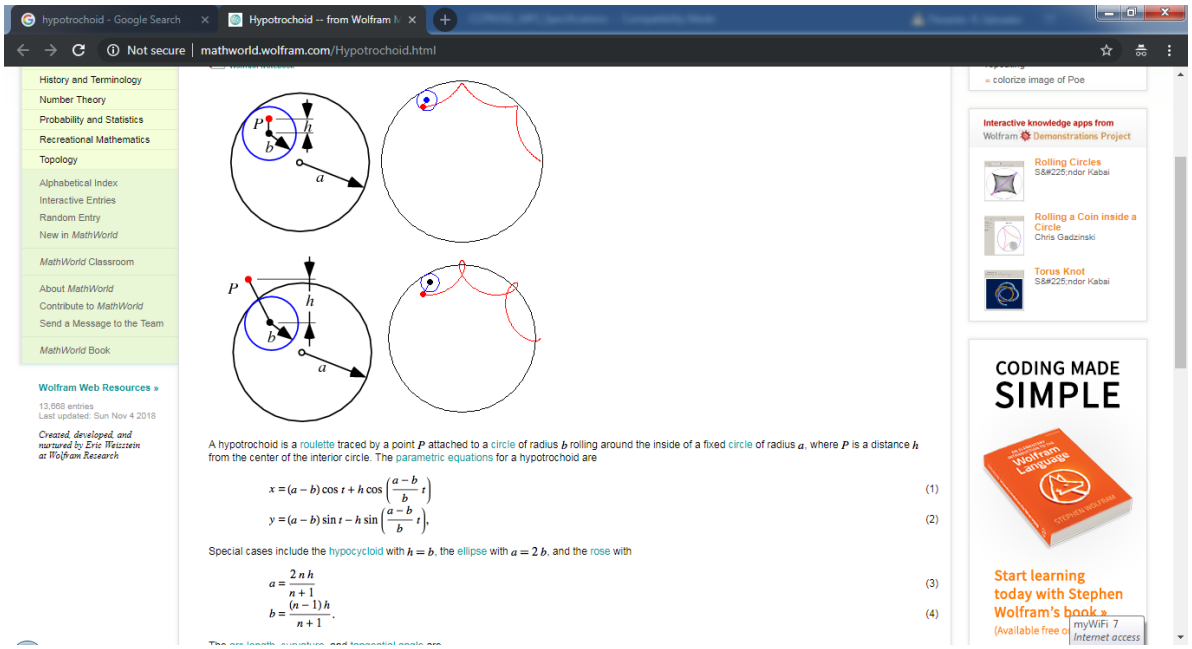
- Change the values of radius *r*. Run the exe file again with input and output redirection. You should get different a different circle with a different radius.
- Please experiment with GLE by editing the **`plot_circle.gle`** script to change the circle to red, blue, green, etc. Use notepad to open and edit the GLE script.

You are now ready to tackle the actual graded part of the MP – producing a Hypotrochoid.

III. HYPOTROCHOIDS (SPIROGRAPH)

Be inspired by first exploring https://nathanfriend.io/inspirograph/ and produce your own 2D spirograph drawings. You probably played with a physical spirograph when you were a kid ☺.

The spirograph is a 2D curve which is known in mathematics as *hypotrochoids*. The parametric equations for computing the (x, y) coordinates of the a hypotrochoid are specified in http://mathworld.wolfram.com/Hypotrochoid.html -- see the screenshot below. You are encouraged to learn more about hypotrochoids by searching and reading other resources that you can find via google and youtube[1].



Only equations (1) and (2) are relevant in the MP, you need not understand the remaining equations. The relevant variables are:

- *x* the *x* coordinate
- *y* the *y* coordinate
- *a* the radius of the big circle
- *b* the radius of the small circle
- *h* determines if a point on the curve will be inside or outside of the big circle
- *t* is an angle in radians (think of *t* as theta).

The variables *a*, *b*, and *h* control the shape of the hypotrochoid. Different values result into different shapes.

IMPORTANT NOTE: the angle *t* is not necessarily limited to just the range 0 to 360 degrees. You will need to generate values of *t* greater than 360 in your MP3 program to get the full complete shape. Please experiment.

TASK #3: Compute and plot a hypotrochoid.
Use the skeleton file **`CODENAME_spirograph.c`**. It only contains comments. You'll need to write all the remaining codes from scratch. Just like in the circle example, you'll need to compile your file separately, and link it with the math object file.

For example:

```
D:\CCPROG\MP> gcc -Wall -c SHAZAM_spirograph.c
D:\CCPROG\MP> gcc -Wall SHAZAM_spirograph.o SHAZAM_math.o -o SHAZAM_spirograph.exe
```

[1] Optional resource for he mathematically curious, please watch https://www.youtube.com/watch?v=n7T91LDJ--E

Your program should ask for three input values corresponding to *a, b, h* via scanf().

It is recommended that you create a text file that contains the values for the inputs. An example input file named as **`SHAZAM_input1.txt`** is provided for you reference. You can then use input and output redirection when your run the program, for example:

```
D:\CCPROG\MP> SHAZAM_spirograph < SHAZAM_input1.txt > SHAZAM_output1.txt
```

The file **`SHAZAM_input1.txt`** contains the input values, while **`SHAZAM_output1.txt`** contains the output x and y coordinates.

Thereafter, use GLE to plot the hypotrochoid. An example script named **`plot_spirograph.gle`** is included for your reference.

Try the following experiments:

- Change the values of ***a, b,*** **and** ***h*** in the **`SHAZAM_input1.txt`** file. Make sure to save the text file. Run the exe file again with input and output redirection. You will produce different hypotrochoids with varying shapes.

- Please experiment with GLE by editing the **`plot_spirograph.gle`** script to change the spirograph color to red, blue, green, etc.

TASK #4. Draw your spirograph masterpiece.
Come up with an interesting and colorful combination of hypotrochoids, i.e., your masterpiece!

This is accomplished by running your **`SHAZAM_spirograph.exe`** file several times with different parameter values and redirect the output, i. e., list of (x, y) coordinates to different text files. This is best done with input and output redirections as described above. For naming convention use your codename for the input and output text files as shown in the example calls below:

```
D:\CCPROG\MP> SHAZAM_spirograph < SHAZAM_input1.txt > SHAZAM_output1.txt
D:\CCPROG\MP> SHAZAM_spirograph < SHAZAM_input2.txt > SHAZAM_output2.txt
D:\CCPROG\MP> SHAZAM_spirograph < SHAZAM_input3.txt > SHAZAM_output3.txt
D:\CCPROG\MP> SHAZAM_spirograph < SHAZAM_input4.txt > SHAZAM_output4.txt
D:\CCPROG\MP> SHAZAM_spirograph < SHAZAM_input5.txt > SHAZAM_output5.txt
```

You will need to edit the GLE script to plot your multiple hypotrochoids. You should have at least 5 hypotrochoids of different shapes and colors.

TASK #5. Submit Files via Canvas.
Submit a ZIPped file named **`CODENAME.zip`** containing the following files:

- **`CODENAME_spirograph.c`** (C source code)
- Your 5 or more input files named as: **`CODENAME_input1.txt`** to **`CODENAME_input5.txt`**
- your 5 or more output files named: **`CODENAME_output1.txt`** to **`CODENAME_output5.txt`**
- **`CODENAME.GLE`** (your modified GLE script source code)
- **`CODENAME.PDF`** which shows your spirograph masterpiece.

Take note of the Canvas deadline – December 3, 2018 at 11:59PM.

TASK #6. Submit your printed deliverable.
Submit the following printed document within the first 10 minutes of our class on December 4 (for S15B and S16B), and December 5 (for S13A and S14A):

- **accomplished mp_checklist.PDF**

IV. TESTING, CHECKING and ASSESSMENT SCHEME

Your source code will be tested in class on the day of the printed deliverables submission. **BLACK-BOX TESTING will be used to check your submission.**

Please refer to the rubric in your syllabus – it specifies the basis for the MP assessment. Note that by default you will receive a perfect grade of 100%. It will remain 100% if there are no programming errors and that you complied properly with all the specifications and that you are able to properly articulate your solution.

Deductions will be applied based on the following:

- **Violation of the Honor Code will result in a FINAL GRADE of 0.0 for the course (not just for this MP).**
- Syntax Error/s: a solution that has a syntax error will automatically receive a grade of 0.0.
- Logical Error/s: deductions for logical errors will depend on the severity of the error.
- Non-compliance with specification: will result to a 1% minor deduction **for each** infraction; for example: incorrect filename, incorrect printout, did not include the programmer's name in the source code as part of the internal documentation.
- Bad programming style – will result to a 10% maximum deduction for source codes that DO NOT follow recommended "good" programming practices thus resulting to difficult-to-read codes due for example to incomprehensible variable name, bad indentation, etc.

--- End ---

Last Note: Consult me immediately if you have any clarification, question or problem regarding this academic activity. Enjoy! ☺☺☺