Studio Project

Production Diary

IMMERSIVE STUDIOS

ASSESSMENT 1

Xihao Chen | 30053752

2022

Table of Contents

[Part 1 – Pre-Production 2](#_Toc106609484)

[DFS Pathfinding Algorithm 2](#_Toc106609485)

[Definitions 3](#_Toc106609486)

[AI Design 4](#_Toc106609487)

[AI Behaviour Flowchart 6](#_Toc106609488)

[AI Production Implications 7](#_Toc106609489)

[Feasibility 7](#_Toc106609490)

[Suitability 8](#_Toc106609491)

[3-Point Estimate 9](#_Toc106609492)

[Controller Device and Controls 10](#_Toc106609493)

[HCI/GUI Research 10](#_Toc106609494)

[Events for HCI Devices 10](#_Toc106609495)

[uGUI vs IMGUI 10](#_Toc106609496)

[Widget Case Studies 10](#_Toc106609497)

[Paper Prototypes 10](#_Toc106609498)

[Part 2 – Production 11](#_Toc106609499)

[AI Implementation 11](#_Toc106609500)

[Testing 11](#_Toc106609501)

[Windows / Web Prototype Builds 11](#_Toc106609502)

[Testing 11](#_Toc106609503)

[Part 3 – Gold Master 11](#_Toc106609504)

[Review and Evaluation 11](#_Toc106609505)

[Final Checks 11](#_Toc106609506)

[Required Modifications 11](#_Toc106609507)

[References 12](#_Toc106609508)

# Part 1 – Pre-Production

## DFS Pathfinding Algorithm

The **Depth First Search** (DFS) algorithm is an algorithm used for pathfinding purposes with tree or graph data structures. The DFS algorithm at a given node in a tree/graph structure explores the children of a parent node recursively (i.e. depth first), checking the child of the child. The recursive nature of the DFS algorithm continues to explore a path as far as possible before backtracking, repeating the process until the goal node is reached.

A benefit of the DFS algorithm is its suitability for searching every possible path in tree/graph structures with few possible paths. A DFS pathfinding algorithm also consumes less memory than other pathfinding algorithms (e.g. Breadth First Search), especially when finding a solution through large graphs.

However, the DFS algorithm is not guaranteed to find the shortest path in a given graph structure. The main shortcoming of DFS is it does not check all the siblings of a node, which may result in the algorithm taking a longer path (i.e. the number of nodes processed). The DFS pathfinding algorithm is also a poor choice for tree/graph structures with many paths as it will explore unnecessary paths.

In a video game context, DFS may be used to solve mazes, discovering a possible path/solution in the process. The algorithm may also influence which path to take through a decision-making tree. For example, a game of chess contains several branching outcomes after each move, and DFS can find several series of moves (known as lines in chess terminology) that may provide the player with a positive outcome.

DFS can influence the development of games that follow a tree structure leading to multiple outcomes. By following a particular sequence of events in the game, a player will reach either a successful or undesirable outcome. Should a player experience an event with an undesirable outcome, they are able to backtrack to a previous save point and try a different path. A DFS algorithm may help players find a gameplay path with a successful conclusion for the player to follow.

Minesweeper

## Definitions

|  |  |
| --- | --- |
| **Term** | **Definition in terms of the DFS algorithm** |
| **Pathfinding** | Pathfinding refers to the method used to find and traverse the shortest/most efficient route through a data structure via an algorithm.  In terms of the Depth First Search (DFS) algorithm, its pathfinding approach is to recursively explore all the children of a node in a path as far as possible, backtracking if necessary and repeating the process until a goal node is reached. |
| **Tree** | A tree is an undirected graph data structure where any 2 nodes are connected by exactly one path and has no cycles (e.g. a path that starts/ends at the same node). The nodes are organised in a hierarchal structure with the singular root node branching into two nodes.  A DFS algorithm traverses through a tree structure from the root node, which is a selected arbitrary node in a graph as the start point. The algorithm then explores a path from the root node as far as possible before backtracking. |
| **Parent** | A parent is a node that precedes any given node in a graph data structure. A parent node may be identified by its branch to another node.  Parent nodes are important for the functionality of the DFS algorithm as a visited parent node is considered the backtracking point for the algorithm. |
| **Child** | A child is a node that descends from any given node is a graph data structure. A child node may be identified by the link to its parent node.  The DFS algorithm recursively checks the children of child nodes down a path until it reaches a node with no children. |

## AI Design

According to the game brief supplied, the AI will follow the procedure outlined below:

* On game start, the AI will start by moving to the first node in the static list of nodes.
* The AI will then use a depth-first search algorithm to determine which node the player has moved to (target node).
* Whenever the AI reaches its current target node, it will use the DFS algorithm to find a new target node (the current node of the player).
* When the AI collides with the player, the end of the round is triggered, and the game is reset.

The AI will also use a trigger collider to determine if the player collided with it, ending the game session and restarting the game shortly afterwards.

With the project file containing a node graph structure, the AI will traverse through these nodes via a DFS pathfinding algorithm to achieve the above outcomes. The following pseudocode will be used to implement the pathfinding behaviour for the AI when tracing the player’s location:

|  |
| --- |
| IF targetNode NOT aiCurrentNode AND targetNode NOT null  currentNode = targetNode  ELSE IF playerTargetNode NOT null AND playerTargetNode NOT aiCurrentNode  currentNode = playerTargetNode  END IF  If currentNode NOT null  Set current direction towards node  Normalise current direction  END IF |

And ditto, for the DFS algorithm:

|  |
| --- |
| WHILE nodeStack > 0  FOR child in currentNode  IF a visited node has no child AND nodeStack has no child  IF currentNode = playerCurrentNode  RETURN child  Push child to nodeStack  END FOR  END WHILE  RETURN null |

The AI design will include a serialised private float variable “speed” that acts as a movement speed modifier (Default value 4). The serialised variable will provide additional flexibility when testing the AI implementation in the Unity engine as it makes the private variable visible in the Unity inspector. With serialisation, the “speed” variable can be set on the fly via the Unity Inspector menu instead of modifying the value from the AI’s script.

### AI Behaviour Flowchart

## AI Production Implications

### Feasibility

The feasibility assessment should determine whether there is a sufficient allocation of resources and technical capabilities to implement the AI into the Unity project.

|  |  |
| --- | --- |
| **Feasibility Question** | **Response** |
| **Would the proposed implementation work in practice?** | The implementation of the AI would work in practice; a part of the implementation was demonstrated during a class session. With a few tweaks and additions to the pathfinding code, the implementation should work smoothly. |
| **Has planning been done to support the strategy?** | The studio project requires some research to be documented in the production diary regarding the Depth First Search (DFS) algorithm, as well as some research into an AI strategy. A brief section of research with human-computer interaction (HCI) devices is also included in the production diary. |
| **Can the required resources be obtained and integrated?** | As the working file for the studio project was provided beforehand, the supporting resources are outlined in the “conditions” section of the studio project. Some of the essential resources include: Blackboard learning materials, online access, Unity game engine, an HCI device and the studio head for consultation purposes. |

### Suitability

#### SWOT Analysis

Does the strategy align with the external environment? For analysis of external forces, we focus on opportunities and threats.

The SWOT analysis (Strengths, weaknesses, opportunities and threats) examines the internal and external factors affecting the implementation of the AI. It may improve the suitability of the implementation by challenging assumptions and to uncover shortcomings with performance.

|  |  |  |  |
| --- | --- | --- | --- |
| **Strengths** | **Weaknesses** | **Opportunities** | **Threats** |
| The game brief provides a clear idea of how the AI will interact in the studio project. | The Depth First Search (DFS) pathfinding algorithm may not find the shortest path to the player. | Consulting with the studio head to gather feedback can help clarify any questions regarding the implementation. | The project is marked as unsatisfactory for not meeting the project/game brief requirements. |
| The DFS algorithm and other game brief components are already implemented. | The components may be incorrectly implemented, requiring further testing to rectify any issues. | Referring to the Collaborate recordings should provide a video reference for testing the game brief components. |

### 3-Point Estimate

You will also need to consider its potential impact on a theoretical production budget and timeline by estimating a potential and duration for the production of the NPC.

A 3-point estimate is detailed below:

|  |  |  |
| --- | --- | --- |
| **3-Point Estimate of Hours Required for Implementation** | | |
| **Optimistic Estimate (O)** | **Most Likely Estimate (M)** | **Pessimistic Estimate (P)** |
| **20** | **30** | **40** |
| **Beta Distribution:** | | |

## Controller Device and Controls

## HCI/GUI Research

### Events for HCI Devices

### uGUI vs IMGUI

## Widget Case Studies

## Paper Prototypes

# Part 2 – Production

## AI Implementation

### Testing

## Windows / Web Prototype Builds

### Testing

# Part 3 – Gold Master

## Review and Evaluation

### Final Checks

### Required Modifications

# References

GeeksforGeeks. (n.d.). *Depth First Search or DFS for a Graph*. Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/

happycoding. (n.d.). *Pathfinding: Depth-First Search*. Retrieved from happycoding.io: https://happycoding.io/tutorials/libgdx/pathfinding#depth-first-search

hurna. (n.d.). *Depth First Search (DFS)*. Retrieved from hurna.io: https://hurna.io/academy/algorithms/maze\_pathfinder/dfs.html

Wikipedia. (2022, May 13). *Depth-first search*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Depth-first\_search