

CS 218 – Assignment #10

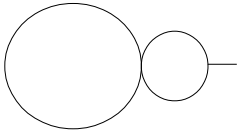
Purpose: Become more familiar with data representation, program control instructions, function handling, stacks, floating point operations, and operating system interaction.

Due: Tuesday (3/07)

Points: 150

Assignment:

Write a simple assembly language program to plot a Spirograph¹ drawing. Imagine the movement of a small circle that rolls on the outside of a rigid circle. Imagine now that the small circle has an arm, rigidly attached, with a plotting pen fixed at some point. That is a epicycloid², often referred to as Spirographs. Use the provided main program which includes the appropriate OpenGL initializations. You will need to add your code (two functions) in the provided functions template.



The program should read radius 1 (fixed circle), radius 2 (moving circle), offset position (rigid arm length), rotation speed, and color from the command line. For example:

```
./spiro -r1 80 -r2 10 -op 80 -sp 2 -cl y
```

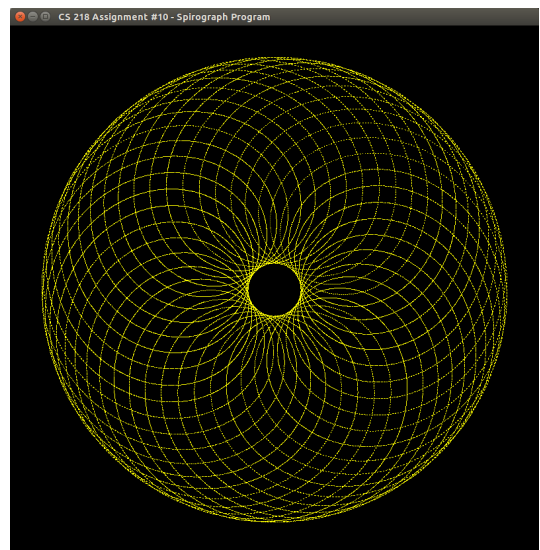
The required format for the date options is: "-r1 <tridecimal number>", "-r2 <tridecimal number>", "-op <tridecimal number>", "-sp <tridecimal number>", and "-cl <color letter>" with no spaces in each specifier. The program must verify the format, read the arguments, and ensure the arguments are valid. The program must and ensure that the **r1**, **r2**, **op**, and **sp** values are between the specified ranges (provided constants). If there are any command line errors, the program should display an appropriate error message and terminate. Refer to the sample executions for examples.



Submission:

Only the functions file will be submitted. As such, the main file can not be altered in any way. When complete, submit only the functions source file:

- A copy of the functions **source file** via the class web page (assignment submission link) by 23:55 PM. **Assignments received after the due date/time will not be accepted.**



1 For more information, refer to: <http://en.wikipedia.org/wiki/Spirograph>

2 For more information, refer to: <https://en.wikipedia.org/wiki/Epicycloid>

Functions

The provided main program calls the following routines:

- Function **getRadii()** to read and check the command line arguments (**r1**, **r2**, **op**, **sp**, and **cl**). The function must read each argument, convert each ASCII/Tridecimal string to integer, and verify the ranges. The range for radius 1 (**r1**) is 0 and 163₁₃ (inclusive). The range for radius 2 (**r2**) and offset position (**op**) is 1 to 163₁₃ (inclusive). The range for speed (**sp**) is 1 to 79₁₃ (inclusive). The allowed colors values (**cl**) are "r" red, "g" green, "b" blue, "p" purple, "y" yellow, or "w" white), lower case only. If all arguments are correct, return the values (via reference) and return to the main with a value of TRUE. If there are any errors, display the appropriate error message and return to the main with a value of FALSE.
- Function **drawSpiro()** to plot the Spirograph functions noted below. The functions should be iterated and will generate a series of (x,y) values which must be plotted.

$$x = \left[(r_1 + r_2) \times \cos(t) \right] + \left[offPos \times \cos \left((r_1 + r_2) \times \frac{(t+s)}{r_2} \right) \right]$$
$$y = \left[(r_1 + r_2) \times \sin(t) \right] + \left[offPos \times \sin \left((r_1 + r_2) \times \frac{(t+s)}{r_2} \right) \right]$$

The **t** value should range between 0.0 and 360.0 and be incremented by **tStep** (predefined) each iteration. As such, it will require 360.0 / **tStep** iterations. Before leaving the function, the **s** value should be incremented by **sStep** which must be set as follows;

$$sStep = \frac{speed}{scale}$$

The function is called repeatedly which generates the animation (based on the changing **s** value between successive calls). The template already includes some of the applicable OpenGL initialization calls (which must not be removed).

OpenGL Installation

For this assignment, we will be using the OpenGL (graphics library) to provide some basic windowing capabilities. As such, the OpenGL development libraries must be installed. This can be done via the command line with the following commands.

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install binutils-gold
sudo apt-get install libgl1-mesa-dev
sudo apt-get install freeglut3 freeglut3-dev
```

It will take a few minutes to install each. You must be connected to the Internet during the installation. After the installation, a re-install of Virtual Box Guest Additions may be required.

OpenGL Calls

The basic OpenGL library calls are included in the provided main and functions template. In order to set the draw color and plot the point, the following OpenGL functions will be required.

```
call glColor3ub(red, green, blue);
call glVertex2f(x, y);
```

The **red**, **blue**, **green** variables are unsigned bytes. The **x** and **y** variables are double floating point values. These calls follow the standard calling convention.

Assembly:

The program must be linked with the OpenGL libraries. An assembly and link script file (asm10) is provided and must be used. *Note*, the script file will require execute privilege (i.e., **chmod +x asm10**). To assemble/link script can be executed as follows:

```
ed-vm% ./asm10 spiro a10funcs
```

Assuming the main file is named **spiro.cpp** and the functions file is named **a10funcs.asm**. *Note*, the **ed@vm%** is the prompt on my machine.

Debugging -> Command Line Arguments

When debugging a program that uses command line arguments, the command line arguments must be entered after the debugger has been started. The debugger is started normally (ddd <program>) and once the debugger comes up, the initial breakpoint can be set. Then, when you are ready to run the program, enter the command line arguments. This can be done either from the menu (Program -> Run) or on the GDB Console Window (at bottom) by typing **run <commandLineArguments>** at the (gdb) prompt (bottom window).

Example Executions (with errors):

Below are some example executions with errors in the command line. The program should provide an appropriate error message (as shown) and terminate. *Note*, the **ed@vm%** is the prompt.

```
ed-vm% ./spiro
Usage: ./spiro -r1 <triDecimal number> -r2 <triDecimal number> -op <triDecimal
number> -sp <triDecimal number> -cl <b/g/r/y/p/w>
ed-vm%
ed-vm% ./spiro -r1 80
Error, invalid or incomplete command line arguments.
ed-vm%
ed-vm% ./spiro -r1 15d -r2 7 -op 85 -sp 5 -cl p
Error, radius 1 value must be between 0 and 163(13).
ed-vm%
ed-vm% ./spiro -r1 150 -r2 7 -op 164 -sp 5 -cl p
Error, offset position value must be between 1 and 163(13).
ed-vm%
ed-vm% ./spiro -r1 15a -r2 7 -op 85 -sp 5 -cl blue
Error, color value must be b, g, r, p, or w.
ed-vm%
ed-vm% ./spiro -r 15a -r2 7 -op 85 -sp 5 -cl b
Error, radius 1 specifier incorrect.
ed-vm%
```

OpenGL Errors

Based on the specific hardware configuration and/or virtual box configuration, the following warning message may be displayed.

OpenGL Warning: Failed to connect to host. Make sure 3D acceleration is enabled for this VM.

This warning message can be ignored. *Note*, some hardware configurations using virtual box may not be able to use OpenGL. An *openGLtest* program is provided to verify the OpenGL installation.

Testing

A script file to execute the program on a series of predefined inputs will be provided. The test script executes the program and performs a series of error tests (with expected output). Refer to the examples for output formatting and error handling. The test script, named **a10tst**, can be executed as follows:

```
ed-vm% ./a10tst spiro
```

The expected error will be shown and then the program is executed (with the specified error).

Example Execution:

Below is an example execution showing the resulting image.

```
ed-vm% ./spiro -r1 c0 -r2 31 -op c0 -sp 2 -cl y      (right image)
ed-vm% ./spiro -r1 80 -r2 10 -op 90 -sp 3 -cl p      (left image)
```

When functioning, the image will rotate based on the speed value (which will vary between different machines).

