CSC 412 - Programming Assignment 06, Fall 2019

Jordan Coats

***Abstract*—The objective of this assignment is to get further experience with the pthread library, experiment with synchronization of pthreads, and learn to use a third-party library in C.**

I. INTRODUCTION

This assignment was to create a ”traveler” thread that moves randomly on a grid and leaves a color trail as they move. They only stop when they reach a corner and terminate. They will color their trail as long as the inkwells of their particular color is filled by pressing the first letter of each color’s corresponding key.

II. TRAVELLIST

To start with I would like discuss the travelList struct.

```
  travelList[k].type = (TravelerType)
        (rand() % NUM_TRAV_TYPES);
travelList[k].row = rand() % NUM_ROWS;
travelList[k].col = rand() % NUM_COLS;
travelList[k].dir = (TravelDirection)
     (rand() % NUM_TRAVEL_DIRECTIONS);
travelList[k].isLive = 1;
pthread_create(&travelList[k].threadID,
    NULL, moveTraveler, &travelList[k]);
numLiveThreads++;
```

It was made up with a color type of traveler, what position it starts in by row and col, which direction it will face initially, and if it is on or not. With this struct I created a for loop that fills in these variables with randomized data and then creates a thread to start the traveler’s movements. The most difficult part of this was creating the pthread but was simple enough with a little guidance.

III. MOVETRAVELER - COLORS

This moveTraveler function is used by the threads to actually move the traveler’s around the grid. Each thread has unique data from the previous for loop so this function takes that data and first while the thread is active, it finds out which color it is from the the type and then once it matches that it will create the color trail behind the traveler. It does this by making sure the inkwell has enough ink then it creates a mask with the particular grid and color to make sure those bits stay the same. From their it checks that the ink in the grid isn’t going to overflow and if it is it will reset it back to max instead of increasing. Next if it is below it will increase it by 32 then change the grids color wit an OR. Finally it decreases the color’s inkwell. The only difference between the three colors is shifting the bits to make sure we’re working with blue (16) or green (8) and a shift back when it’s finished changing those values. This is hopefully the correct implementation for the first extra credit for the color’s to gradually fill instead of solid on the first pass.

IV. MOVETRAVELER - DIRECTION

The next part of my moveTraveler function is the directions. To handle the choice with a bit of randomness I first take account of the direction it is facing and then only pull from the perpendicular options. I push those valid directions to a vector that is then, by using the rand() function, I change the direction for the traveler. Once the traveler has it’s new direction it will move along the grid in that corresponding way. Such as in the case of North:

```
if (t->dir == NORTH)
{
t->row++;
}
```

Finally the last part of this function then looks for if the traveler is inside one of the corners and if it will terminate itself or it sleeps.

V. OTHER ISSUES

For the current implementation of the travelers the only issue I can find at the moment is the jitteriness of the arrows which is caused by the simple AI implementation and the corners do not color because of the order of the code. That issue can be fixed most likely with putting the color code after the movement code but it isn’t a high priority to fix.

VI. CONCLUSION

This was a very fun assignment and very interesting use of threads to control the travelers and I felt like it was a great educational experience with the bitwise operations.