Programming Assignment Report

Assignment 02

Pacman problem

Kristopher Adams
CS3030 – Operating Systems
Abraham N Aldaco Gastelum
3/1/2025

Problem

The problem given Is to design an algorithm to successfully find the single food piece on the given layout. We were tasked to design a breadth first search, depth first search, AStar, and a hill climb algorithm. For AStar and Hill climb we were tasked to use two differing heuristics, Manhattan and Euclidean.

Execution

Included in the zip is a list of every command executed that I used for this assignment. Each algorithm was used with each of the required layouts. Included are the commands to run each of the algorithms that use heuristics with both Euclidian and Manhattan. they are located in the ***commands.txt*** file.

Screenshots

(there are too many commands to screenshot every command line output) Each were ran without the Gui visible but it does not change the algorithmic output. Towards the end are the outputs showing how my function work visually.

Hillclimb partial command line outputs

```
D:\Desktop\code\AI\pacman_Python3>python pacman.py -l smallMaze -p SearchAgent -a fn=hlclimb,heuristic=manhattanHeuristic,prob=PositionSearchProblem --frameTime 0
[SearchAgent] using function hlclimb
[SearchAgent] using problem type PositionSearchProblem
Found a food at (20, 1)
Warning: this does not look like a regular search maze
Local optimum reached, stopping search.
Path found with total cost of 10 in 0.0 seconds
Search nodes expanded: 11

D:\Desktop\code\AI\pacman_Python3>python pacman.py -l smallMaze -p SearchAgent -a fn=hlclimb,heuristic=euclideanHeuristic,prob=PositionSearchProblem --frameTime 0
[SearchAgent] using function hlclimb
[SearchAgent] using problem type PositionSearchProblem
Found a food at (20, 1)
Warning: this does not look like a regular search maze
Local optimum reached, stopping search.
Path found with total cost of 10 in 0.0 seconds
Search nodes expanded: 11

D:\Desktop\code\AI\pacman_Python3>python pacman.py -l tinyMaze -p SearchAgent -a fn=hlclimb,heuristic=manhattanHeuristic,prob=PositionSearchProblem --frameTime 0
[SearchAgent] using function hlclimb
[SearchAgent] using problem type PositionSearchProblem
Found a food at (1, 1)
Path found with total cost of 8 in 0.0 seconds
Search nodes expanded: 8
Pacman emerges victorious! Score: 502
Ending graphics raised an exception: 0
Average Score: 502.0
Scores:        502.0
Win Rate:      1/1 (1.00)
Record:        Win

D:\Desktop\code\AI\pacman_Python3>python pacman.py -l tinyMaze -p SearchAgent -a fn=hlclimb,heuristic=euclideanHeuristic,prob=PositionSearchProblem --frameTime 0
[SearchAgent] using function hlclimb
[SearchAgent] using problem type PositionSearchProblem
Found a food at (1, 1)
Path found with total cost of 8 in 0.0 seconds
Search nodes expanded: 8
Pacman emerges victorious! Score: 502
Average Score: 502.0
Scores:        502.0
Win Rate:      1/1 (1.00)
Record:        Win
```

DFS partial command line outputs

```
Path found with total cost of 96 in 0.0 seconds
Search nodes expanded: 96
Pacman emerges victorious! Score: 414
Average Score: 414.0
Scores:        414.0
Win Rate:      1/1 (1.00)
Record:        Win

D:\Desktop\code\AI\pacman_Python3>python pacman.py -l openMaze -p SearchAgent -a fn=dfs,prob=PositionSearchProblem -q
[SearchAgent] using function dfs
[SearchAgent] using problem type PositionSearchProblem
Found a food at (1, 1)
Path found with total cost of 298 in 0.0 seconds
Search nodes expanded: 806
Pacman emerges victorious! Score: 212
Average Score: 212.0
Scores:        212.0
Win Rate:      1/1 (1.00)
Record:        Win

D:\Desktop\code\AI\pacman_Python3>python pacman.py -l smallMaze -p SearchAgent -a fn=dfs,prob=PositionSearchProblem -q
[SearchAgent] using function dfs
[SearchAgent] using problem type PositionSearchProblem
Found a food at (20, 1)
Warning: this does not look like a regular search maze
Path found with total cost of 28 in 0.0 seconds
Search nodes expanded: 38
Pacman emerges victorious! Score: 482
Average Score: 482.0
Scores:        482.0
Win Rate:      1/1 (1.00)
Record:        Win

D:\Desktop\code\AI\pacman_Python3>python pacman.py -l tinyMaze -p SearchAgent -a fn=dfs,prob=PositionSearchProblem -q
[SearchAgent] using function dfs
[SearchAgent] using problem type PositionSearchProblem
Found a food at (1, 1)
Path found with total cost of 10 in 0.0 seconds
Search nodes expanded: 15
Pacman emerges victorious! Score: 500
Average Score: 500.0
Scores:        500.0
Win Rate:      1/1 (1.00)
Record:        Win
```

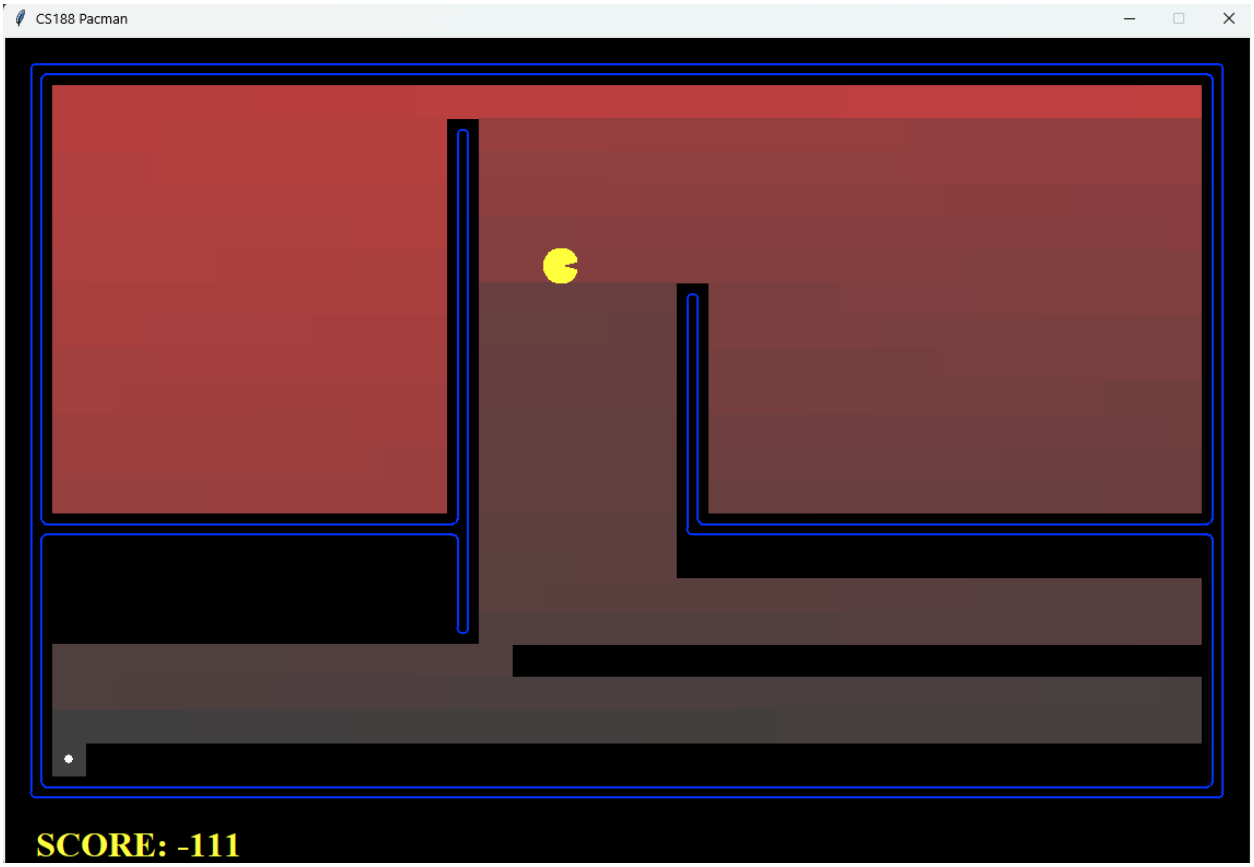
BFS partial command line outputs

```
Path found with total cost of 68 in 0.0 seconds
Search nodes expanded: 269
Pacman emerges victorious! Score: 442
Average Score: 442.0
Scores:        442.0
Win Rate:      1/1 (1.00)
Record:        Win

D:\Desktop\code\AI\pacman_Python3>python pacman.py -l mediumScaryMaze -p SearchAgent -a fn=bfs,prob=PositionSearchProblem
[SearchAgent] using function bfs
[SearchAgent] using problem type PositionSearchProblem
Found a food at (1, 1)
Path found with total cost of 72 in 0.0 seconds
Search nodes expanded: 279
Pacman died! Score: -514
Average Score: -514.0
Scores:        -514.0
Win Rate:      0/1 (0.00)
Record:        Loss

D:\Desktop\code\AI\pacman_Python3>python pacman.py -l openMaze -p SearchAgent -a fn=bfs,prob=PositionSearchProblem -q
[SearchAgent] using function bfs
[SearchAgent] using problem type PositionSearchProblem
Found a food at (1, 1)
Path found with total cost of 54 in 0.0 seconds
Search nodes expanded: 682
Pacman emerges victorious! Score: 456
Average Score: 456.0
Scores:        456.0
Win Rate:      1/1 (1.00)
Record:        Win

D:\Desktop\code\AI\pacman_Python3>python pacman.py -l smallMaze -p SearchAgent -a fn=bfs,prob=PositionSearchProblem -q
[SearchAgent] using function bfs
[SearchAgent] using problem type PositionSearchProblem
Found a food at (20, 1)
Warning: this does not look like a regular search maze
Path found with total cost of 16 in 0.0 seconds
Search nodes expanded: 82
Pacman emerges victorious! Score: 494
Average Score: 494.0
Scores:        494.0
Win Rate:      1/1 (1.00)
Record:        Win
```

ASTAR partial command line outputs

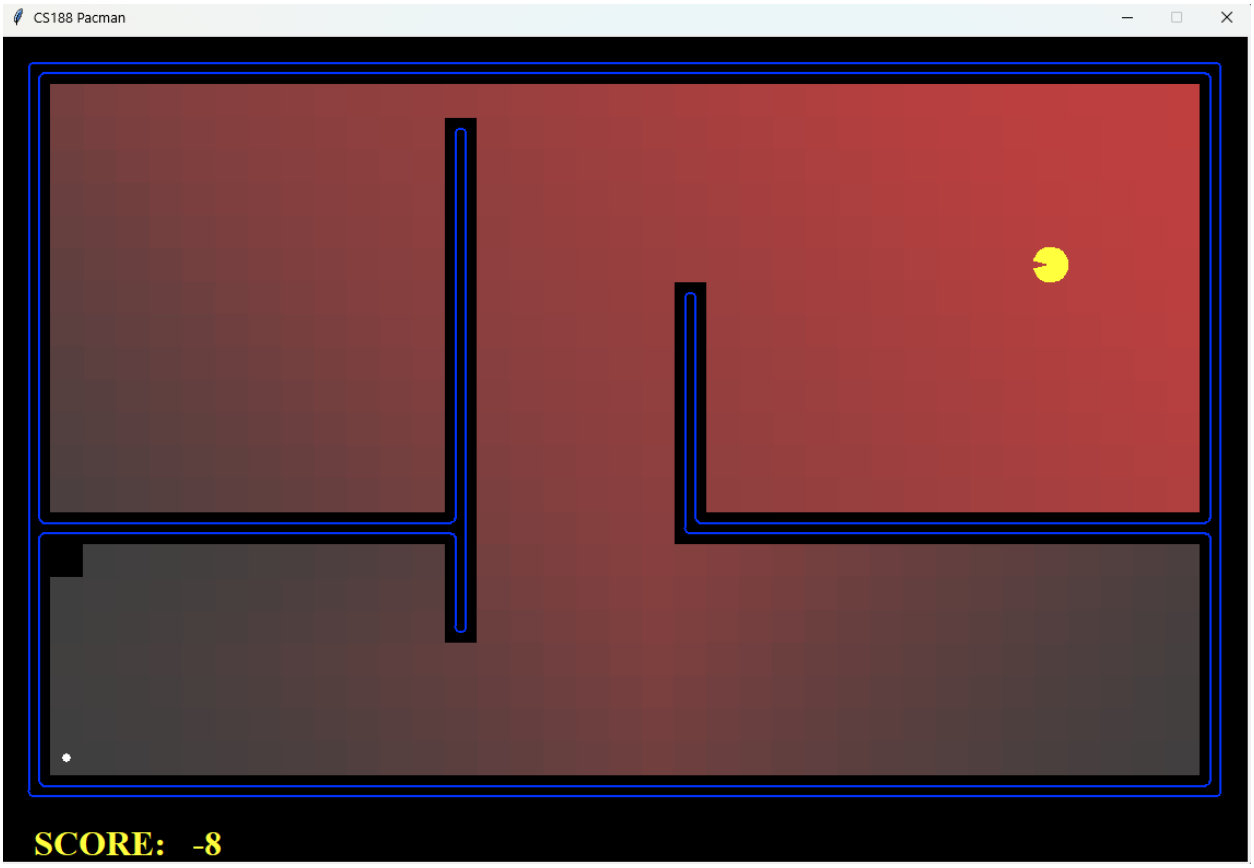
```
Path found with total cost of 72 in 0.0 seconds
Search nodes expanded: 238
Pacman died! Score: -513
Average Score: -513.0
Scores:        -513.0
Win Rate:      0/1 (0.00)
Record:        Loss

D:\Desktop\code\AI\pacman_Python3>python pacman.py -l mediumScaryMaze -p SearchAgent -a fn=astar,prob=PositionSearchProblem,heuristic=euclideanHeuristic -q
[SearchAgent] using function astar and heuristic euclideanHeuristic
[SearchAgent] using problem type PositionSearchProblem
Found a food at (1, 1)
Path found with total cost of 72 in 0.0 seconds
Search nodes expanded: 253
Pacman died! Score: -516
Average Score: -516.0
Scores:        -516.0
Win Rate:      0/1 (0.00)
Record:        Loss

D:\Desktop\code\AI\pacman_Python3>python pacman.py -l openMaze -p SearchAgent -a fn=astar,prob=PositionSearchProblem,heuristic=manhattanHeuristic -q
[SearchAgent] using function astar and heuristic manhattanHeuristic
[SearchAgent] using problem type PositionSearchProblem
Found a food at (1, 1)
Path found with total cost of 54 in 0.0 seconds
Search nodes expanded: 535
Pacman emerges victorious! Score: 456
Average Score: 456.0
Scores:        456.0
Win Rate:      1/1 (1.00)
Record:        Win

D:\Desktop\code\AI\pacman_Python3>python pacman.py -l openMaze -p SearchAgent -a fn=astar,prob=PositionSearchProblem,heuristic=euclideanHeuristic -q
[SearchAgent] using function astar and heuristic euclideanHeuristic
[SearchAgent] using problem type PositionSearchProblem
Found a food at (1, 1)
Path found with total cost of 54 in 0.0 seconds
Search nodes expanded: 550
Pacman emerges victorious! Score: 456
Average Score: 456.0
Scores:        456.0
Win Rate:      1/1 (1.00)
Record:        Win
```

DFS GUI



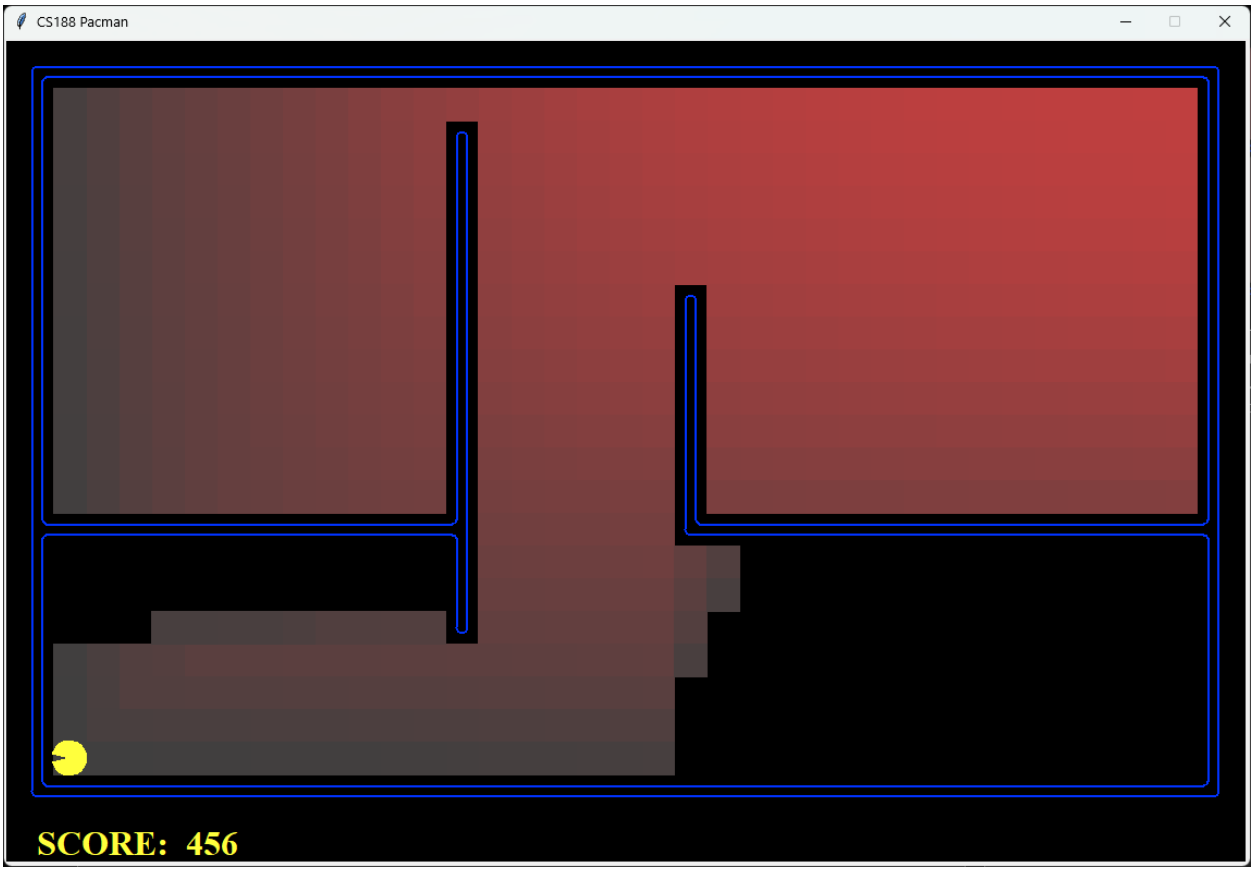
In this example it searches the Open maze layout for the furthest point to the left and ends up getting stuck and then having to find a differing path that instead goes down at some point. It gets stuck again on the right side so it again has to find a path that does not go right into the second catch.

BFS GUI



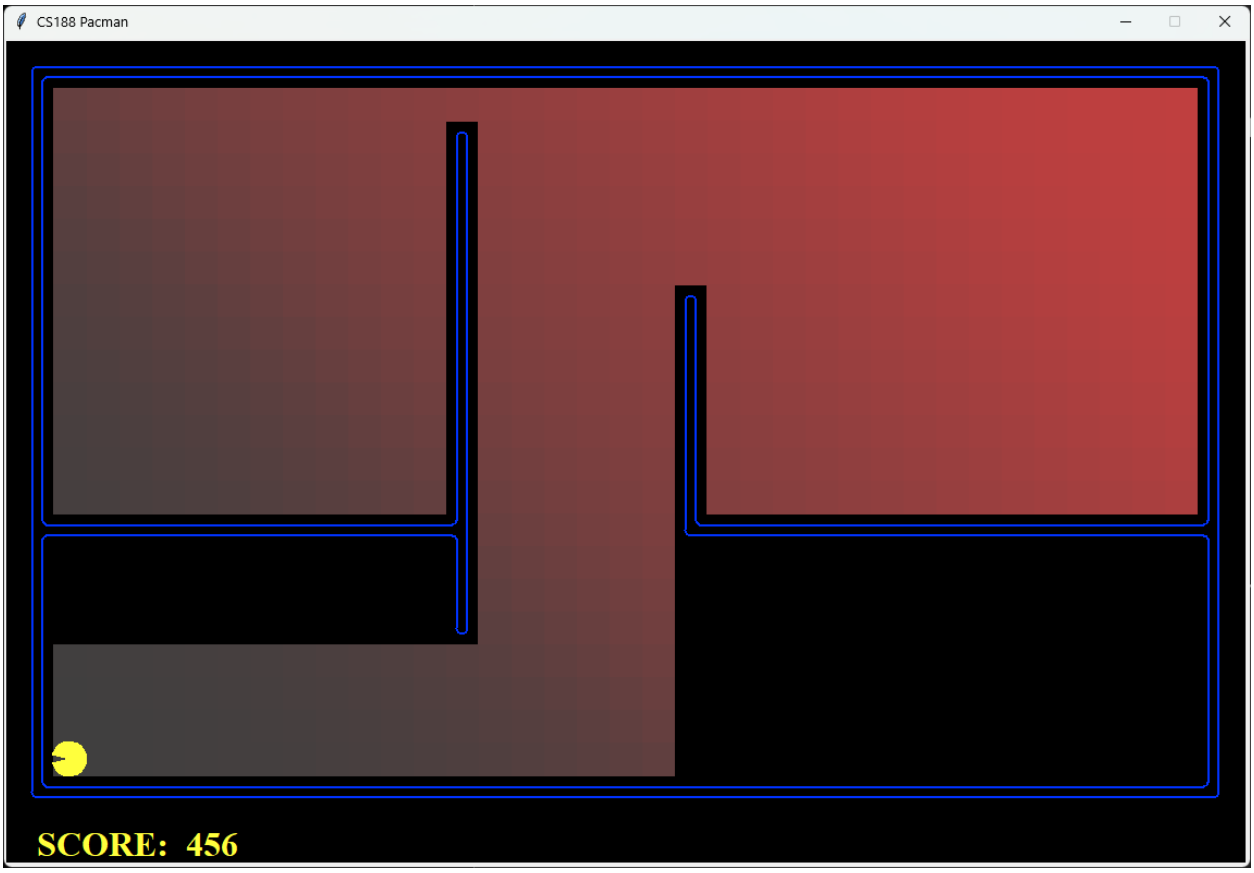
This is using the Open maze layout and is instead searching outward in a diamond pattern to get the best result. The visualization shows it got stuck on the right catch. This caused it to find a path that cut left to get to the food piece.

A* Euclidean GUI



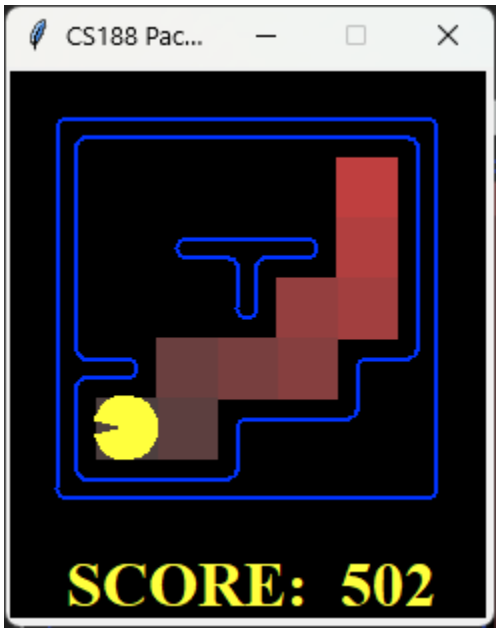
Using the Euclidian heuristic this algorithm was able to find the optimal solution to the problem. It also did not explore the right lower block.

A* Manhattan GUI



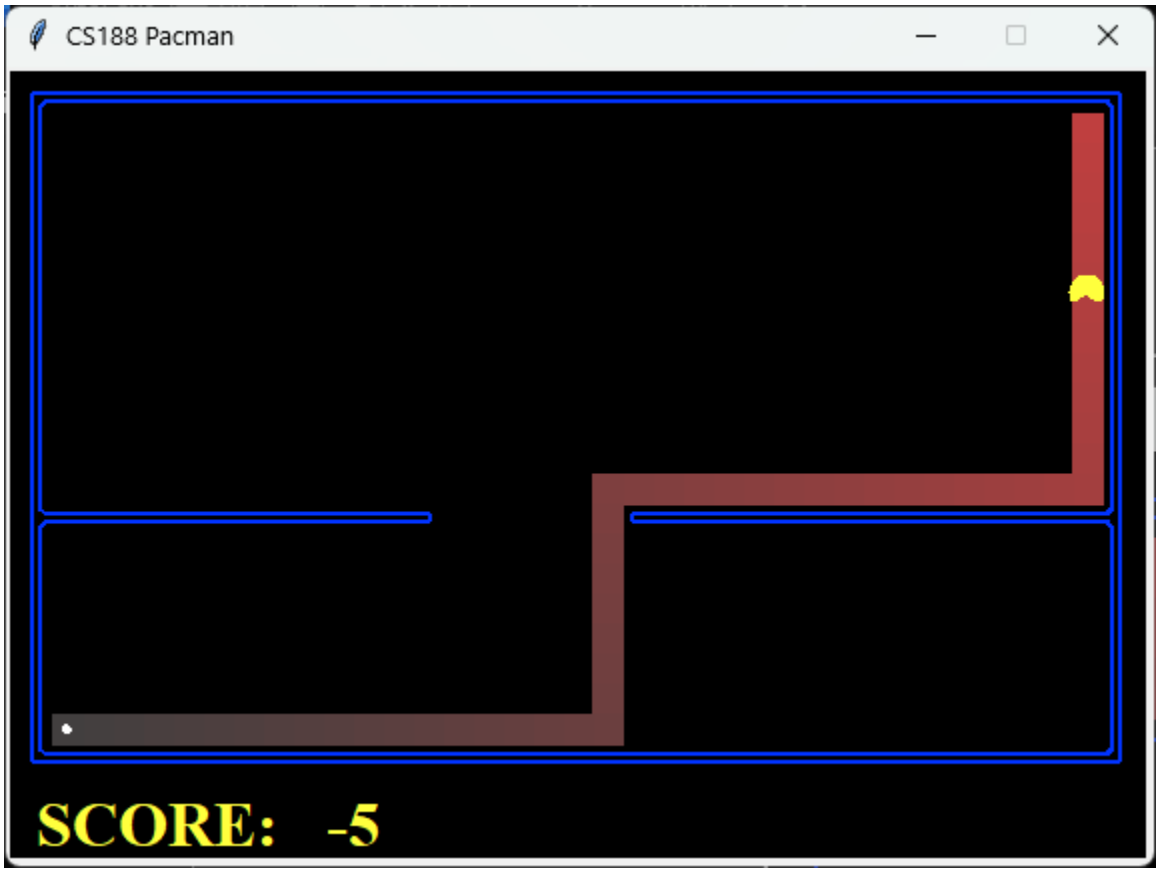
Using the Euclidian heuristic this algorithm was able to find the optimal solution to the problem. It also did not explore the right lower block. It also seemed to be better than the Euclidian heuristic overall but still ending up with the same score.

Hill Climb Euclidean and Manhattan GUI



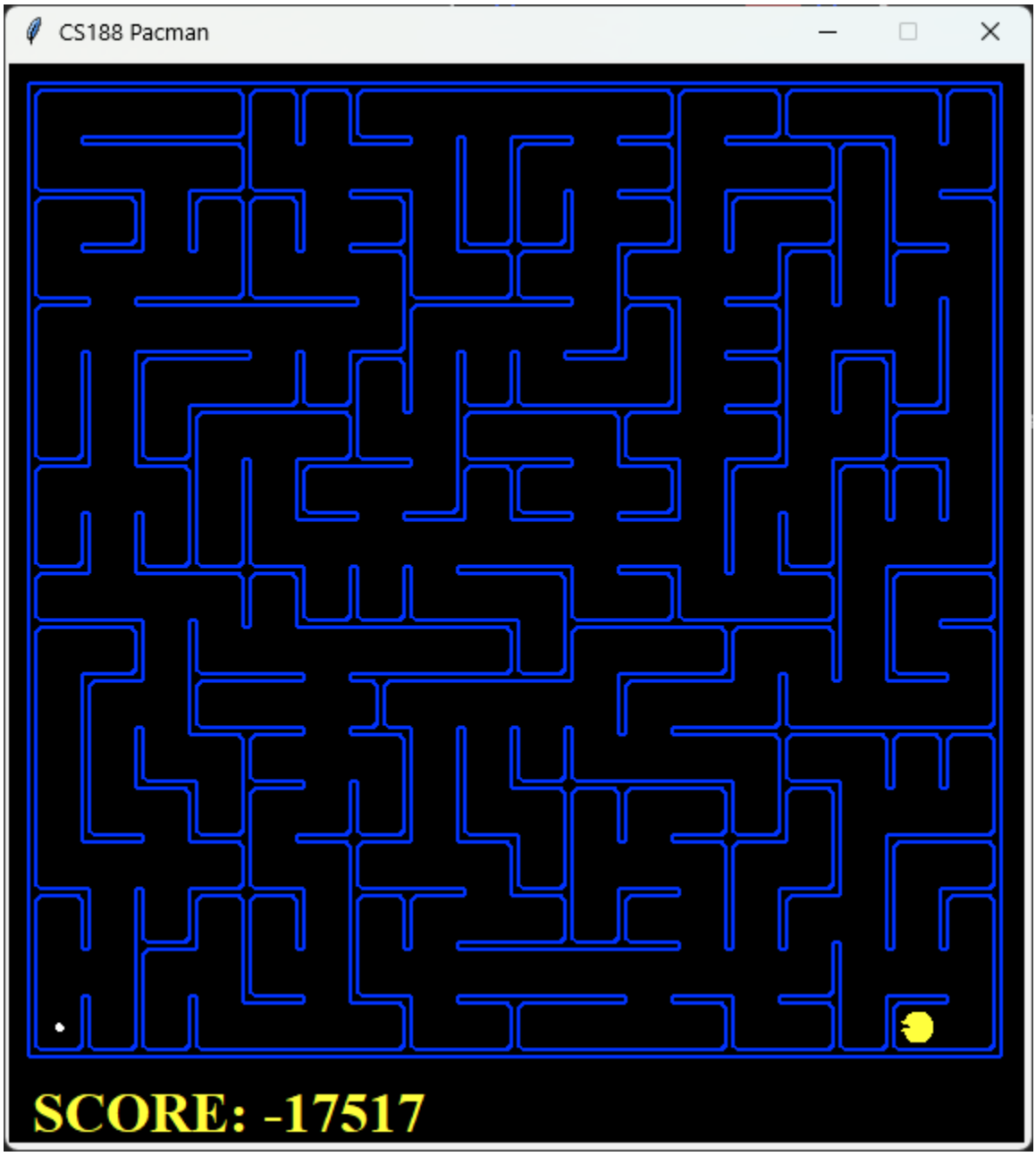
I could not get this algorithm to work on any other layout. It seemed to get stuck and never find the end. But this maze does not show the algorithms effectiveness.

HillClimb Manhatten and Euclidean GUI



This shows how effective this algorithm is at running when using this heuristic. It goes form point a to b while not exploring any other nodes. This is by far the best algorithm when trying to got from a to b.

HillClimb Error GUI



This one will forever hang on this point. I did not have time to implement the functionality to fully understand where the walls were. So it gets stuck on any wall in its path to success.