Map descriptor format

Occupancy grid representation
A map of the environment can be represented as an occupancy grid, where each cell of an evenly spaced grid stores information about
1) whether the corresponding environmental space has been explored (whether through remote sensing or by actually entering the square), and
2) what the content of that cell is (free space, obstacle type, etc).

Assumptions
We can enforce the following simplifying assumptions:
1) environments are represented using a 15x20 grid (150x200cm arenas using 10x10cm grid cells)
2) all grids are fully surrounded by exterior walls which do not need to be explicitly represented
3) grid cells can have only two content types: "free space" or "obstacle"

Format overview
Efficient storage of the occupancy grid is useful for storage and transmission of data. In particular, the choice of an efficient coding mechanism allows for a shortened string to be used as a representation of the map. A typical occupancy grid will contain several cells that have not been explored. We first store an index that encodes whether each cell has been explored, followed by the actual information about the contents of explored cells. In typical use cases, this will result in space savings.

Note that in all cases, we position the arena with the START squares at the bottom left and the GOAL squares at the top right. Traversal of the grid proceeds row by row beginning from the bottom row. Within each row, traversal of the grid proceeds from left to right.

Format specifications

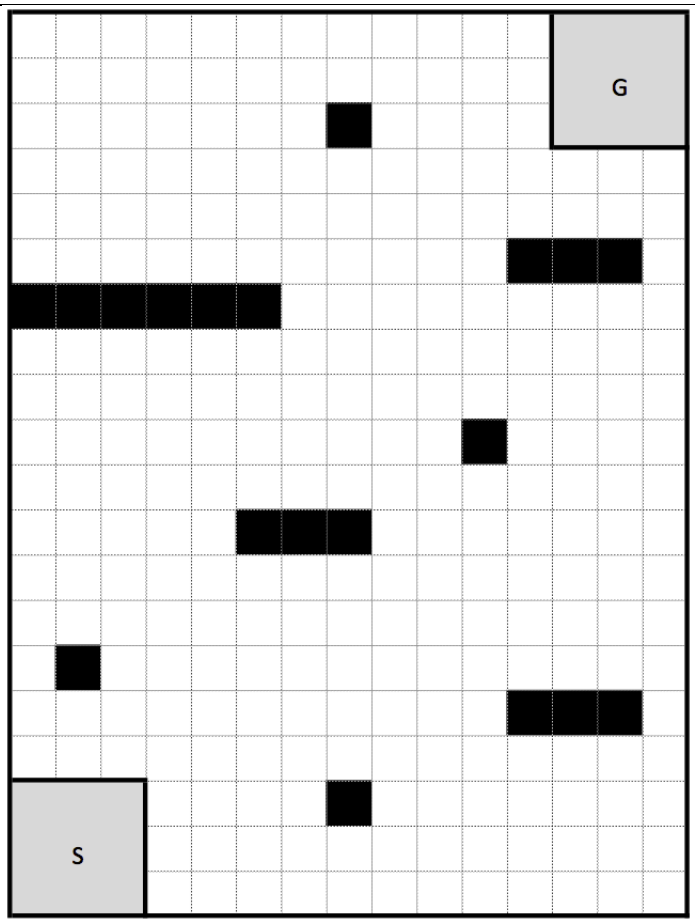
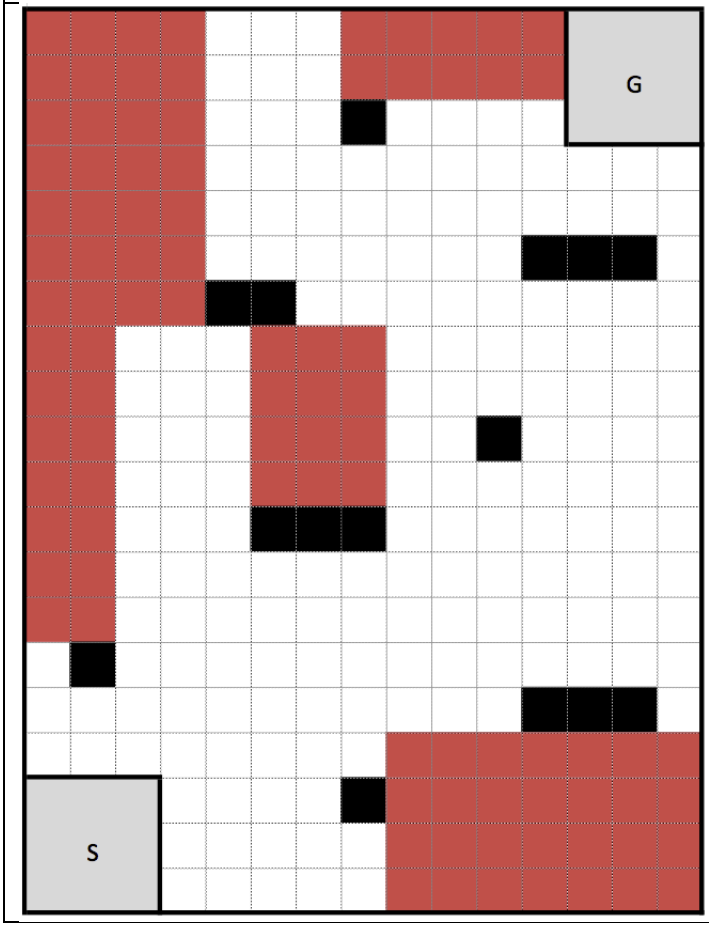
Part 1 : The "explored/unexplored" state of a cell is represented by 1 bit. For a 15x20 grid, this will occupy 300 bits. Use the value 0 to represent a cell that is unexplored/unknown, and the value 1 to represent a cell whose state is known/has been explored.

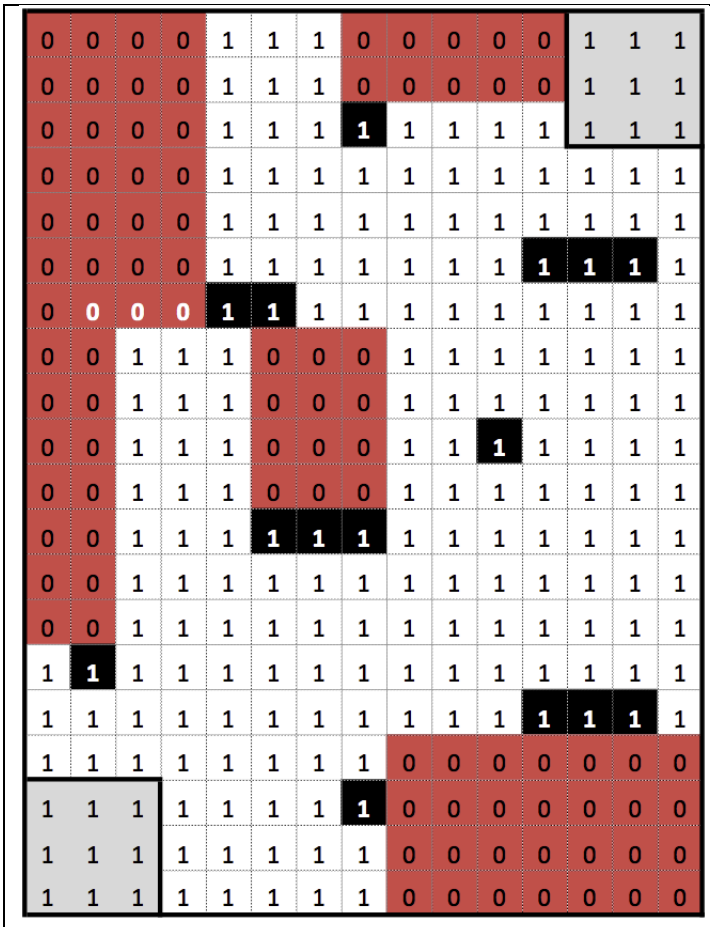
Set the first two and last two bits of the entire sequence to 11; this will pad the entire bit sequence to a total of 38 bytes.

Part 2 : Only cells which were previously marked as "explored" in Part 1 will be represented here. Use the value 0 to represent a cell that is empty, and the value 1 to represent a cell that contains an obstacle. Use '0' values to pad your bitstream up to full byte lengths (multiples of 8 bits).

Parts 1 and 2 should be displayed separately as hexadecimal characters.

A sample arena layout and the corresponding worked steps are displayed below for reference.

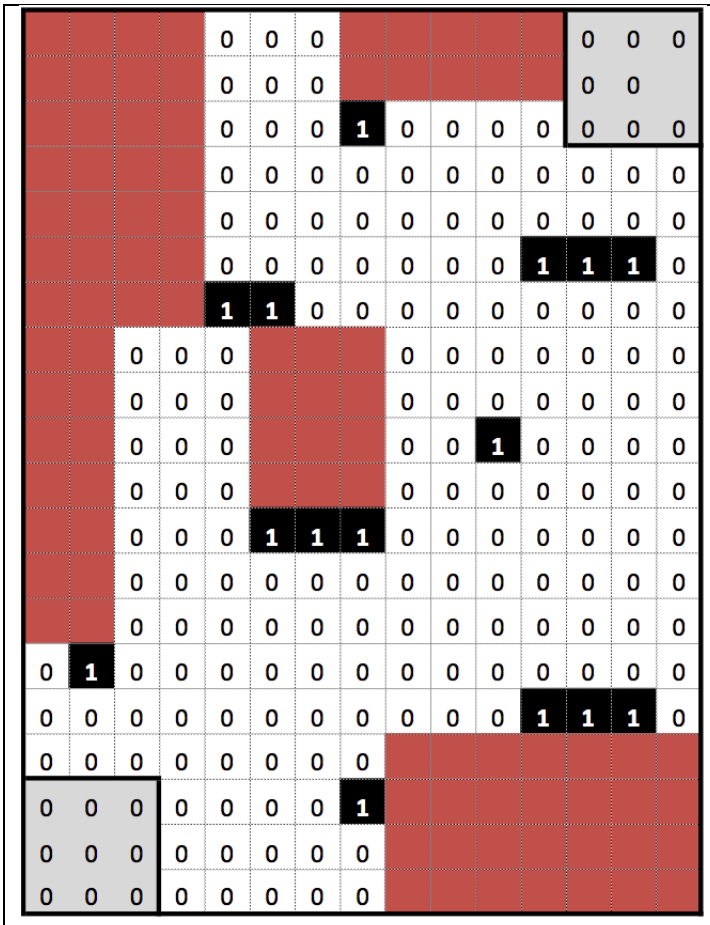
	This is a sample arena layout
	In this scenario, after the exploration phase has been terminated, the grid cells marked red are unexplored (their contents are not known) while the rest have been explored.



Part 1: The index is constructed by marking each unexplored square as 0 and each explored square as 1. The resulting bitstream is shown below, broken into rows for clarity in this example. When converting to hexadecimal for display, the entire bit sequence should be treated as contiguous. Note the padding sequences of '11' at the beginning and end.

11
111111110000000
111111110000000
111111110000000
111111110000000
111111111111111
111111111111111
001111111111111
001111111111111
001111111111111
001110001111111
001110001111111
001110001111111
001110001111111
000011111111111
000011111111111
000011111111111
000011111111111
000011111111111
000011100000111
000011100000111
11

The first 6 digits of the resulting hexadecimal sequence are:
FFC07F



Part 2: Only grid cells that were marked as “explored” in Part 1 are represented here by a bit. Cells that are known to be empty space are marked with a 0, while cells that are known to contain an obstacle are marked with a 1. In this example, there are 208 known cells represented, so no padding of the bit stream is required.

00000000
00000000
00000001
00000000
0000000000001110
0100000000000000
0000000000000
0000000000000
0001110000000
0000000000
0000010000
0000000000
0000000000
11000000000
00000001110
00000000000
00000000000
00010000000
000000
000000

The first 6 digits of the resulting hexadecimal sequence are:
000001