

Agent Ai Project Report

by Fraser Sweetman

FILE	3850_FRASER_SWEETMAN_AGENT_AI_PROJECT_REPORT_110679_15 30432961.DOCX		
TIME SUBMITTED	29-MAR-2018 11:23PM (UTC+0100)	WORD COUNT	858
SUBMISSION ID	85844998	CHARACTER COUNT	4044

School of Applied Computing ASSIGNMENT SPECIFICATION

Student name:	Fraser Sweetman	Student P number:	P148687
Programme:	<u>BSc/HND Computer Games Development; Computer Graphics and Visualisation;</u>		
Lecturer:	Gordon Dickers	Internal Verifier:	Andrew Baker
Assignment Title:	Agent AI: Search algorithms		
Hand Out Date:	5/2/2018	Submission deadline:	29/3/2018

Referencing:	In the main body of your submission you must give credit to authors on whose research your work is based. Append to your submission a reference list that indicates the books, articles, etc. that you have read or quoted in order to complete this assignment (e.g. for books: surname of author and initials, year of publication, <u>title of book</u> , edition, publisher: place of publication).		
Disclosure:	I declare that this assignment is all my own work and that I have acknowledged all materials used from the published or unpublished works of other people. All references have been duly cited.		
Student's Signature:	<i>Hardcopy assignments will not be accepted without a signature here</i> FSweetman	Date:	29/03/2018
<u>Late submission will result in a late penalty mark, as follows:</u> Up to one week late, maximum mark of 40% for first attempts. No late submission for resit work. (Academic Quality Handbook 2016/17 7.5 (5)) Only the Extenuating Circumstances Committee may grant an extension.			

Learning Outcomes tested (from module syllabus)
1. Demonstrate a sound knowledge and understanding of essential Game AI concepts and principles.
2. Analyse practical AI problems associated with games program development in order to synthesise appropriate software solutions.

Contents

Introduction	Page 3
How to Run Programme	Page 3
How it works	Page 3
References	Page 4
Bibliography	Page 4

Introduction

The objective of the programme is to use an AI to find the shortest path through a maze. The maze must have a start point and a finish point and can have obstacles such as a key to unlock a door that the AI has to collect and walls/barriers for the AI to navigate around.

How to run the programme

Load the programme on using visual studio 2015 and run it by either pressing ctrl and f5 or by going to the Debug tab then selecting Start Without Debugging. If you wish to edit the maze open the maze1.maz file in Notepad++ make the desired changes and save the file or save the file as a new .maz format file and change the file the programme reads the maze from in the visual studio programme file.

How it works

[1]This AI programme uses a maze.maz file to store the shape of the maze as well as the start and finish point and any obstacles the programme first try's to open the file and if unsuccessful will display an error message. Stating that it could not open the file. To solve this, make sure the programme s looking for the correct file (case sensitive) and that the file is in the folder inside the project along with the main.cpp file.

After the programme opens the file and gets the maze layout it will try display the maze in a window and will try to find the start position x and y coordinates. Next it will try to locate the finish point. The programme is then designed to display the location of the start and finish points x and y coordinates as text.

References

[1] file read with starter position locator <https://moodle.uwtsd.ac.uk/course/view.php?id=2266>

Bibliography

[1] Path Planning – A*(A-Star) <https://www.youtube.com/watch?v=icZj67PTFhc>

Appenix

A.

Main.cpp code for project

```
// reading a text file
#define WIDTH 7
#define HEIGHT 5
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main(void) {
    char maze[WIDTH*HEIGHT];
    string line;
    ifstream myfile("maze1.maz");
    int starty;
    int startx;
    int finishy;
    int finishx;
    //check to see if it can load the maze file print statement if it can't
    if (myfile.is_open())
    {
        int count = 0;
        while (getline(myfile, line))
        {
            cout << line << endl;
            line.copy(&maze[WIDTH*count], WIDTH, 0);
            count++;
        }
        myfile.close();
    }

    else cout << "Unable to open file";
    cout << endl << endl << endl;
    for (int y = 0; y < HEIGHT; y++)
    {
        for (int x = 0; x < WIDTH; x++)
        {
            //if (maze[WIDTH*y + x] == 's')
            cout << maze[WIDTH*y + x];
        }
        cout << endl;
    }

    for (int y = 0; y < HEIGHT; y++)
    {
        for (int x = 0; x < WIDTH; x++)
        {
            if (maze[WIDTH*y + x] == 's')
            {
                startx = x; starty = y;
            }
        }
    }
}
```

```

        } //cout << "start located at position:(" << x << "," << y << "," << ")
of maze" << endl;
        //cout << maze[WIDTH*y + x];
    }

}

//finding the finish location// works if in bracet abloce bt cause star
location finder to break
for (int y = 0; y < HEIGHT; y++)
{
    for (int x = 0; x < WIDTH; x++)
    {
        if (maze[WIDTH*y + x] == 'f')
        {
            finishx = x;  finishy = y;
        }
    }
    cout << "start located at position:(" << startx << "," << starty << "," << ") of maze"
    << endl;
    cout << "The Finish is located at position:(" << finishx << "," << finishy << "," <<
    ") of maze" << endl;
    return 0;
}
}

```

B. Maze File

```

*****
S      *
***   *
*  f  *
*****

```

Agent Ai Project Report

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

Instructor

31 /100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

GRADING FORM: SMU GENERIC_GJ

FRASER SWEETMAN

30.50

RESEARCH-INFORMED LITERATURE



Some references included but do try to include good quality refs/ established texts - eg Norvig and Russel. Avoid overusing electronic refs. No use of dfs or related algorithms and data structures in your implementation.

2

KNOWLEDGE AND UNDERSTANDING



Some understanding of C/C++ code is

demonstrated - eg maze is loaded and can be displayed but nothing on search/dfs & requisite data structures or shortest path.

4

ANALYSIS



Apart from the loading of a maze file and the printing of this onto the console output device, there is no evidence of further analysis of the problem specification nor of any interpretation of the search requirement aspect of the assignment.

4.5

PRACTICAL APPLICATION AND DEP.



Some basic file loading of a maze text file and the displaying of this but nothing further. No dfs or similar search implemented.

18

SKILLS FOR PROFESSIONAL PRAC.



A brief report with some references - see the research-informed literature feedback section above. Some Essential SE practices in evidence in code - comments, but no real structured approach or attempt to separate interface from implementation.

2