UNIX and C Programming Assingment

Semester 2 2018

Amy Trevaskis 15129275

Contents:

Introduction

The task for the assignment was to create a terminal based "Turtle Graphics" program. The program in question works by reading in text file in a specific format (as a series of commands) processing the data inside the file into x, y coordinates (along with pattern and color information) and feeding said data into a function called line() which in turn uses the data to display the image on the terminal screen.

Part of the requirements of the assignment was to use a Linked List to store the command file data, write the data out to a log file, provide a multi target make file for different compilation conditions (such as Debug and Simple modes) as well as adhering to the appropriate coding practice as outlined by Curtin University.

Description of general functions (by source file)

1. cList.c:

The program uses a double ended singly linked List to store and iterate over the data, as such all functions used in the list are in the file cList.c

createList

Takes in no parameters and returns a LinkedList pointer. Allocated memory for a LinkedList pointer then sets the head to "NULL". The function creates an empty linked list to be filled with any datatype (with a void* pointer for data)

removeFirst

Removes the head of the linked list while freeing the node and returning the data. It was used as easy access to the list data however was not needed in the final overall design. It was however a useful function to have in a generic linked list.

insertFirst

Takes in a pointer to a LinkedList struct and a void pointer to the data. Inserts the data into the front of the list (or assigns it as head if the list is unpopulated) a standard function for inserting into and populating a LinkedList.

insertLast

Works similarly to insertFirst only inserts an item at the “back” of the list through reference to a “tail” pointer. Unlike insertFirst it allows the data to be accessed in the order it was originally read. This function was primarily used when reading in data from the file.

freeList

Takes in a Linked List and frees each nodes data, node and then the list itself. An Important function to avoid unnecessary memory leaks as List structs are malloc’d.

printList

Iterates through the list accessing and printing each nodes data, Primarily used for debugging and double checking the contents of the list.

2. **read.c**

   Contains functions for reading in the file.

   readIn

   The function takes in a FILE* pointer and LinkedList* as parameters. The file is opened And the data (in this case a char* array) is inserted into the list (with the insertLast ()function) with a while -loop. The function both reads the file and populates the list.

   caseA

   The function converts a char* pointer to uppercase by subtracting the corresponding integer for each char. Used to convert all commands to uppercase to allow for validation.

3 .calc.c

Contains functions relating to calculations of x and y coordinates.

calcX

Takes in two real values representing current angle and distance. Uses cos math function and must also convert the angle to radians. Calculates the X coordinate.

calcY

Similar function to calcX only calcY uses the sin function(as it calculates the y coordinate) Also converts to radians.

printC

Prints a single char onto the screen (for the "Pattern" command from the file) is the function pointed to by the type PlotFunc. What essentially makes up the picture on the terminal.

getAngle

Returns a double representing an angle (to go into calcX and calcY functions). Makes sure the angle is not below 0 or over 360 degrees.

4. **TurtleGraphics.c (main)**

processCmd

Takes in a Linked List, iterates through the list processing the data, passing it to the line function and writing out the coordinates to a log file. It is what feeds the parameters to the line() function.

Final Problem Solution

In order to convert the file into a coordinate system: first the file was opened and read via a loop in the readIn function. Each line was read in as a string and added directly to the list after being converted to upper case.

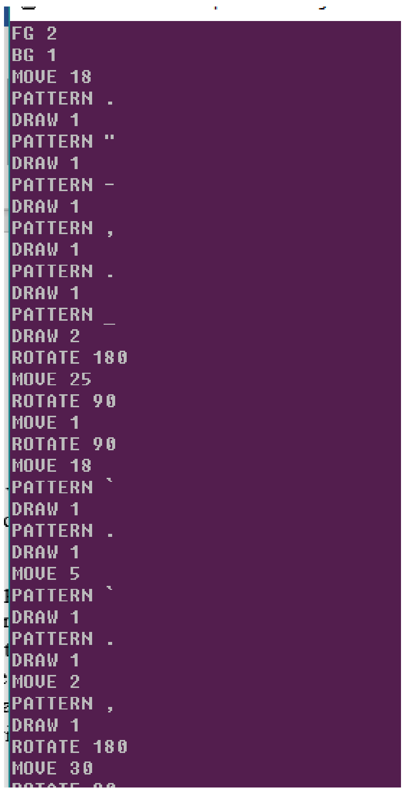
The list now having strings representing the lines of the file was passed to the processCmd function where the first char of each line was checked with a corresponding char (‘D’ for draw, ‘M’ for move ect) depending on which commend was identified the string was processed using the sscanf function to parse the needed datatype. If it was a value for the Rotate Command it was passed into calcX and calcY (along with distance) if it was the Pattern command the char was pointed to by a void pointer that would be passed to line along with a pointer to a function (the pointer being PlotFunc and the function being printC)

Throughout the loop all the data was processed and passed into line, generating the image on screen. After the loop was complete the Linked List was freed. The unrounded output of the calcX and calcY functions was written out to a long file.

Another alternate approach might have been to use an individual struct to hold the file information. For example a “command” struct with a char* for the name/command and a void pointer for the data rather than reading strings directly into the list.

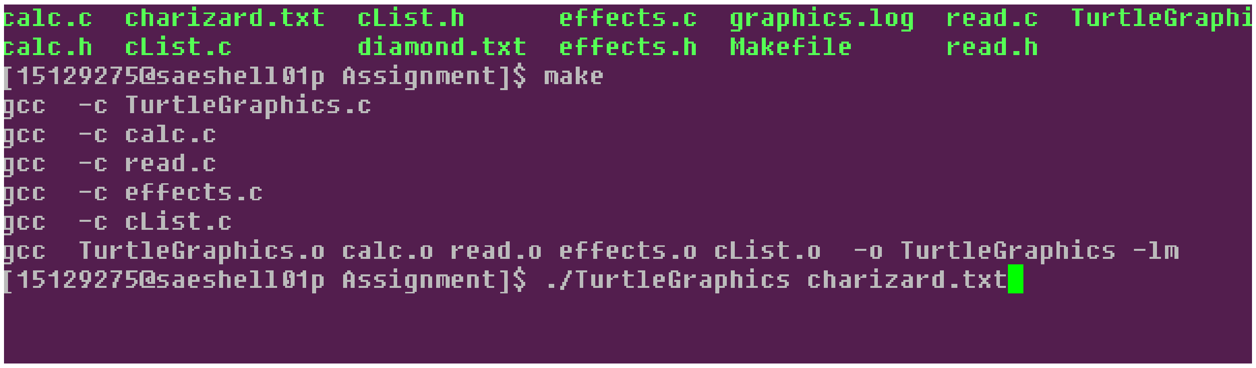
Testing

This is an example of (part of) the file used to test the program:



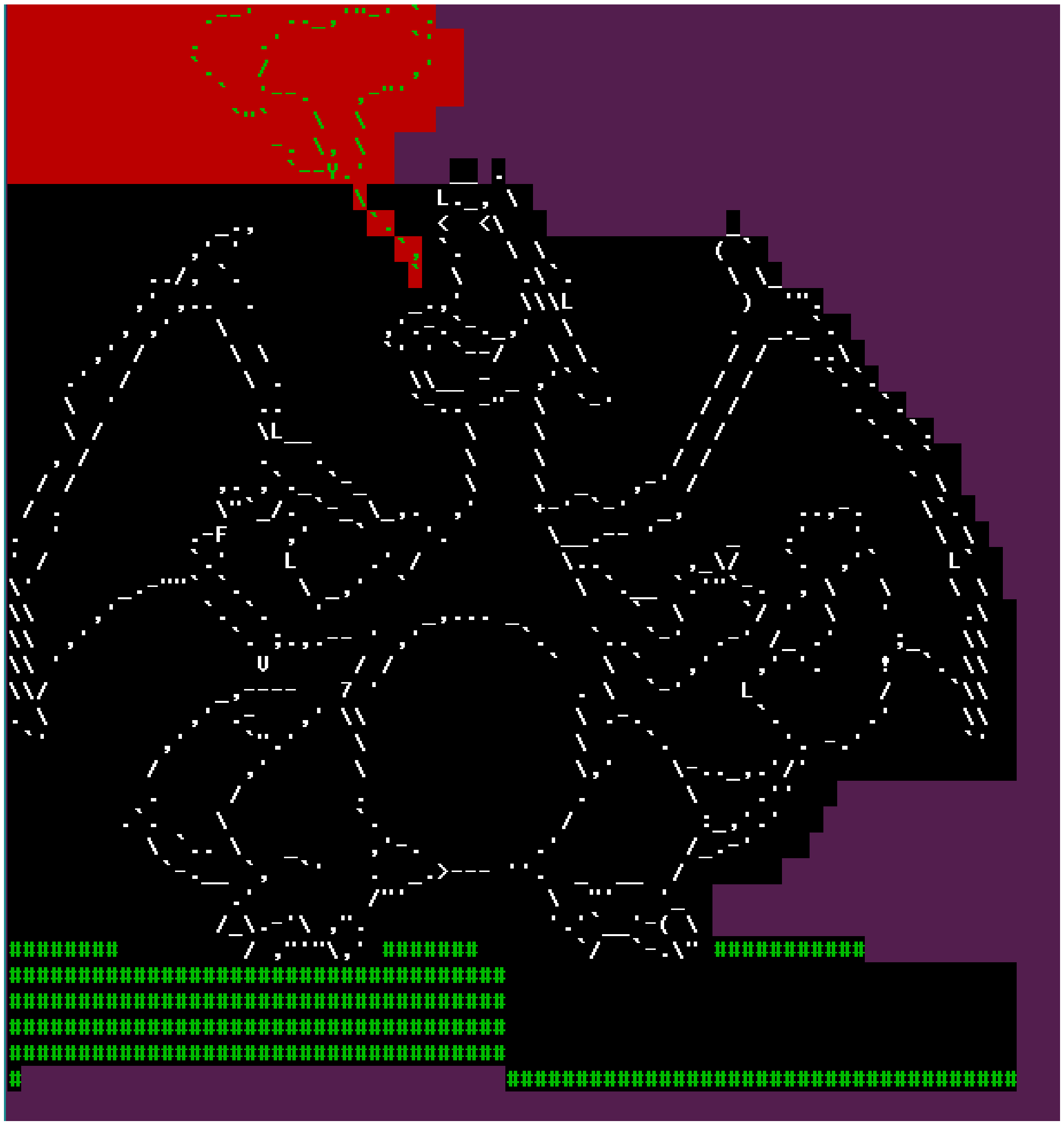
```
FG 2
BG 1
MOVE 18
PATTERN .
DRAW 1
PATTERN "
DRAW 1
PATTERN -
DRAW 1
PATTERN ,
DRAW 1
PATTERN .
DRAW 1
PATTERN _
DRAW 2
ROTATE 180
MOVE 25
ROTATE 90
MOVE 1
ROTATE 90
MOVE 18
PATTERN `
DRAW 1
PATTERN .
DRAW 1
MOVE 5
PATTERN `
DRAW 1
PATTERN .
DRAW 1
MOVE 2
PATTERN ,
DRAW 1
ROTATE 180
MOVE 30
```

the following the command Line was used:



```
calc.c   charizard.txt   cList.h      effects.c   graphics.log   read.c   TurtleGraphi
calc.h   cList.c         diamond.txt  effects.h   Makefile       read.h
[15129275@saeshell01p Assignment]$ make
gcc  -c TurtleGraphics.c
gcc  -c calc.c
gcc  -c read.c
gcc  -c effects.c
gcc  -c cList.c
gcc  TurtleGraphics.o calc.o read.o effects.o cList.o  -o TurtleGraphics -lm
[15129275@saeshell01p Assignment]$ ./TurtleGraphics charizard.txt
```

Which resulted in this output:



Conclusion

Through reading in file to a Linked List of strings I was able to produce the above image which is very similar to what was required.