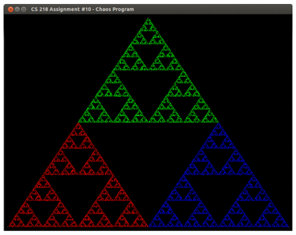
CS 218
Homework, Assignment #10 - Chaos

Purpose: Become more familiar with data representation issues, program control instructions, function handling, and operating system interaction.
Due: Friday (6/23)
Points: 200 (grading will include functionality, documentation, and coding style)

Assignment:

Write a simple assembly language program to plot a series of points on the screen using the provided algorithm. The number of points to plot must be read from command line (as ternary). For example:



```
ed-vm% ./chaos -it 122212121111 -rs 121
```

The required format for the command line is:

./chaos -it <ternaryNumber> -rs <ternaryNumber>

The program must read the iterations specifier ('-it') and the rotation speed (-rs) as ternary numbers, and ensure the numbers are valid. Additionally, the program must ensure that thes numbers with the specified range (provided defined constants), inclusive. The iterations is unsigned. If there are any errors, the program should display an appropriate error message (pre-defined) and terminate.

The provided C++ main program calls routines to check and read the command line arguments. The openGL system will call a user-written routine, *drawChaos()*, to plot the points. All functions must follow the standard calling convention as discussed in class.

Chaos Point Plotting Algorithm

Implement the following algorithm to generate the initial three points (***x***,***y***), where ***i***=0 to 2, positions for plotting:

$$initX[i] = \sin\left((rSpeed + (i * dStep)) * \frac{pi}{180} \right) * scale$$

$$initY[i] = \cos\left((rSpeed + (i * dStep)) * \frac{pi}{180} \right) * scale$$

Implement the following algorithm to generate (x,y) positions for plotting:

```
seed = 987123
for i = 1 to iterations do
      s = rand(seed)
      seed = s
      n = s mod 3
      x = x + ( (init_x(n) - x) / 2 )
      y = y + ( (init_y(n) - y) / 2 )
      set color (r,g,b)
      plot (x, y)
end_for
```

Note, for the color 0=>red (255,0,0), 1=>green (0,255,0), and 2=>blue (0,0,255).

Submission:

When complete, submit:

- A copy of the source file via the class web page (assignment submission link) by 11:55 PM. ***Assignments received after the allotted time will not be accepted!***

Scoring

Scoring will include functionality, documentation, and coding style.

- ~30 pts -> Documentation and coding style
- ~75 pts -> Command line routine (with complete error checking)
- ~95 points -> Display Chaos points (including random number generation)

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. *Note*, after the installation, a re-install of Virtual Box Guest Additions may be required.

Independent Assembly

Your functions must be placed into a *separate file* and linked with the provided main. Only the functions file, not the provided main, will be submitted on-line. As such, you must not change the provided main! A template of the functions file will be provided that includes the string definitions. The program must be linked with the OpenGL libraries. A compile, 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 chaos a10prs
```

Assuming the main file is named **`chaos.cpp`** and the functions file is named **`a10prs.asm`**.

Rotation Speed

Before the plotting is performed, the rStep value should be set as follows;

$$rStep = \frac{rorateSpeed}{rScale}$$

Before leaving the function, the ***rSpeed*** value should be incremented by ***rStep***.

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

Random Number Generation

To generate a pseudo random number, use the *linear congruential generator* method. It is fast, simple, and (if instantiated with the right constants) gives reasonable pseudo-random numbers. The next random number is generated from the previous one by:

$$R_{n+1} = (A \times R_n + B) \bmod 2^{16}$$

The initial random number R_n (on which the rest are based on is referred to as the “seed”). The value for A must be a prime number. For our purposes, set A=9421, B=1, and SEED to 987123. *Note*, to provide a random number between 0 and 2, the plot points algorithm uses the “mod 3" function (remainder after division).

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 chaos
```

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

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 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  glVertex2d(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.

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.

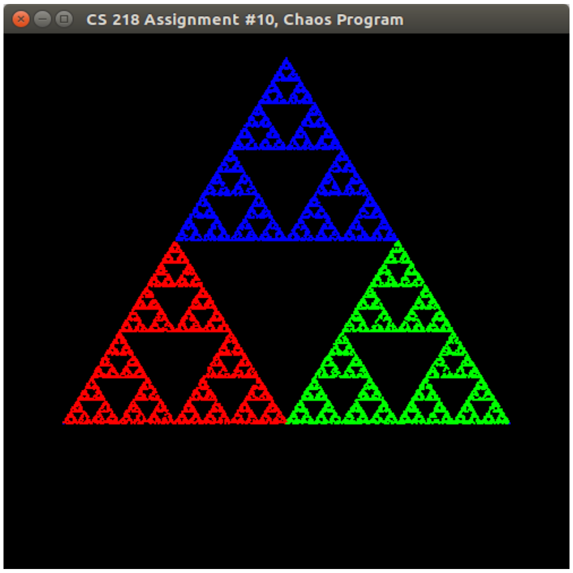
```
ed-vm% ./chaos
Usage: chaos -it <ternaryNumber> -rs <ternaryNumber>
ed-vm%
ed-vm% ./chaos -it 122122112222
Error, invalid or incomplete command line argument.
ed-vm%
```

```
ed-vm% ./chaos -ot 122122112222 -rs 121
Error, iterations specifier incorrect.
ed-vm%
ed-vm% ./chaos -it 1221221122222222
Error, iterations value must be between 100110 (3) and 10022220020 (3).
ed-vm%
ed-vm% ./chaos -it 1221221122 -rs 11222211222
Error, rotation speed value must be between 0 (3) and 1122221122 (3).
ed-vm%
```

<u>Example I/O:</u>

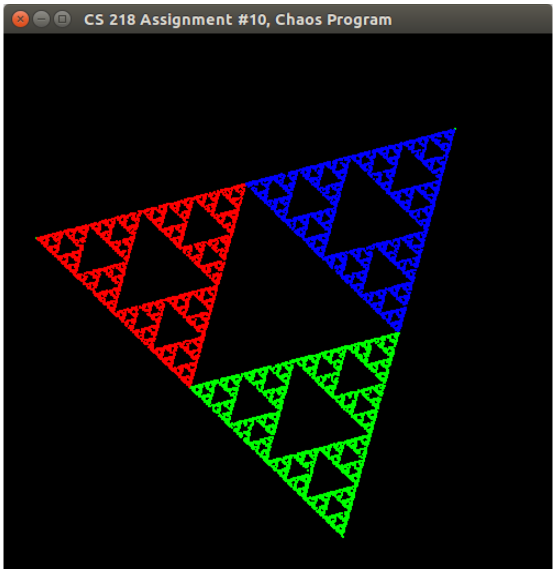
A correct output will appear similar to the following:

```
ed-vm% ./chaos -it 1221221122 -rs 12122
```



Note, the window can be terminated by typing 'q' or 'x' (while the mouse is in the window) or clicking on the **x** in the upper left corner.

Note, the image should rotate at a rate based on the used provided rotate speed.


CS 218 Assignment #10, Chaos Program