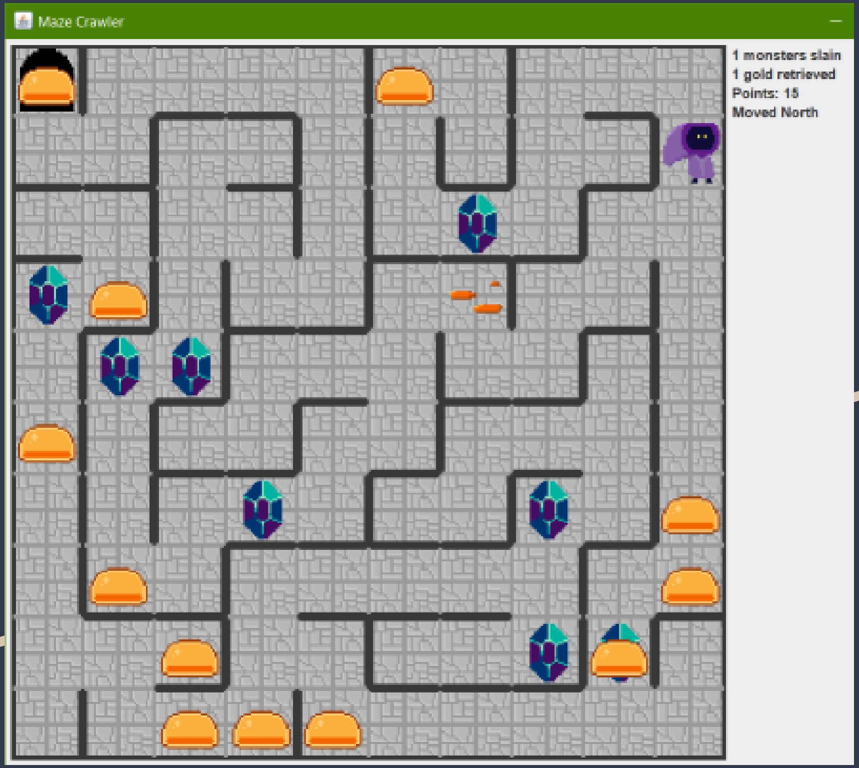
Dungeon Crawl AI

By Hailey Carter and Craig Montgomery

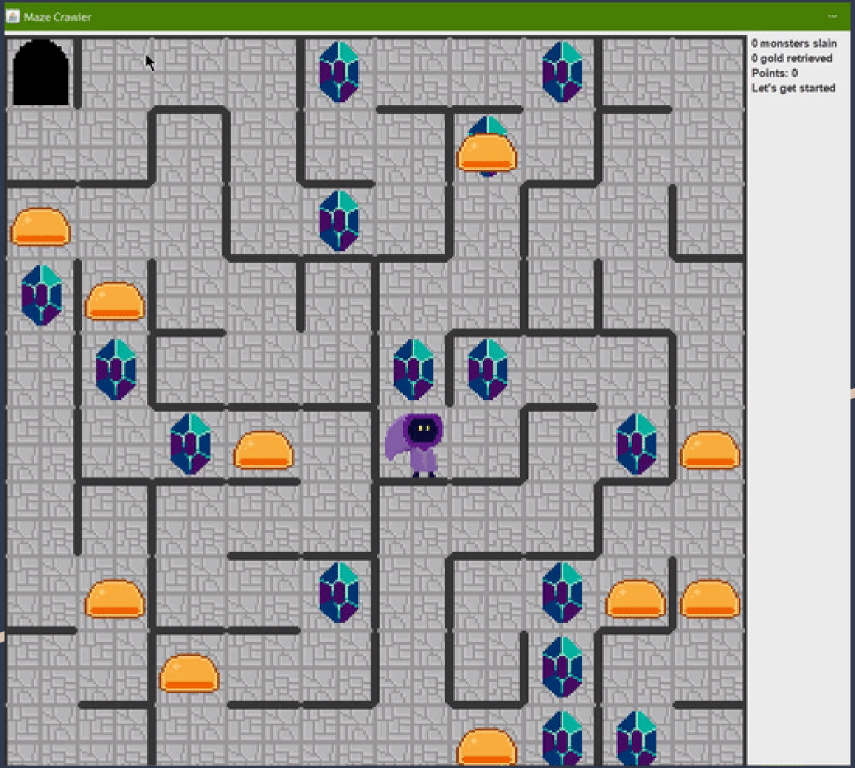
Setup



1 monsters slain
1 gold retrieved
Points: 15
Moved North

We built a simple game based off the dungeon crawl. We created a small creature that must explore a maze to find an exit, slaying slime creatures and collecting gems along the way. We then created an AI that solves the maze using a BFS. This was the basis for our project.

Process

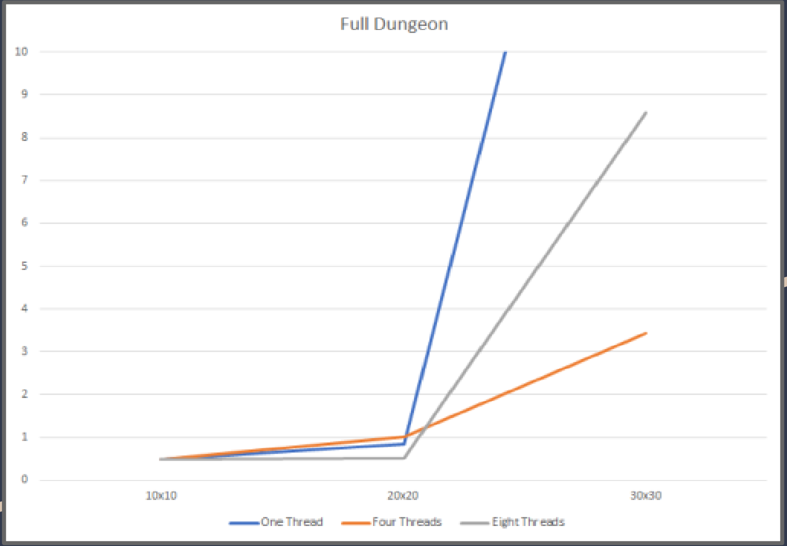


0 monsters slain
0 gold retrieved
Points: 0
Let's get started

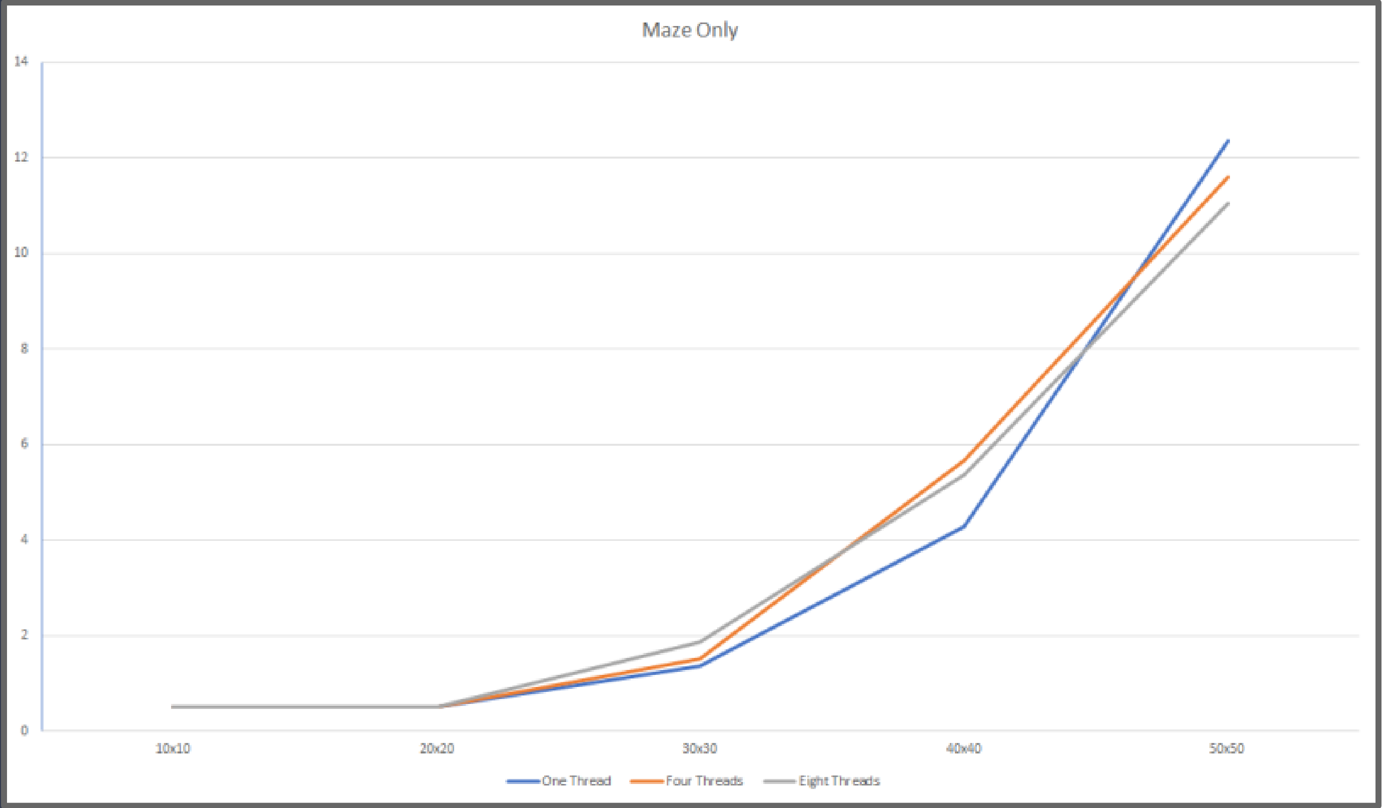
We then rebuilt the ‘brain’ of the AI to be multithreaded instead of single threaded. The threads shared a ‘work’ array of states to get the children of, and a hashtable of states that have already been visited.

We expected this multithreading to drastically improve the speed and ability of our program. But that wasn’t exactly what we got.

Results

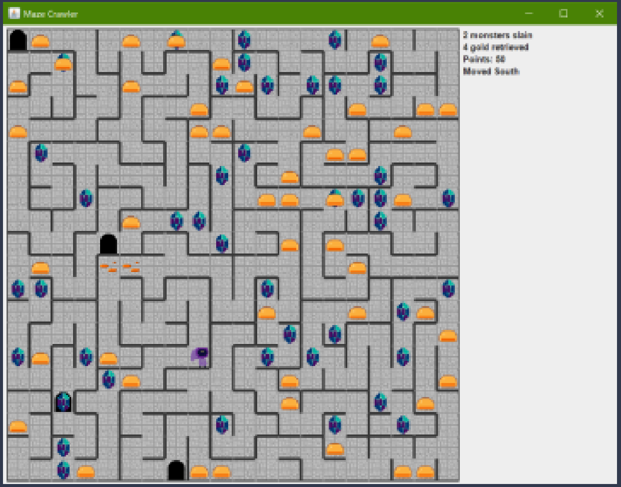


Our results were unexpected, the multithreaded process was not nearly as fast as we thought. Sometimes it was even faster for the program to run fewer threads. Although we were able to get clearer results when running the dungeon without fetching gems or slaying monsters. (See next slide)

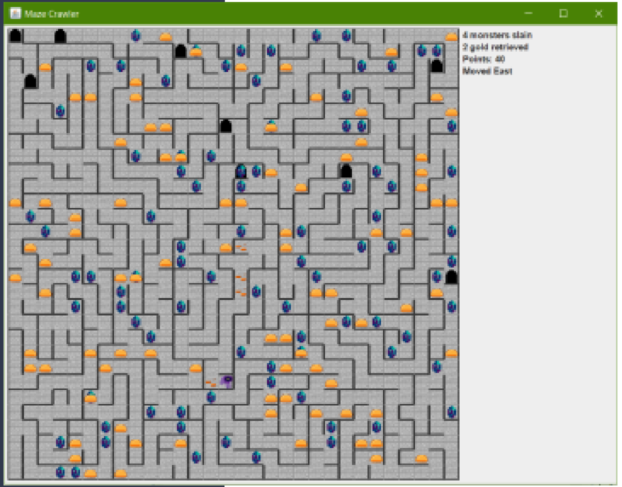


This graph, like the one previous, shows the average speeds of the program on different sized mazes, you can see we were able to test the program on much larger mazes when working without enemies and gems.

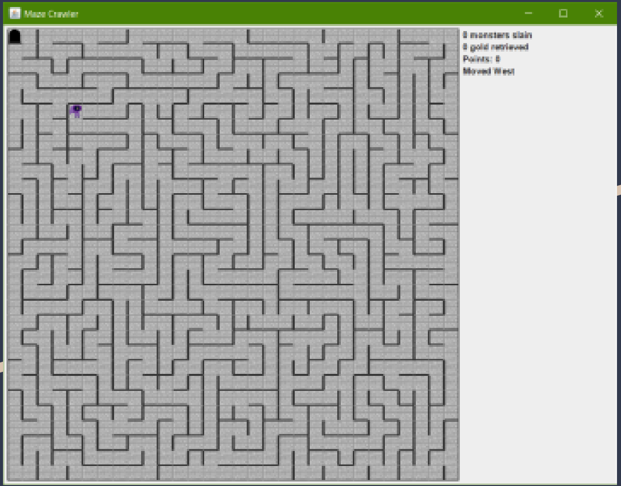
Various maze types and sizes that were tested



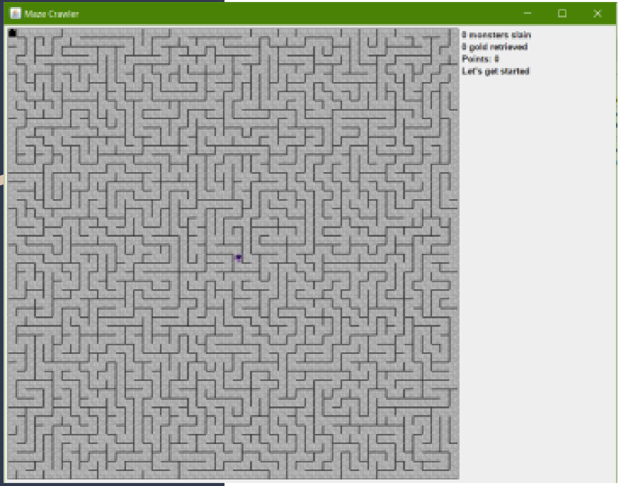
2 monsters slain
4 gold retrieved
Points: 50
Moved South



4 monsters slain
2 gold retrieved
Points: 40
Moved East



0 monsters slain
0 gold retrieved
Points: 0
Moved West



0 monsters slain
0 gold retrieved
Points: 0
Let's get started

Notable Problems

Minor problem:

Because the BFS used multiple threads, the queue and hash table needed to be thread-safe

Major Problem:

Most Java Swing classes are unsafe to use with threads.

We tried unsuccessfully to construct objects that extended JPanels within the multithreading. The constructor method for this class was not being called properly, and caused the threads to die when trying to call the constructor

Solutions

The problem of the queue and hash being thread-safe was resolved using the classes **LinkedBlockingQueue** and **ConcurrentHashMap**

The Swing problem was solved by moving the data necessary for the searching into a separate class. This allowed the threads to properly search through the maze

Moral of the story: Do not create swing objects with the threaded part of a program