

Project report

AI used in game



February 28, 2016

Yuntian liu

C00151315

Contents

[Introduction 2](#_Toc444765860)

[Artificial intelligence 2](#_Toc444765861)

[AI technology in game 2](#_Toc444765862)

[Maze game 4](#_Toc444765863)

[Technology use 4](#_Toc444765864)

[Algorithms use 4](#_Toc444765865)

[What I choose 6](#_Toc444765866)

[Conclusion 6](#_Toc444765867)

[What I achieved 6](#_Toc444765868)

[What not achieved 6](#_Toc444765869)

[What I learnt 7](#_Toc444765870)

[References 7](#_Toc444765871)

# Introduction

This is a computer science based research project, the aim of this project is to discuss what is artificial intelligence and how it works in a game.

# Artificial intelligence

What is “Artificial intelligence”?

Artificial intelligence (AI) is the [intelligence](https://en.wikipedia.org/wiki/Intelligence) exhibited by machines or software. It is also the name of the academic field of study which studies how to create computers and computer software that are capable of intelligent behavior. It is a branch of computer sicence and it contains robot, language recognition, image recognition, natural language processing, expert system etc. Technology products developed by artificial intelligence will be the “container of knowledge” in the future.

Artificial intelligence is not human intelligence, but it can think like a human, and may exceeded human intelligence.

# AI technology in game

Normally, there are 6 AI technologies used in game:

1. Finite- state machine

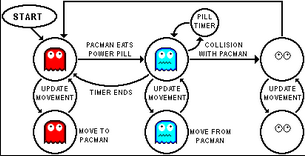
FSM is a mathematical [model of computation](https://en.wikipedia.org/wiki/Model_of_computation) used to design both [computer programs](https://en.wikipedia.org/wiki/Computer_programs) and [sequential logic](https://en.wikipedia.org/wiki/Sequential_logic) circuits. It is the base of managing a game world. It can simulate NPC’s emotional state, analyze input from players, or manage object’s state. Figure 1 is an Finite- state machine example In a pacman game.

Figure 1

1. Scripting language

A scripting or script language is a [programming language](https://en.wikipedia.org/wiki/Programming_language) that supports scripts, programs written for a special [run-time environment](https://en.wikipedia.org/wiki/Run-time_environment) that [automate](https://en.wikipedia.org/wiki/Automate) (Eva, 2011)  the [execution](https://en.wikipedia.org/wiki/Execution_(computing)) of tasks that could alternatively be executed one-by-one by a human operator. A scripting language usually used to control AI pattern in a game.

Scripting languages: JavaScripte, VBScript, Perl, (Sheppard, 2000) PHP, Python, (Wall, 2007) Ruby, Lua.

1. Fuzzy logic

Fuzzy logic is a form of [many-valued logic](https://en.wikipedia.org/wiki/Many-valued_logic) in which the [truth values](https://en.wikipedia.org/wiki/Truth_value) of variables may be any real number between 0 and 1, considered to be "fuzzy". By contrast, in [Boolean logic](https://en.wikipedia.org/wiki/Boolean_algebra), the truth values of variables may only be 0 or 1, often called "crisp" values. Fuzzy logic has been extended to handle the concept of partial truth, where the truth value may range between completely true and completely false. (Novák)

Usually we use “if/then” rules, for example:

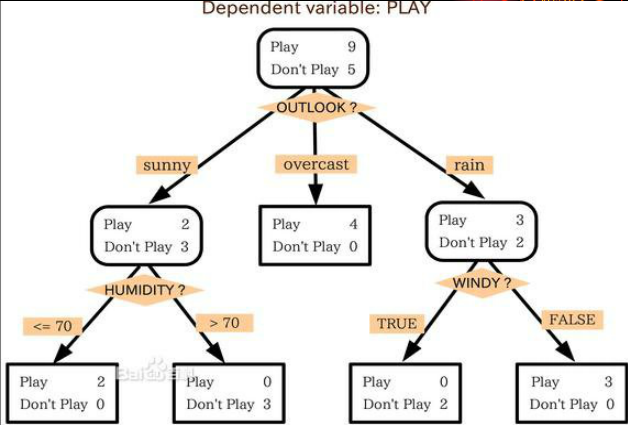
IF temperature IS very cold THEN stop fan

IF temperature IS cold THEN fan speed is zero

IF temperature IS warm THEN fan speed is moderate

IF temperature IS hot THEN fan speed is high

1. Decision tree

Decision trees are commonly used in [operations research](https://en.wikipedia.org/wiki/Operations_research), specifically in[decision analysis](https://en.wikipedia.org/wiki/Decision_analysis), to help identify a strategy most likely to reach a [goal](https://en.wikipedia.org/wiki/Goal).

1. Artificial neural networks

ANNs are a family of models inspired by [biological neural networks](https://en.wikipedia.org/wiki/Biological_neural_network) (the [central nervous systems](https://en.wikipedia.org/wiki/Central_nervous_system) of animals, in particular the [brain](https://en.wikipedia.org/wiki/Brain)) and are used to estimate or [approximate](https://en.wikipedia.org/wiki/Universal_approximation_theorem) [functions](https://en.wikipedia.org/wiki/Function_(mathematics)) that can depend on a large number of [inputs](https://en.wikipedia.org/wiki/Argument_of_a_function) and are generally unknown. Artificial neural networks are generally presented as systems of interconnected "[neurons](https://en.wikipedia.org/wiki/Artificial_neuron)" which exchange messages between each other. The connections have numeric weights that can be tuned based on experience, making neural nets adaptive to inputs and capable of learning. (Neural Networks, 2016)

1. Genetic algorithm

In the field of [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence), a genetic algorithm (GA) is a [search](https://en.wikipedia.org/wiki/Search_algorithm) [heuristic](https://en.wikipedia.org/wiki/Heuristic_(computer_science)) that mimics the process of [natural selection](https://en.wikipedia.org/wiki/Natural_selection). This heuristic (also sometimes called a [metaheuristic](https://en.wikipedia.org/wiki/Metaheuristic)) is routinely used to generate useful solutions to [optimization](https://en.wikipedia.org/wiki/Optimization_(mathematics)) and [search problems](https://en.wikipedia.org/wiki/Search_algorithm). (Mitchell, 1994) Genetic algorithms belong to the larger class of [evolutionary algorithms](https://en.wikipedia.org/wiki/Evolutionary_algorithm) (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as [inheritance](https://en.wikipedia.org/wiki/Heredity), [mutation](https://en.wikipedia.org/wiki/Mutation_(genetic_algorithm)), [selection](https://en.wikipedia.org/wiki/Selection_(genetic_algorithm)), and [crossover](https://en.wikipedia.org/wiki/Crossover_(genetic_algorithm)).

# Maze game

Now let’s talk about my project.

To study AI used in a game. I decided to develop a maze game using AI algorithms. First, I need to know what kind of technology I should use in this project.

## Technology use

In this project, it should has a window to display the maze. And in this year, we have already learnt how to use OpenGL. So I will use Visual studio as IDLE, and OpenGL library.

The most important part is to write an algorithm to generate a maze.

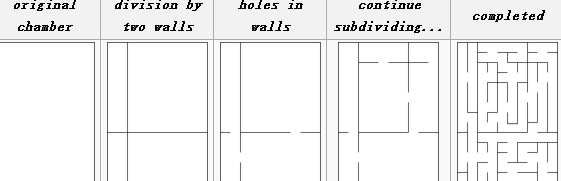
## Algorithms use

In general, the default rules are:

1. Maze map is a rectangular area;
2. Walls will take place in a maze map;
3. Walls and roads are separated.

There are many algorithms for Maze generation:

* Recursive division method

Mazes can be created with recursive division, an algorithm which works as follows: Begin with the maze's space with no walls. Call this a chamber. Divide the chamber with a randomly positioned wall (or multiple walls) where each wall contains a randomly positioned passage opening within it. Then recursively repeat the process on the subchambers until all chambers are minimum sized. This method results in mazes with long straight walls crossing their space, making it easier to see which areas to avoid.

* Kruska’s algorithm

[Kruskal’s algorithm](http://en.wikipedia.org/wiki/Kruskal's_algorithm) is a method for producing a minimal spanning tree from a weighted graph. The algorithm I’ll cover here is actually a randomized version of Kruskal’s; the original works something like this:

Throw all of the edges in the graph into a big burlap sack.

Pull out the edge with the lowest weight. If the edge connects two disjoint trees, join the trees. Otherwise, throw that edge away.

Repeat until there are no more edges left.

The randomized algorithm just changes the second step, so that instead of pulling out the edge with the lowest weight, you remove an edge from the bag at random. Making that change, the algorithm now produces a fairly convincing maze.

* Prim’s algorithm

1. Start with a grid full of walls.

2. Pick a cell, mark it as part of the maze. Add the walls of the cell to the wall list.

3. While there are walls in the list:

1. Pick a random wall from the list and a random direction. If the cell in that direction isn't in the maze yet:

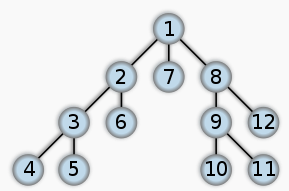
1. Make the wall a passage and mark the cell on the opposite 2. side as part of the maze.

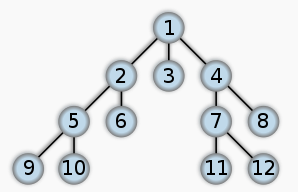
Add the neighboring walls of the cell to the wall list.

2. Remove the wall from the list.

It will usually be relatively easy to find the way to the starting cell, but hard to find the way anywhere else.

* Depth-first search and breadth first search

In this application, Frequently implemented with a stack, this approach is one of the simplest ways to generate a maze using a computer. Consider the space for a maze being a large grid of cells (like a large chess board), each cell starting with four walls. Starting from a random cell, the computer then selects a random neighbouring cell that has not yet been visited. The computer removes the wall between the two cells and marks the new cell as visited, and adds it to the stack to facilitate backtracking. The computer continues this process, with a cell that has no unvisited neighbours being considered a dead-end. When at a dead-end it backtracks through the path until it reaches a cell with an unvisited neighbour, continuing the path generation by visiting this new, unvisited cell (creating a new junction). This process continues until every cell has been visited, causing the computer to backtrack all the way back to the beginning cell. We can be sure every cell is visited. (Even, 2011)

The defining characteristic of breadth search (Graph500 list, 2015) is that, whenever BFS examines a maze cell c, it adds the neighbours of c to a set of cells which it will to examine later. In contrast to DFS, these cells are removed from this set in the order in which they

were visited; that is, BFS maintains a queue of cells which have been visited but not yet examined. Thus, a cell can have three states:

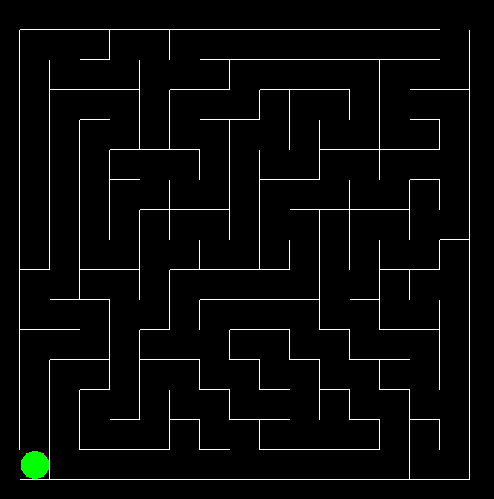
* Unvisited. The cell has not yet been visited by BFS.
* Visited but Not Examined. The cell has been discovered, but BFS has not evaluated whether its neighbours should be added to the Queue.
* Examined. The cell has been visited, and all its neighbours are/have been in the queue (ie, they are already "Visited but Not Examined" or "Examined").

## What I choose

Depth-first search and breadth search algorithm are the two of the easiest methods to generate maze. And we have already learnt these algorithm in year 3, and most of the maze game generated by depth first search method on the Internet, So I choose depth first search as my AI algorithm to generate a maze. It is conveniet for me to look up the material and learn code from others.

# Conclusion

## What I achieved

I used depth first search method and stack to create a random maze from level 1 to level 20. And I reset the cell size in each level make them become harder and harder. User can use W/A/S/D to move the green ball, or user can press F1 to find way automaticly.

## What not achieved

1. In this project, there are many algorithms to generate maze game, but I only used one of them. It is a regret that I did not try other algorithms.
2. Some documents are not completed, There are lot of contents missing in those documents. because I am not from English speaking country, most of my gramar of sentences are not standard. So I am not satisfied with these documents.

## What I learnt

I did a long research before Starting this project andI looked up the materials from the internet. During this five months. I got a lot experience of using openGL and C++.

This helped me have better understood of AI used in game. I learnt AI types during research, and many AI algorithms to generate a maze. This will helps me a lot in other AI applications.

I got so many experence from this project.

# References

Eva, O. (2011, 01 08). Retrieved from www.evaoceguera.com

Even, S. (2011). *Graph Algorithms.* Cambridge University Press, pp. 46–48, ISBN 978-0-521-73653-4.

*Graph500 list*. (2015, 11). Retrieved from Graph500: http://www.graph500.org/specifications#sec-5

Mitchell. (1994). *An Introduction to Genetic Algorithms. .*

*Neural Networks*. (2016, 02 28). Retrieved from dmoz: https://www.dmoz.org/Computers/Artificial\_Intelligence/Neural\_Networks

Novák, V. P. (n.d.). *Mathematical principles of fuzzy logic Dodrecht.* Kluwer Academic. ISBN 0-7923-8595-0.

Sheppard, D. (2000). *Beginner's Introduction to Perl.* dev.perl.org. Retrieved 2011-01-08.

Wall, L. (2007, 12 6). Retrieved from Programming is Hard, Let's Go Scripting..: http://www.perl.com/pub/2007/12/06/soto-11.html