Richard Burke, Curtis Hartman

Report

1. Document and justify your design and implementation decisions

We created many pointers to certain thing such as the Travelers to make passing them from function-to-function much easier. A huge focus of our project was implementing ways to make the code flexible. For example, if one wanted to change the programming of the traveler you would just have to change one function. Also when it comes to dealing with the pthreads you can change it and all the pthreads will be changed. We also took into consideration if the vectors were to change during the program, such as if a traveler were to be deleted mid-process the rest of the code wouldn’t be affected.

1. List current limitations of your program

While the travelers are able to move about the grid correctly, not bumping into any walls, other travelers, or themselves, the AI is not designed to be able to get to the exit other than by pure luck. The traveler will just randomly move, possibly looping in on itself and not being able to move anymore. The travelers are not able to prevent themselves from getting themselves trapped.

1. Detail whatever difficulties you may have run into

Learning how to use OpenGL and GLUT was very time consuming and very confusing. We don’t mean messing with the glut functions but just understanding how the callback system works and trying to understand why things didn’t/don’t visually appear despite the values being changed. Trying to understand the starter code was very difficult as anything having to do with drawing on the screen, i.e. glutmainloop(), was difficult to understand and made the first few versions of the project difficult to get working. We tried to implement the segments to follow each other after the head but visually it doesn't work. The code isn’t faulty (causing crashes or w/e) but out lack of understanding the magic behind glut and OpenGL just makes it very hard to understand why things aren’t visually matching what we have made. On exit, the traveler is progressively deleted, but a ‘ghost’ of the head respawns at the original spawn point, and this ghost head doesn't count as an obstacle, it's just visual problem. We hard coded the number of columns and number of rows, as well as the number of travelers. When we tried to input argc and argv normally, we kept getting an error saying we were missing the strtol\_l c library. Unfortunately we weren’t able to find any resolutions to that issue and ended up biting the bullet and going ahead with the hard coded values. When it came to programming the game version it proved very difficult to prevent the player from seg-faulting when trying to go out of the grid.

*(If we didn’t have the issues with argc & argv this was our intended input)*

Execution of Version 1 -

Execution line -

g++ -Wall -g main.cpp gl\_frontEnd.cpp -lm -lGL -lglut -pthread -o [name of executable]

Execution of program -

./[name of executable] [number of rows] [number of columns]

Execution of Version 2 and onward-

Execution of program -

./[name of executable] [number of rows] [number of columns] [number of travelers]

But the actual cmd is just ./[name of executable file]

Extra Credit Section

4.1 Progressive disappearance (4 pts)

Mandatory for solo developers When a traveler’s “head” reaches the exit, the traveler doesn’t disappear immediately, but one segment at a time.

Present in all versions.

4.3 Extra credit (5 pts): Deadlock resolution

1. Discussion of the problem

If 2 travelers met at the same place in an alleyway without anywhere else to go, (i.e. One heading from the North and the other from the South with nowhere to go to change direction), this would create a deadlock, as the program would give one traveler the lock to grab the last free spot and then the travelers would have nowhere to go.

1. Outline of how you would solve it.

In a game perspective, whichever traveler gets that lock first kills the other one, resolving the deadlock and allowing a traveler to continue to move without hitting having to go through the other traveler. To outline this, we would have whichever traveler that doesn’t get the lock be killed and disappear, and do this in the same way that a traveler disappears when they loop into themselves. If a traveler traps themselves then that traveler would also die.

New Extra Credit [Game Version]

1. Take control of a traveler [5 pts]

In this version, the user of the program should be able to use the usual w-a-s-d key to control the displacements of the first traveler thread (index 0). That thread would still be submitted to the same synchronization constraints as the regular "autonomous" threads.

3. Take control of any traveler [5 pts]

In this version, the user of the program can hit the index (0-9) of a thread to take control of the corresponding traveler, without waiting for the traveler previously controlled to make it to the exit. When the user selects a new traveler to control, the traveler that was user-controlled before that should revert to autonomous mode.