

**Course Report**

**Graph search algorithms**

**Course Name: Artificial intelligence**

**Name: Norbert Posan**

**Student Number: 20175243015**

**Introduction**

I developed a graph search based small animator. It is able to show how the 3 different algorithm works step-by-step. The developed code is available on my Github. https://github.com/norbinto/NUIST\_AI

**Configuration**

The game is configurable with the MainWindow.xaml.cs file:

public static bool IS\_PRINT\_DETAILED\_SOLUTION = false;

public static bool IS\_PRINT\_SOLUTION = true;

public static int TIME\_BETWEEN\_ITERATIONS = 160;

/// <summary>

/// 0=DFS, 1=BFS, 2=A star

/// </summary>

public static int ALGORITHM\_CHOOSER = 2;

The IS\_PRINT\_DETAILED\_SOLUTION configure the debugger command line detailed printing. It writes all the important information to the command line.

The IS\_PRINT\_SOLUTION configures the debugger command line printed solution and on the GUI will be printed with green too if it is true.

The TIME\_BETWEEN\_ITERATIONS is the time between 2 node operation in milliseconds.

The ALGORITHM\_CHOOSER is the way how can the user choose the algorithm.

In the Map.cs file the user can modify or update the maze.

0 - means wall

1 – ground

2 – start position

3 – goal position

The colors on the map

RED – wall, not able to walk across

GREY – ground, the place to move

GREEN – start position

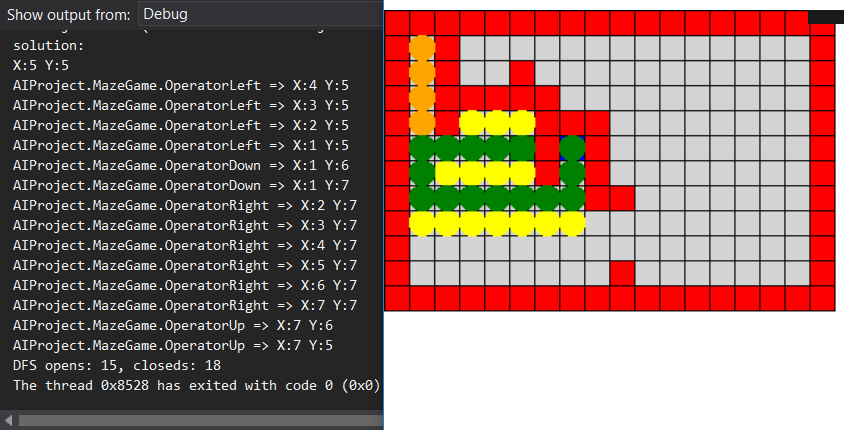
BLUE – goal position

DARKGREEN - solution

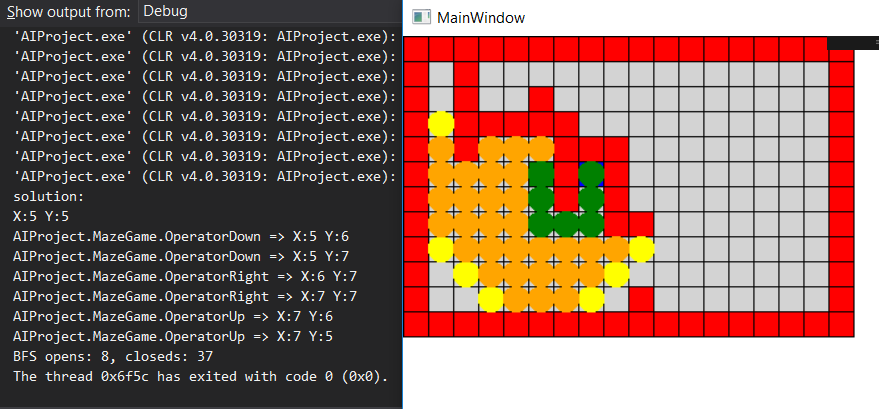
YELLOW – opened node (child nodes are not computed and not checked, is it goal or not)

ORANGE – closed node (checked it is goal or not, some or all of the children nodes are computed)

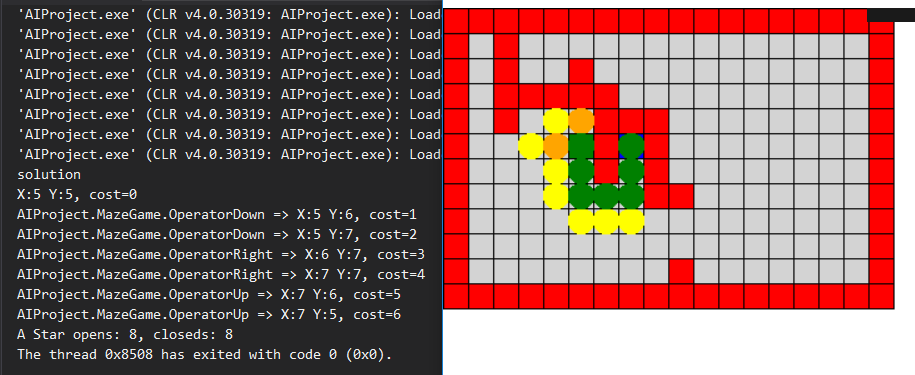
**Experiment**



The Deep First Search finished the maze with 15 opened and 18 closed nodes. Together it needs 33 nodes to find a solution



The Breadth First Search finished the maze with 8 open nodes and 37 closed nodes. Together it needs 45 nodes to find a solution



The A\* algorithm needed 8 opened and 8 closed nodes to find a solution in the maze. It is 16 together what is the best performance between these algorithms.

The A\* used a cost function. Every step costed the same, constant 1.

The A\* used a heuristic function. The heuristic function was an Eukelade distance between the actual state and the goal state.

**Code**

The code is available on Github. https://github.com/norbinto/NUIST\_AI