Design

To make the program as user friendly as possible a window should be created to display the current map layout to the user. I have decided to use the pygame module instead to tkinter as it makes re-drawing the map when it changes easier.

Modules used: pygame, random, os, sys, time, and threading

# Creating a graphical display

The pygame module allows for the inputs and outputs to be graphical and interactable using the mouse instead of being purely text based. Before outputting to the canvas it has to be initialised using pygame.init(). This however only initialises the pygame module which is not yet something that can be used to display text or graphics. Using the variable g as the canvas g=pygame.display.set\_mode([width,height]). width and height are variables that have been set to 1600 and 900 and will be the amount of pixels inside the window. Things can now be displayed on the canvas g in the pygame window

As the generated map will likely be larger than what can be displayed onscreen there will have to be a way of moving or changing the area that is being displayed to the user. To make it easier to navigate a mini-map will be included in the top left corner.

# Generating a map

The map is generated by two for loops, inside the rows loop is another for the columns; inside the column loop random.choice(bias) is used to choose between three values in the list bias containing [0,1,1]. Once a number is chosen it is appended to the list column *n* times because of the for loop, for each completed cycle the column list is appended to the row list and cleared so that it can be appended to again.

# Editing a map

Instead of appending to the end of lists to

# Solving the map

Each map has a start and end point for the algorithm to solve, it will have to navigate between walls and obstacles to get there.