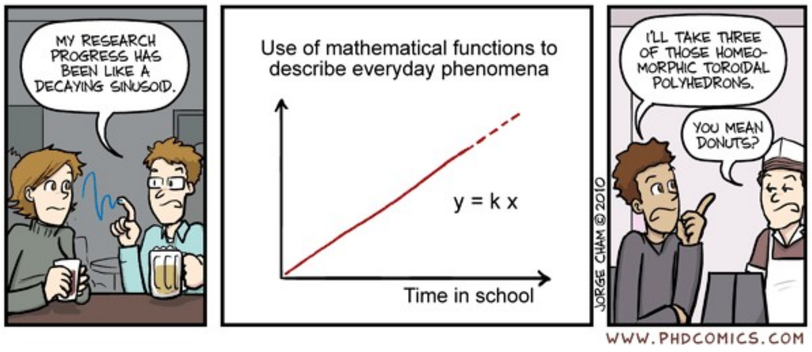
CS 218 – Assignment #10

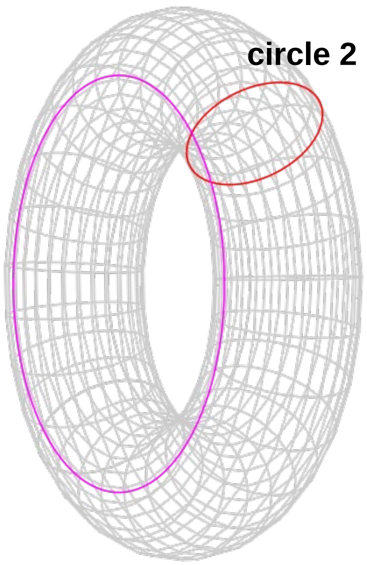
Purpose: Become more familiar with data representation, program control instructions, function handling, stacks, floating point operations, and operating system interaction.
Points: 150

Assignment:

In geometry, a torus is a surface of revolution generated by revolving a circle in three-dimensional space about an axis coplanar with the circle. Real-world examples of toroidal objects include inner tubes, rings, donuts, and bagels. Write an assembly language program to plot a three dimensional Torus[1].


MY RESEARCH PROGRESS HAS BEEN LIKE A DECAYING SINUSOID.
Use of mathematical functions to describe everyday phenomena
y = k x
Time in school
I'LL TAKE THREE OF THOSE HOMEOMORPHIC TOROIDAL POLYHEDRONS.
YOU MEAN DONUTS?
JORGE CHAM © 2010
WWW.PHDCOMICS.COM

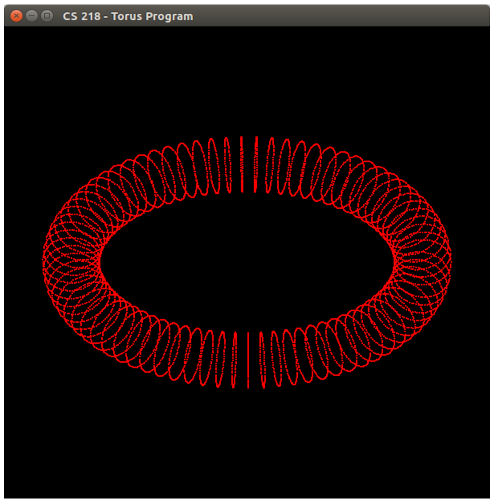
Use the provided main program which includes the appropriate OpenGL initializations. You will need to add your code (three functions) in the provided functions template.



The program should read radius (radius 1) of the major circle (circle 1), radius (radius 2) of the minor circle (circle2), window height, window width, and color from the command line. For example:

```
./torus -r1 40 -r2 7 -h 800 -w 800 -cl 21664283
```

The required format for the date options is: "-r1 <nonaryNumber>", "-r2 <nonaryNumber>", "-h <nonaryNumber>", "-w <nonaryNumber>", and "-cl <nonaryNumber> with no spaces in each specifier. *Note*, a plus sign ("+") for the numeric values is optional (unlike the previous assignment). The program must verify the command line format, read the arguments, and ensure the arguments are valid. The program must also ensure that the ***r1***, ***r2***, ***height***, ***width*** and ***color*** 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 of the error handling.



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 procedures. Only the *functions* file should be submitted. As such, the provided main file can not be altered in any way.

1 For more information, refer to: https://en.wikipedia.org/wiki/Torus

Functions

The provided main program calls the following routines:

- Boolean function ***getParams()*** to read and check the command line arguments (***r1***, ***r2***, ***height***, ***width***, and ***color***). The function must read each argument, convert each ASCII/nonary string to integer, and verify the ranges. The range for radius of radius 1 (***r1***) and radius 2 (***r2***) is 1 and 121_9 (inclusive). The range for ***height*** and ***width*** is 121_9 to 1331_9 (inclusive). The range for the color value (***color***) is 55_9 to 34511010_9 (inclusive). 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 (provided) and return to the main with a value of FALSE.

- Integer function ***cvtnonary2int(strAddr)*** to convert the passed nonary string into an integer. If the conversion is successful, the function should return the number. Otherwise, the function should return -1. The code from a previous assignment should be converted into a function and updated to ensure that a plus sign ("+") is optional since all values will be unsigned.

- Void function ***drawTorus()*** to plot the torus functions noted below. The functions should be iterated and will generate a series of (***x***,***y***,***z***) values which must be plotted.

$$for\left(double\; t=0.0;\; t<(2.0*pi);\; t+=tStep\right)$$

$$for\left(double\; u=0.0;\; u<(2.0*pi);\; u+=tStep\right)$$

$$x = 70.0 \times \left[\cos(t) \times \left(\frac{radius\,1}{10} + \left(\frac{radius\,2}{10} \times \cos(u)\right)\right)\right]$$

$$y = 70.0 \times \left[\sin(t) \times \left(\frac{radius\,1}{10} + \left(\frac{radius\,2}{10} \times \cos(u)\right)\right)\right]$$

$$z = 70.0 \times \left[\frac{radius\,2}{10} \times \sin(u)\right]$$

The ***t*** and ***u*** values should range between 0.0 and (**2.0 × π**) and be incremented by ***tStep*** (predefined) each iteration. As such each loop will require (**2.0 × π**) / ***tStep*** iterations.

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.

RGB Color

RGB stands for Red, Green, Blue. It is a color model used in digital imaging and computer graphics to represent colors. In the RGB model, colors are created by mixing different intensities of red, green, and blue light. Each color is represented by a value between 0 and 255, where 0 represents the absence of that color and 255 represents the maximum amount of that color. By combining different values of red, green, and blue, it is possible to create a wide range of colors. For example, red is represented by [255,0,0], green is represented by [0,255,0], yellow is represented by [255,255,0], purple is represented by [255,0,255], blue is represented by [0,0,255], and white is represented by [255,255,255].

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  glVertex3f(x, y, z);                   // plot point (float)
```

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

Assemble and Linking Instructions

You will be provided a main function (`torus.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.

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).

openGL Warning (possible)

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.

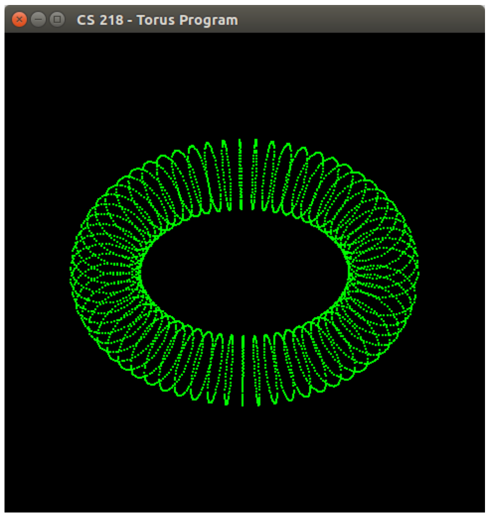
<u>Example Executions (with errors):</u>
Below are some example executions with errors in the command line. The program should provide an appropriate error message (as shown) and terminate. *Note*, the `ed@vm%` is the prompt.

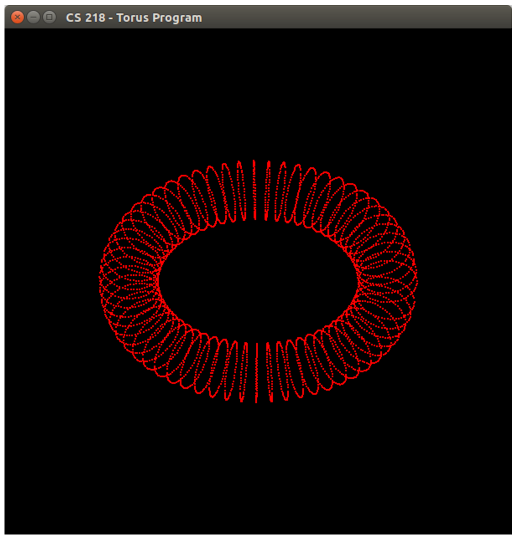
```
ed-vm% ./torus
Usage:  ./torus -r1 <nonaryNumber> -r2 <nonaryNumber> -h <nonaryNumber> -w
<nonaryNumber> -cl <nonaryNumber>
ed-vm%
ed-vm% ./torus -r1 80
Error, invalid or incomplete command line arguments.
ed-vm%
ed-vm% ./torus -r1 181 -r2 5 -h 600 -w 600 -cl 34401103
Error, radius 1 value must be between 1(9) and 121(9).
ed-vm%
ed-vm% ./torus -r1 18 -r2 500 -h 600 -w 600 -cl 34401103
Error, radius 2 value must be between 1(9) and 121(9).
ed-vm%
ed-vm% ./torus -r1 18 -r2 5 -h 6600 -w 600 -cl 34401103
Error, height value must be between 121(9) and 1331(9).
ed-vm%
ed-vm% ./torus -r1 18 -r2 5 -h 600 -w 600 -cl 134401103
Error, color value must be between 55(9) and 34511010(9).
ed-vm%
ed-vm% ./torus -r3 18 -r2 5 -h 600 -w 600 -cl 34401103
Error, radius 1 specifier incorrect.
ed-vm%
```

<u>Example Execution:</u>
Below is an example execution showing the resulting image.

```
ed-vm% ./torus -r1 22 -r2 5 -h 600 -w 600 -cl 108483         (left image)
ed-vm% ./torus -r1 18 -r2 5 -h 600 -w 600 -cl 34401103       (right image)
```

When functioning, the image can be rotated by typing the up, down, right, or left arrow keys. The image point plotting density can be increased by typing a '***i***' or decreased by typing a '***d***'. The window must be selected.





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 (**`a10procs.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.

Note, the final output (40%) will be manually scored. As such, the initail codeGrade score will be incomplete and potentially misleading.