CS 218 – Assignment #10

Purpose: Become more familiar with data representation, program control instructions, function handling, stacks, floating point operations, and operating system interaction.
Points: 200

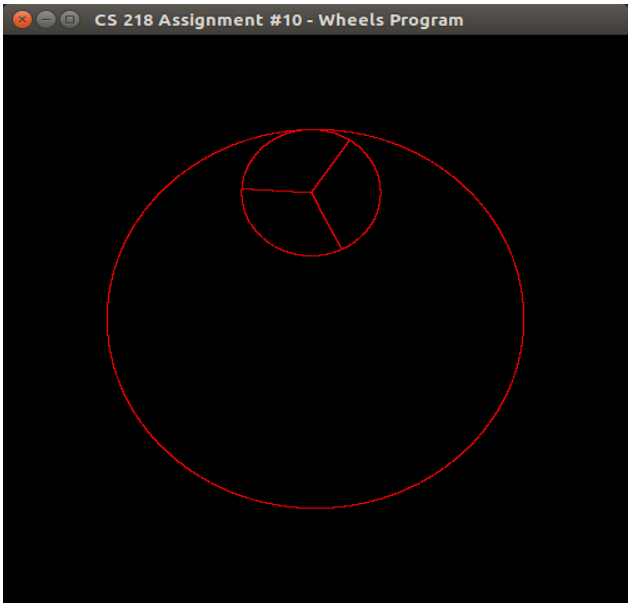
Assignment:
Write a simple assembly language program using OpenGL to calculate and display the output from a series of provided functions. Use the provided main program which includes the appropriate OpenGL initializations. You will need to add your code (two functions) in the provided functions template.

The program should read the draw speed, draw color, and picture size from the command line. The required format for the options is as follows: "-sp <septNumber> -cl <septNumber> -sz <septNumber>" with a valid septenary number for each argument. For example:

```
ed-vm% ./wheels -sp 2 -cl 262022046 -sz 2313
```

Note, the **`ed@vm%`** is the prompt on my machine.

The program must verify the format, read the arguments, and ensure the arguments are valid. The program must ensure that the speed, color, and picture size within the provided ranges. If there are any input errors, the program should display an appropriate error message and terminate. Refer to the sample executions for examples.



Submission:
All functions ***must*** follow the standard calling convention as discussed in class. The functions for the command line arguments and drawing function must be in a separate assembly source file from the provided main program. The provided main program should be linked with the functions. Only the functions file will be submitted. As such, the main file can not be altered in any way.

Functions

The provided main program calls the following routines:

- Function ***getParams()*** to process and check the command line arguments for the picture width and picture height. The function should read each value, convert the ASCII/septenary to integer and verify the range. If all is correct, return the values (via reference) and return to the main with a return value of TRUE. If there are any errors, display the appropriate error message and return to the main with a return value of FALSE.
- Function ***drawWheels()*** to plot the below functions. The formulas should be iterated and each will generate an (***x***,***y***) value pair each must be plotted with the openGL call. The ***t*** value should range between 0.0 and 2π increment by ***tStep*** (predefined) each iteration.

$$x1 = \cos(t)$$
$$y1 = \sin(t)$$

$$x2 = \frac{\cos(t)}{3} + \frac{2\cos(2\pi s)}{3} \qquad x3 = \frac{2\cos(2\pi s)}{3} + \frac{t\cos(4\pi s)}{6\pi}$$
$$y2 = \frac{\sin(t)}{3} + \frac{2\sin(2\pi s)}{3} \qquad y3 = \frac{2\sin(2\pi s)}{3} - \frac{t\sin(4\pi s)}{6\pi}$$

$$x4 = \frac{2\cos(2\pi s)}{3} + \frac{t\cos\left(4\pi s + \frac{2\pi}{3}\right)}{6\pi} \qquad x5 = \frac{2\cos(2\pi s)}{3} + \frac{t\cos\left(4\pi s - \frac{2\pi}{3}\right)}{6\pi}$$
$$y4 = \frac{2\sin(2\pi s)}{3} - \frac{t\sin\left(4\pi s + \frac{2\pi}{3}\right)}{6\pi} \qquad y5 = \frac{2\sin(2\pi s)}{3} - \frac{t\sin\left(4\pi s - \frac{2\pi}{3}\right)}{6\pi}$$

Before leaving the function, the ***s*** value should be incremented by ***sStep*** which must be set as follows;

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

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. Thius 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);    // set color
call glVertex2d(x, y);                // plot point (float)
```

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.

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 (Propgram -> 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% ./wheels
Usage: ./wheels -sp <septNumber> -cl <septNumber> -sz <septNumber>
ed-vm%
ed-vm% ./wheels -sp 3
Error, invalid or incomplete command line argument.
ed-vm%
ed-vm% ./wheels -sp 2 -cl 262022046 -sz 1313 extra
Error, invalid or incomplete command line argument.
ed-vm%
ed-vm% ./wheels -s 2 -cl 262022046 -sz 1313
Error, speed specifier incorrect.
ed-vm%
ed-vm% ./wheels -sp 200 -cl 262022046 -sz 1313
Error, speed value must be between 1 and 101(7).
ed-vm%
ed-vm% ./wheels -sp 2 cl 262022046 -sz 1313
Error, color specifier incorrect.
ed-vm%
ed-vm% ./wheels -sp 2 -cl 2620220547 -sz 2313
Error, color value must be between 0 and 262414110(7).
ed-vm%
ed-vm% ./wheels -sp 2 -cl 262022046 -szz 1313
Error, size specifier incorrect.
ed-vm%
ed-vm% ./wheels -sp 2 -cl 262022047 -sz 3313
Error, color value must be between 0 and 262414110(7).
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 correction functional of the openGL installation.

Assemble and Linking Instructions

You will be provided a main function (**`wheels.cpp`**) that calls the functions. Your functions should be in a separate file (**`a10procs.asm`**). The files will be assembled individually and linked together.

When assembling, and linking the files for assignment #10, use the provided **makefile** to assemble, and link. *Note*, **only** the functions file, **`a10procs.asm`**, will be submitted. The submitted functions file will be assembled and linked with the provided main. As such, do not alter the provided main.

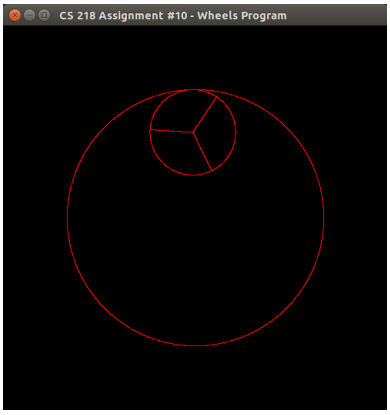
Example Execution:

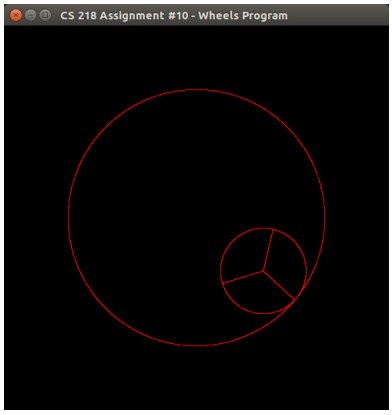
Below is an example execution showing the resulting image.

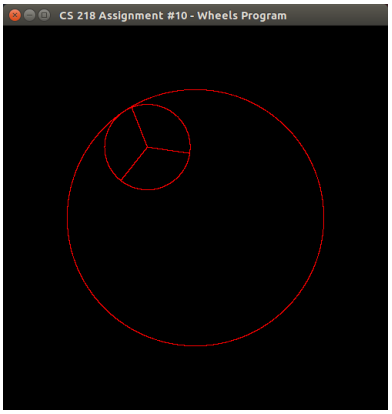
```
ed-vm% ./wheels -sp 2 -cl 262022046 -sz 1313
```

The appropriate speed value will vary on different machines.

When functioning, the inner circle will rotate inside outer circle.







Submission:

- All source files must assemble and execute on Ubuntu with **`yasm`**.
- Submit source files
  - Submit a copy of the program source file via the on-line submission.
  - Only the functions file (**`ast8procs.asm`**) will be submitted.
- Once you submit, the system will score the project and provide feedback.
  - If you do not get full score, you can (and should) correct and resubmit.
  - You can re-submit an unlimited number of times before the due date/time (at a maximum rate of 5 submissions per hour).
- Late submissions will be accepted for a period of 24 hours after the due date/time for any given assignment. Late submissions will be subject to a ~2% reduction in points per an hour late. If you submit 1 minute - 1 hour late -2%, 1-2 hours late -4%, … , 23-24 hours late -50%. This means after 24 hours late submissions will receive an automatic 0.

Program Header Block

All source files must include your name, section number, assignment, NSHE number, and program description. The required format is as follows:

```
;  Name: <your name>
;  NSHE ID: <your id>
;  Section: <section>
;  Assignment: <assignment number>
;  Description: <short description of program goes here>
```

Failure to include your name in this format will result in a loss of up to 3%.

Scoring Rubric

Scoring will include functionality, code quality, and documentation. Below is a summary of the scoring rubric for this assignment.

Criteria	Weight	Summary
Assemble	-	Failure to assemble will result in a score of 0.
Program Header	3%	Must include header block in the required format (see above).
General Comments	7%	Must include an appropriate level of program documentation.
Program Functionality (and on-time)	90%	Program must meet the functional requirements as outlined in the assignment (based on the following breakdown): • Command Line (45%) • Execution with output (5%) • Fully correct output (40%) Must be submitted on time for full score.