Originally, I was going to create the map by using bots randomly walking in different direction. However, I discovered a more efficient and better alternative algorithm called ‘Recursive Backtracking’.

**mapGeneration\_Botwalk.m** (The first map generation algorithm applied)

Rule:

X amount of bots can spawn and each has a limited amount of ‘Walks’

Bots will spawn somewhere near the middle of the array

Bots can walk left, right, up or down, but would not do so if it’s the edge of the array

Bots leave a white trail behind them, carving the path for the map and consuming ‘Walks’

The map is generated when all bots consume their ‘Walks’

**Output1: Output2: Output3:**

****

**Conclusion:** As shown, the ‘maps’ generated are not very pretty, It would not be fun to play with for a maze. A better algorithm had to be found

**Backtracking\_TemplateGenerator.m**

Before I began implementing the recursive backtrack algorithm, I must create a template where:

Walls are represented by black pixels (0), Paths are represented by white pixels (1), discovered Paths are represented by grey pixels (0.5), and every odd row/col is a wall

**Output 1 (30x30): Output 2 (51x51): Output 3 (101x51):**

****

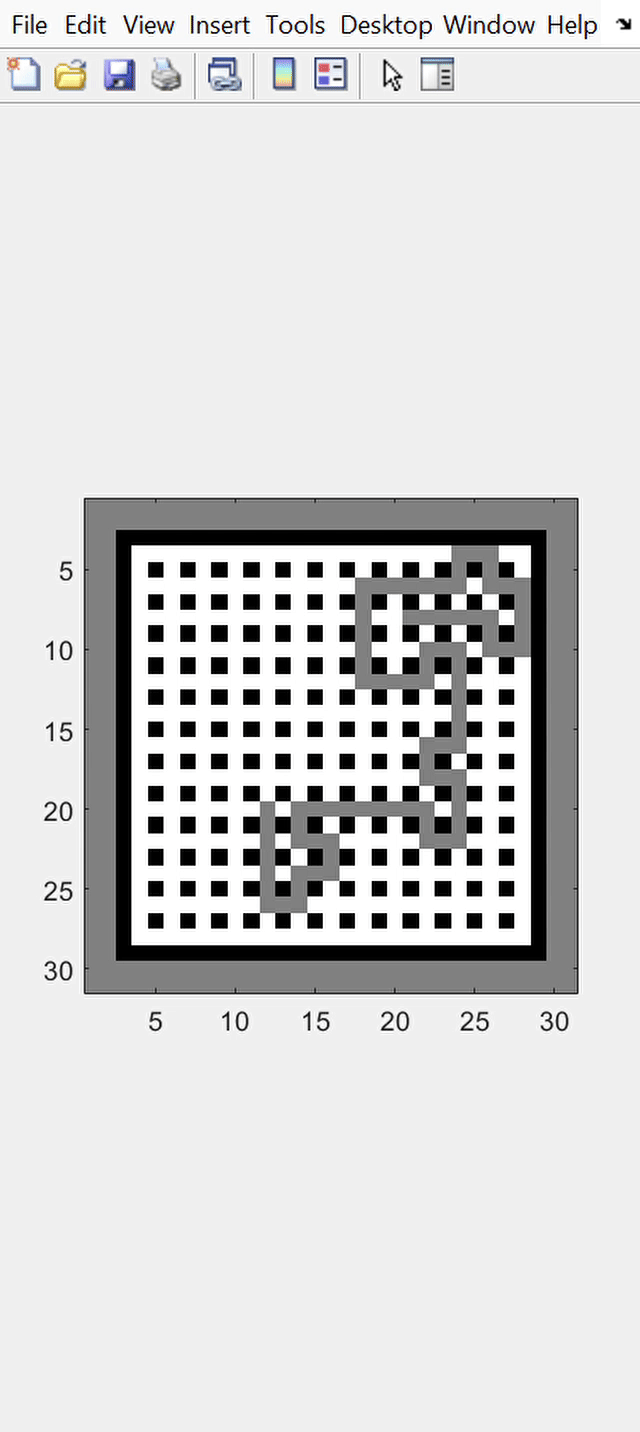
**Conclusion**: Code functions as expected, where every odd part of the template is a wall, and every even part is a path, surrounded by walls.

**backtrackingAlgorithm.m**

**Test #1 Basic test (31x31)**

**Input:** MAPROW = 31, MAPCOL = 31

**Output: Conclusion**

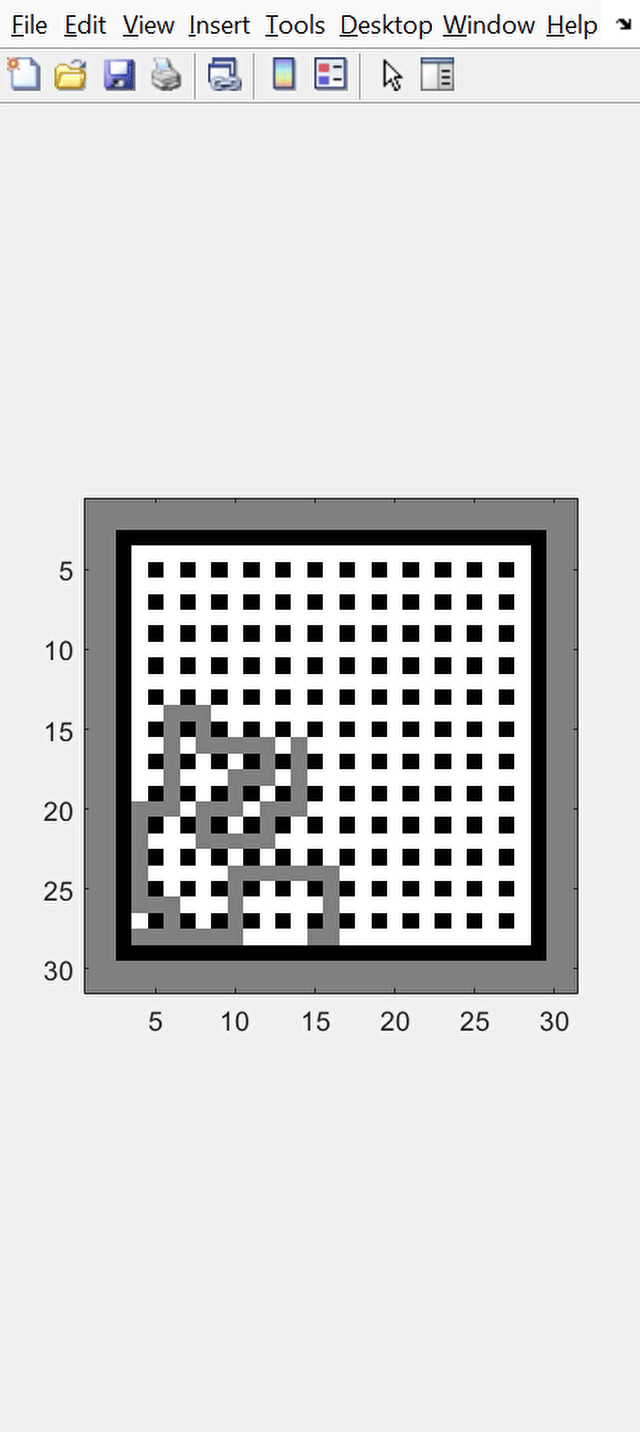


After discovering a path and then meeting a dead-end, the algorithm does not save its discovered position, thus reopening the same path to meet the same dead-end forever.

**Test #2 Basic test 2 (31x31)**

**Input:** MAPROW = 31, MAPCOL = 31

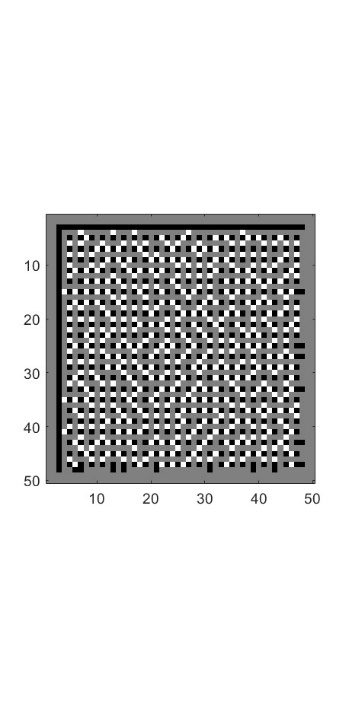
**Output: Conclusion**



Turning the maze array into a **Global variable** (global newMap) solved the issue, since the discovered path is now saved, and the algorithm does not repeat the same mistake. This is a hacky and short fix, so I decided not to create a new file for it.

**Test #3 random map Size (50x50)**

**Output: Conclusion:**

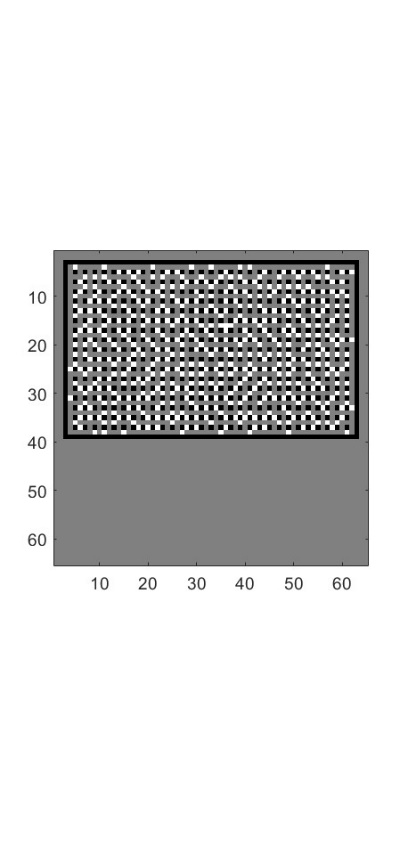
****

Using an even-numbered row or column does not generate a proper template, since one side is not the same as the other. This causes issue such as walls touching the border, making the algorithm be able to walk into the border walls. I have not found a fix yet, so it’s advised to use an **odd-numbered array**

**Test #4 different row and columns (41x65)**

**Input:** MAPROW = 40, MAPCOL = 65

**Output: Conclusion:**

****

The output of this was expected, all possible path discovered, and no walls were walked through.

**playerFlagSpawn\_math.m**

**Expected output:** I’m expecting white pixels on each quadrant that is not too close to the border and far from the middle.

**Input:** No input needed

**Output 1: Output 2: Output 3:**

****

**Conclusion:** Overall, all 3 outputs are providing similar results and meets the expected output, so this has been a success