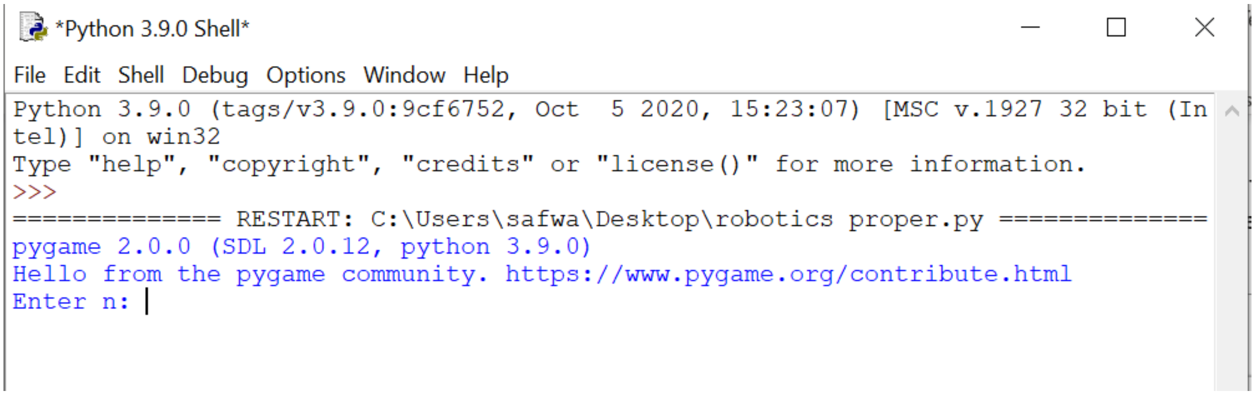
Safwan Hasan

This readMe file will explain the process and limitations of this program.

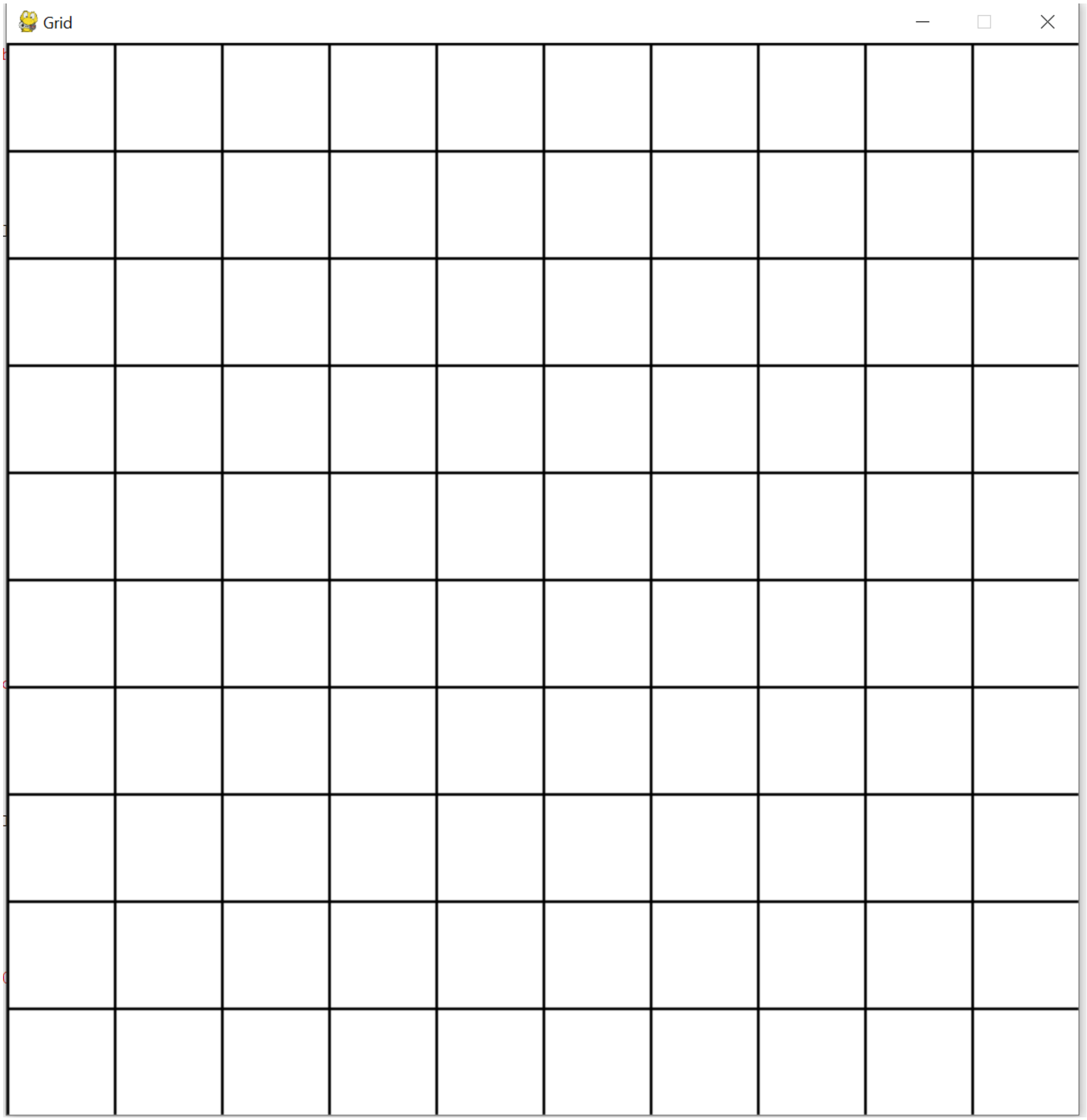
At first, a grid was implemented by taking a user input that determines the size of the grid. In other words, in a nxn grid, “n” was the input. This would then be used to draw lines on the screen pygame.draw.line() function. The following was the result:



```
Python 3.9.0 (tags/v3.9.0:9cf6752, Oct  5 2020, 15:23:07) [MSC v.1927 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============== RESTART: C:\Users\safwa\Desktop\robotics proper.py ==============
pygame 2.0.0 (SDL 2.0.12, python 3.9.0)
Hello from the pygame community. https://www.pygame.org/contribute.html
Enter n: 
```

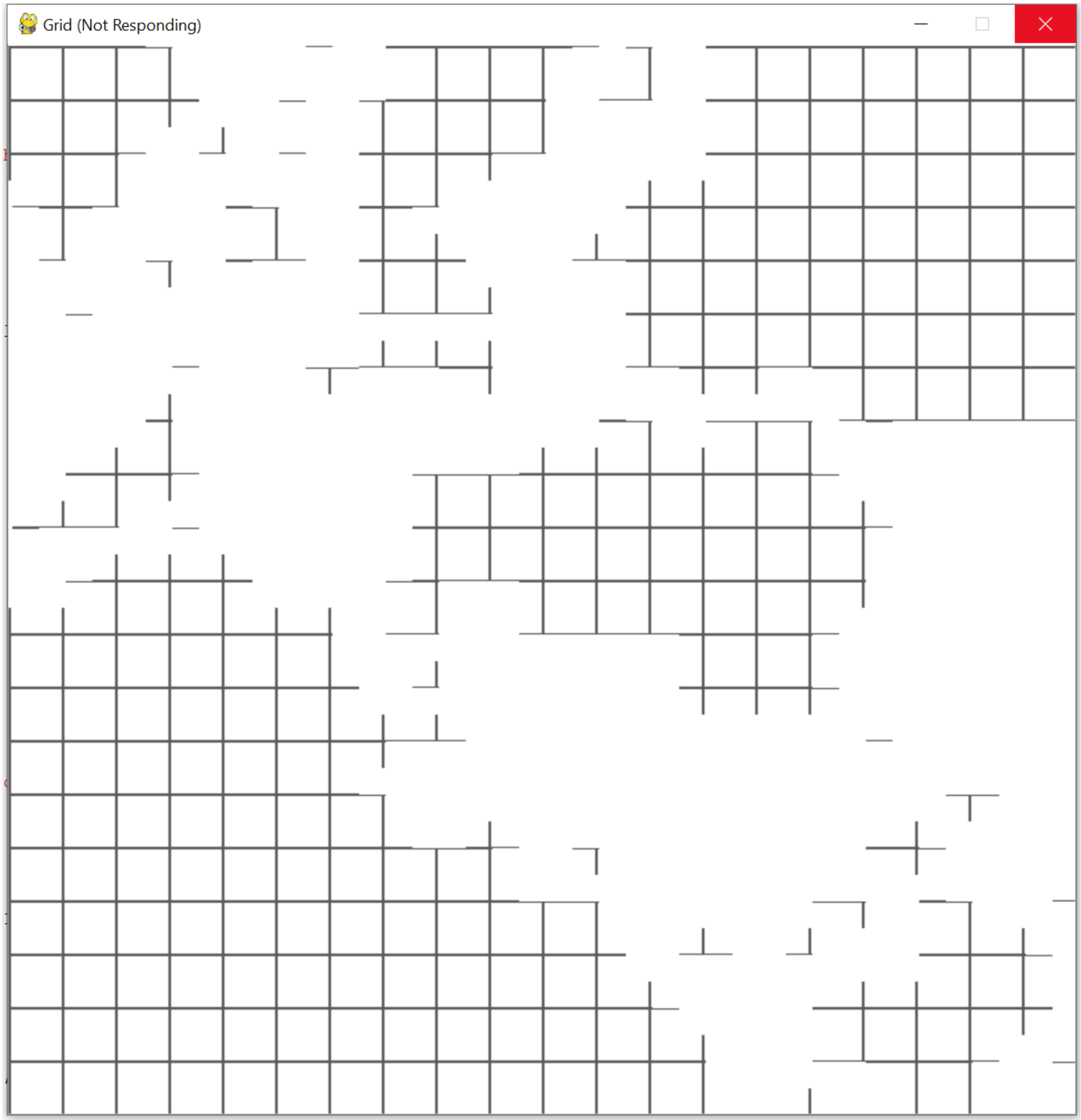
As this is a simple code, the input was written in the shell screen.

After inputting the number 10, a 10x10 grid was implemented:



Next, the maze was implemented.

It was created by selecting a random location on the screen and drawing white rectangles in a random direction. If these directions have already been, it will not be considered. This provided the following output:



Note: I would advise you to a high “n” value to provide a better-looking maze.

Limitations:

- The pygame screen starts not respond after brief inactivity (when not inputting an n value) and after the maze algorithm has run. It is unsure why this occurs at the moment.

References:

Provided lots of help in understanding maze algorithm:
https://github.com/tonypdavis/PythonMazeGenerator