

UNIX and C Programming (COMP1000)

Semester 2 2018 Assignment Report

Liam Powell (19236221)

October 21, 2018

Contents

1	Brief Descriptions of Functions	3
1.1	commands.h	3
1.1.1	lookupCommand	3
1.2	effectswrapper.h	3
1.2.1	lineWrapper	3
1.2.2	clearScreenWrapper	4
1.2.3	penDownWrapper	4
1.2.4	setFgColourWrapper	4
1.2.5	setBgColourWrapper	4
1.3	tokeniser.h	4
1.3.1	tokeniseLine	4
1.4	parser.h	4
1.4.1	parseCommand	4
1.5	state.h	5
1.5.1	createState	5
1.5.2	destroyState	5
1.6	error.h	5
1.6.1	createErrorPrinters	5
1.6.2	errorPrinterChangeState	5
1.6.3	errorPrintersCheckError	5
1.6.4	destroyErrorPrinters	5
1.7	commands/command.h	5
1.7.1	CommandParser typedef	5
1.7.2	CommandHandler typedef	6
1.7.3	CommandDestructor typedef	6
1.8	commands/helpers.h	6

1.8.1	doubleParser	6
1.8.2	intParser	6
1.8.3	safeDoubleAdd	6
1.8.4	degToRad	6
1.9	fgetslength/fgetslength.h	6
1.10	linkedlist/linkedlist.h	6
1.10.1	linkedListCreate	6
1.10.2	linkedListDestroy	7
1.10.3	linkedListLength	7
1.10.4	linkedListGet	7
1.10.5	linkedListPop	7
1.10.6	linkedListInsert	7
1.11	strncasecmp/strncasecmp.h	7
1.11.1	strncasecmp	7
1.12	test/malloc.c	7
2	Input File Coordinate Conversion	8
2.1	How I Implemented It	8
2.2	Alternative Way to Implement It	8
3	Sample Input and Output	8
3.1	Spiral	8
3.1.1	Input File	8
3.1.2	Terminal Input	9
3.1.3	Terminal Output	9
3.1.4	graphics.log	10
3.2	Spiral 2	10
3.2.1	Input File	10
3.2.2	Terminal Input	11
3.2.3	Terminal Output	11
3.2.4	graphics.log	12
3.3	From Assignment Spec	12
3.3.1	Input File	12
3.3.2	Terminal Input	12
3.3.3	Terminal Output	13
3.3.4	graphics.log	13
3.4	Invalid Command	14
3.4.1	Input File	14
3.4.2	Terminal Input	14
3.4.3	Terminal Output	14

3.4.4	graphics.log	14
3.5	Out of Bounds	14
3.5.1	Input File	14
3.5.2	Terminal Input	14
3.5.3	Terminal Output	15
3.5.4	graphics.log	15
3.6	Out of Bounds 2	15
3.6.1	Input File	15
3.6.2	Terminal Input	15
3.6.3	Terminal Output	15
3.6.4	graphics.log	16
3.7	Out of Bounds 3	16
3.7.1	Input File	16
3.7.2	Terminal Input	16
3.7.3	Terminal Output	16
3.7.4	graphics.log	17

1 Brief Descriptions of Functions

See the documentation in the header files for more detailed descriptions of function usage.

1.1 `commands.h`

1.1.1 `lookupCommand`

Lookup a command by it's name, returning the relevant parser, handler, and destructor.

1.2 `effectswrapper.h`

These functions provide a wrapper around the provided effects library which integrates with the rest of the program. most of these functions don't do much except `lineWrapper`, which adds error checking and compile time configuration to the line function.

1.2.1 `lineWrapper`

Wrapper around `line()` from `effects.h`. Takes doubles instead of ints and a `DrawFunc*` instead of a `PlotFunc`. This makes command handlers a bit simpler. If `plotter` returns non-zero then this function will terminate early

and return the return value from `plotter`. Before `plotter` is called the state is updated to the point which is being plotted. If the macro `DRAW_LAST_POINT` is false then this function will not call the provided `DrawFunc` for the last point on a line. This function also does some input range checking, details about that are in `config.h`.

1.2.2 clearScreenWrapper

Wrapper around `clearScreen()` from `effects.h`. Currently this function adds no additional functionality and only exists for the sake of having a consistent API.

1.2.3 penDownWrapper

Wrapper around `penDown()` from `effects.h`. Changes the cursor position in the drawing state to `(0, 9999)`.

1.2.4 setFgColourWrapper

Wrapper around `setFgColour()` from `effects.h`. Changes the foreground struct field in the drawing state to the correct colour code.

1.2.5 setBgColourWrapper

Wrapper around `setBgColour()` from `effects.h`. Changes the background struct field in the drawing state to the correct colour code.

1.3 tokeniser.h

1.3.1 tokeniseLine

Puts null terminators at the end of each argument in a line of input and returns pointers to the start of each argument as well as the lengths of those arguments.

1.4 parser.h

1.4.1 parseCommand

Parses a command using the parser returned by `lookupCommand` after we separate the arguments using `tokeniseLine`.

1.5 state.h

1.5.1 createState

Initialise a `DrawingState` struct and setup the screen by setting the background and foreground colours and the clearing it.

1.5.2 destroyState

Just a wrapper around `penDownWrapper`.

1.6 error.h

These functions provide `printf` wrappers that are used throughout the program. This is used because we have three streams that we might want to print to from any function (log, debug, and error) and it's cumbersome to pass three streams around.

1.6.1 createErrorPrinters

Setup error printers.

1.6.2 errorPrinterChangeState

Change the state of the error printers when there is a drawing on the screen, this allows us to avoid printing over the drawing if an error occurs.

1.6.3 errorPrintersCheckError

This is used to check if an error occurred during any call to `printf`, We only check this once near the end of the program since failing to print to a log, debug, or error file will not affect the rest of the program.

1.6.4 destroyErrorPrinters

Cleans up resources used by `createErrorPrinters`.

1.7 commands/command.h

1.7.1 CommandParser typedef

Parses the arguments of a command. This function will be called exactly once for for a given `Command` struct. This function is free to change any field in the `out`, the default values are provided for convenience.

1.7.2 CommandHandler typedef

Handles a command. This function will be called zero or more times with a `Command` struct that was previously passed to the associated `CommandParser` exactly once and passed to a `CommandHandler` zero or more times. This function is free to change any value in the `Command` struct. If the parser returned non-zero this function will never be called.

1.7.3 CommandDestructor typedef

Frees any resources in a `Command` struct. This function will be called exactly once with a `Command` struct that was previously passed to a `CommandParser` exactly once and passed to a `CommandHandler` zero or more times. This function is free to change any value in the `Command` struct.

1.8 commands/helpers.h

1.8.1 doubleParser

Parses a string to a double within a range.

1.8.2 intParser

Parses a string to an int within a range.

1.8.3 safeDoubleAdd

Adds two doubles if they can be added without causing overflow.

1.8.4 degToRad

Converts from degrees to radians.

1.9 fgetslength/fgetslength.h

Line reading function similar to `fgets` except the size parameter has been changed to `size_t` and the return value is the number of characters that were read. This allows us to handle lines that contain `'\0'`.

1.10 linkedlist/linkedlist.h

1.10.1 linkedListCreate

Create a new empty linked list.

1.10.2 linkedListDestroy

Destroy a linked list. No functions can be used on a linked list after this function is called on it.

1.10.3 linkedListLength

Get the length of a linked list.

1.10.4 linkedListGet

Get an element from a linked list.

1.10.5 linkedListPop

Remove an element from the linked list and return it.

1.10.6 linkedListInsert

Insert an element at the index, pushing other nodes forward.

1.11 strncasecmp/strncasecmp.h

1.11.1 strncasecmp

This is my own implementation of strncasecmp from POSIX.

Reference: <http://pubs.opengroup.org/onlinepubs/9699919799/functions/strncasecmp.html>

1.12 test/malloc.c

This file is used to make sure that the program can handle malloc failing at any point. It calls the main function of the program and makes the first call to malloc and all calls after that fail, then it calls main again and makes the second call to malloc and all calls after that fail, etc. until the program doesn't reach the failing malloc call, meaning that we've tested malloc failing at every possible point. Then we do the same thing again except only one call to malloc fails and the rest are allowed to succeed.

2 Input File Coordinate Conversion

2.1 How I Implemented It

In `rotateHandler` I add the angle to the current angle in the state struct, stored in degrees. In `moveHandler` I get the polar coordinates from the distance and the angle stored in the state struct, I then convert the polar coordinates to cartesian coordinates and then add them to the current coordinates in the state struct. In `drawHandler` I get the polar coordinates from the distance and the angle stored in the state struct, I then convert the polar coordinates to cartesian coordinates and then use them to call `lineWrapper`.

2.2 Alternative Way to Implement It

I could have instead made `lineWrapper()` take a distance and an angle. I didn't take this approach because I wanted `effectswrapper.h` to be similar to `effects.h`.

3 Sample Input and Output

I've swapped black and white in the outputs.

3.1 Spiral

3.1.1 Input File

```
1 rotate -45
2     move      20
3 fg  1
4 bg  2
5 rotate -45
6 draw 1
7 rotate 45
8 fg 2
9 bg 3
10 draw 3
11 rotate 45
12 fg 3
13 bg 4
14 draw 3
15 rotate 45
```



```

16 fg 4
17 bg 5
18 draw 5
19 rotate 45
20 fg 5
21 bg 6
22 draw 5
23 rotate 45
24 fg 7
25 bg 0
26 draw 7
27 rotate 45
28 fg 8
29 bg 1
30 draw 7
31 rotate 45
32 fg 9
33 bg 2
34 draw 9
35 rotate 45
36 fg 10
37 bg 3
38 draw 9
39 rotate 45

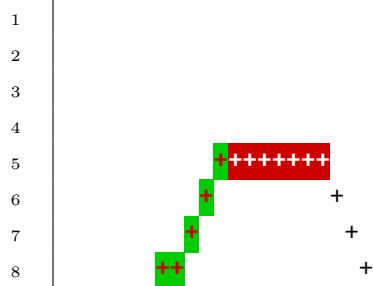
```

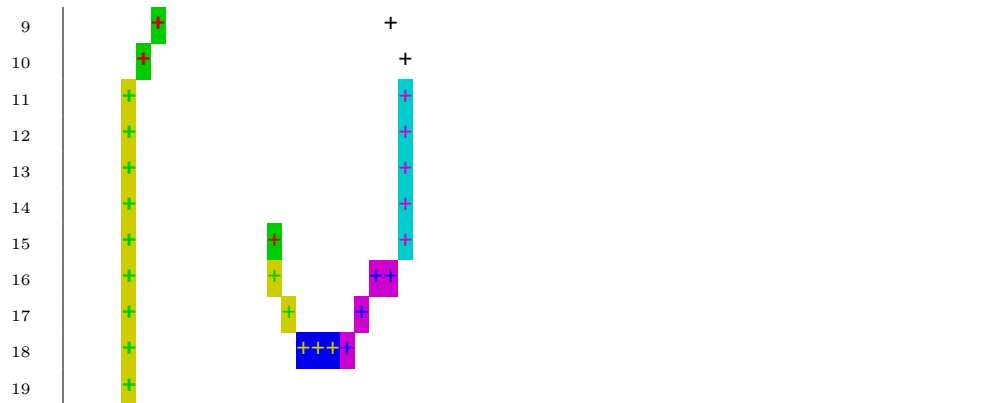
3.1.2 Terminal Input

```
1 ./TurtleGraphics test/inputs/spiral.txt
```

3.1.3 Terminal Output

stdout:





stderr:

1

3.1.4 graphics.log

```

1 ---
2 MOVE ( 0.000, 0.000)-( 14.142, 14.142)
3 DRAW ( 14.142, 14.142)-( 14.142, 15.142)
4 DRAW ( 14.142, 15.142)-( 16.263, 17.263)
5 DRAW ( 16.263, 17.263)-( 19.263, 17.263)
6 DRAW ( 19.263, 17.263)-( 22.799, 13.728)
7 DRAW ( 22.799, 13.728)-( 22.799, 8.728)
8 DRAW ( 22.799, 8.728)-( 17.849, 3.778)
9 DRAW ( 17.849, 3.778)-( 10.849, 3.778)
10 DRAW ( 10.849, 3.778)-( 4.485, 10.142)
11 DRAW ( 4.485, 10.142)-( 4.485, 19.142)

```

3.2 Spiral 2

3.2.1 Input File

```

1 rotate -45
2 move 4
3 rotate 45
4 fg 0
5 bg 1
6 draw 1
7 rotate 90

```

```

8 draw 1
9 rotate 90
10 fg 1
11 bg 2
12 draw 2
13 rotate 90
14 draw 2
15 rotate 90
16 fg 2
17 bg 3
18 draw 3
19 rotate 90
20 draw 3
21 rotate 90
22 fg 3
23 bg 4
24 draw 4
25 rotate 90
26 draw 4
27 rotate 90
28 fg 4
29 bg 5
30 draw 5
31 rotate 90
32 draw 5
33 rotate 90

```

3.2.2 Terminal Input

```
1 ./TurtleGraphics test/inputs/spiral2.txt
```

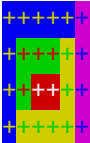
3.2.3 Terminal Output

stdout:

```

1
2
3
4
5

```



```
6 | ++++++
```

```
stderr:  
1 |
```

3.2.4 graphics.log

```
1 ---  
2 MOVE ( 0.000, 0.000)-( 2.828, 2.828)  
3 DRAW ( 2.828, 2.828)-( 3.828, 2.828)  
4 DRAW ( 3.828, 2.828)-( 3.828, 1.828)  
5 DRAW ( 3.828, 1.828)-( 1.828, 1.828)  
6 DRAW ( 1.828, 1.828)-( 1.828, 3.828)  
7 DRAW ( 1.828, 3.828)-( 4.828, 3.828)  
8 DRAW ( 4.828, 3.828)-( 4.828, 0.828)  
9 DRAW ( 4.828, 0.828)-( 0.828, 0.828)  
10 DRAW ( 0.828, 0.828)-( 0.828, 4.828)  
11 DRAW ( 0.828, 4.828)-( 5.828, 4.828)  
12 DRAW ( 5.828, 4.828)-( 5.828, -0.172)
```

3.3 From Assignment Spec

This file is from the assignment specification.

3.3.1 Input File

```
1 rotate -45  
2 move 30  
3 FG 1  
4 Pattern #  
5 DRAW 10  
6 Rotate 90  
7 draw 10  
8 ROTATE 90  
9 dRAW 10  
10 ROTATE 90  
11 DRAW 10
```

3.3.2 Terminal Input

```
1 | ./TurtleGraphics test/inputs/fromSpec.txt
```

3.3.3 Terminal Output

stdout:

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15      #
16     # #
17    #   #
18   #     #
19  #       #
20 #         #
21 #         #
22 #         #
23 #         #
24 #         #
25 #         #
26 #         #
27 #         #
28 #         #
29      #
```

stderr:

```
1
```

3.3.4 graphics.log

```
1 ---
2 MOVE ( 0.000, 0.000)-( 21.213, 21.213)
```

```
3 DRAW ( 21.213, 21.213)-( 28.284, 28.284)
4 DRAW ( 28.284, 28.284)-( 35.355, 21.213)
5 DRAW ( 35.355, 21.213)-( 28.284, 14.142)
6 DRAW ( 28.284, 14.142)-( 21.213, 21.213)
```

3.4 Invalid Command

This file contains an invalid command.

3.4.1 Input File

```
1 not a real command
```

3.4.2 Terminal Input

```
1 ./TurtleGraphics test/inputs/invalidCommands.txt
```

3.4.3 Terminal Output

stdout:

```
1
```

stderr:

```
1 Unrecognised command on line 1.
```

3.4.4 graphics.log

3.5 Out of Bounds

3.5.1 Input File

```
1 draw -5
```

3.5.2 Terminal Input

```
1 ./TurtleGraphics test/inputs/outOfBounds.txt
```

3.5.3 Terminal Output

stdout:

```
1
```

stderr:

```
1 Tried to draw a line out of bounds on line 1, lines can only be d
2 rawn between (0, 0) and (32767, 32767).
3 The compile time variable DRAW_LAST_POINT is false, which means t
4 he line you tried to draw might be in bounds but the end of the l
5 ine is out of bounds. This behaviour is intentional as it ensures
6 that DRAW_LAST_POINT will not affect whether or not an input fil
7 e is valid.
```

3.5.4 graphics.log

```
1 ---
2 DRAW ( 0.000, 0.000)-( -5.000, 0.000)
```

3.6 Out of Bounds 2

3.6.1 Input File

```
1 draw 99999999999
```

3.6.2 Terminal Input

```
1 ./TurtleGraphics test/inputs/outOfBounds2.txt
```

3.6.3 Terminal Output

stdout:

```
1
```

stderr:

```
1 Tried to draw a line out of bounds on line 1, lines can only be d
2 rawn between (0, 0) and (32767, 32767).
3 The compile time variable DRAW_LAST_POINT is false, which means t
4 he line you tried to draw might be in bounds but the end of the l
5 ine is out of bounds. This behaviour is intentional as it ensures
```

```
6 | that DRAW_LAST_POINT will not affect whether or not an input fil
7 | e is valid.
```

3.6.4 graphics.log

```
1 | ---
2 | DRAW ( 0.000, 0.000)-(9999999999.000, 0.000)
```

3.7 Out of Bounds 3

3.7.1 Input File

```
1 | rotate -45
2 | draw 10
3 | rotate -90
4 | draw 20
```

3.7.2 Terminal Input

```
1 | ./TurtleGraphics test/inputs/outOfBounds3.txt
```

3.7.3 Terminal Output

stdout:

```
1 | +
2 |  +
3 |   +
4 |    +
5 |     +
6 |      +
7 |       +
```

stderr:

```
1 | Tried to draw a line out of bounds on line 4, lines can only be d
2 |rawn between (0, 0) and (32767, 32767).
3 | The compile time variable DRAW_LAST_POINT is false, which means t
4 |he line you tried to draw might be in bounds but the end of the l
5 |ine is out of bounds. This behaviour is intentional as it ensures
6 | that DRAW_LAST_POINT will not affect whether or not an input fil
7 |e is valid.
```


3.7.4 graphics.log

```
1 ---  
2 DRAW ( 0.000, 0.000)-( 7.071, 7.071)  
3 DRAW ( 7.071, 7.071)-( -7.071, 21.213)
```