**AT03 Production Diary**

**1.1.1 A\* Algorithm Research**

A\* is a pathfinding algorithm used in graph data structures, functioning similarly to Dijkstra’s algorithm. A\* uses an additional heuristic function compared to Dijkstra’s algorithm, sacrificing some pathfinding accuracy for computational efficiency. In a graph data structure, A\* calculates the cost of travel to all unvisited neighbouring nodes after entering a node via the F Cost function (with being a neighbouring node). A\* will move to the node with the lowest cost value returned from , updating the priority queue and iterating the process until the target node is reached.

The balance of optimality (Finding the best solution) and completeness (Guaranteeing whether the best solution was found after exhausting all possibilities) of the A\* pathfinding algorithm makes its implementation popular for pathfinding/graph traversal applications. However, as A\* is dependent on the accuracy of its heuristic method, the pathfinding algorithm may take longer to find a solution in a graph structure compared to other pathfinding algorithms such as Greedy.

In a video game context, the A\* pathfinding algorithm may be used for player/NPC movement where the goal is a single location, prioritising routes which are closer to the goal location to ensure they move through the best path. 2D games are one such platform for applying A\* pathfinding on players/NPCs (e.g. Command and Conquer).

**A\* Terminology Definitions**

**F Cost:**

The F Cost is a function used in the A\* pathfinding algorithm. The function is of the form , which determines the total estimated cost from the starting node to the destination node. When entering a node in a graph data structure, the A\* algorithm will calculate the travel cost for all neighbouring nodes with the F Cost function, moving to the node with the least cost after processing.

and are two parameters of the F Cost function:

* is the exact cost to move from the starting node to a specific node .
* is the heuristic method’s estimated cost from node to the destination node.

The value of relative to determines the speed and accuracy of the pathfinding process. With A\*, values that are higher than sacrifice accuracy for performance (Forfeiting a guaranteed shortest path) and vice versa for values . The ideal value for is to match , ensuring A\* follows the best path and has optimal performance.

**Heuristic:**

A heuristic technique is a problem-solving approach that finds an approximate solution when traditional methods are too time consuming/cannot find an exact solution.

In terms of the A\* pathfinding algorithm, the heuristic is represented as a function that estimates the cost to move from a node to the destination. An example of a heuristic technique in A\* is the Euclidean Distance heuristic, which calculates the cost between nodes in a straight-line via a mathematical formula. The heuristic is slightly more accurate than the Manhattan Distance heuristic (Which calculates the total of horizontal/vertical movements as squares to reach the target), but slower due to having a larger area to search.

The heuristic function in the A\* pathfinding algorithm must be admissible, i.e. the estimated cost to the destination node cannot overestimate the lowest possible exact cost to the destination node from the current node position.

**Priority Queue:**

A priority queue is a form of queue that assigns a priority value to each node in a graph data structure. The priority queue processes nodes based on their assigned priority, serving higher priority elements first.

The A\* pathfinding algorithm uses a priority queue that adds nodes it visits to the queue, adjusting the priority of nodes based on the F Cost function cost and removing visited nodes from the queue.

**1.2.1 Specifications and Software Analysis**

**3D Model Export Specifications:**

**Character Animation Specifications:**

**Animation Features, Toolsets, and Capabilities in Unity:**

**1.2.2 Unity Scene Navigation**

**Panning the viewport camera:**

**Zooming the viewport camera in/out:**

**Rotating the viewport camera:**

**1.2.3 Enhancing Workflow in Unity**

**Keyboard Hotkey/Shortcut Procedures:**

**Navigational Input Procedure:**

**1.3.1 HCI Device Integration Research**

**Potential HCI Devices:**

**Selected HCI Devices and Associated Control Schemes:**

**1.3.2 Planned HCI Integration**

**C# Event System Summary:**

**Unity GUI Library Reviews:**

**1.3.3 UI Widget Example Overviews**

**2.1.1 AI Behaviour Chart**

*Insert your behaviour chart for the AI here.*

**2.1.2 AI Design Review**

**Review of AI Design Feasibility:**

**AI Production Timeline and Cost Estimates:**

**2.2.1 Animation Reference Materials**

<https://www.roblox.com/bundles/356/Rthro-Animation-Package>

3D animations for multiple actions (Walk, Run, Idle, etc)

**2.3.1 UI Widget Paper Prototype**

*Insert the paper prototypes for the required UI widget here.*

**3.1.2/3.3.2 Testing Log**

*Please add rows as required.*

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case Description | Expected Results | Actual Results | Success? |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**3.2.2 Animation Troubleshooting**

**Error Scenario Summary:**

**Relevant Official Unity Documentation:**

**Relevant Unity User Feedback:**

**3.3.2 Web Browser Testing**

**Web Build Running in Mozilla Firefox:**

**Web Build Running in Google Chrome:**

**4.1 Required Amendments**

**4.2 Final Review**

**AI Evaluation:**

**Final Checks:**

|  |  |
| --- | --- |
| **Final Checks** | **Confirmed** |
| * Camera movement responds to relevant inputs |  |
| * Player movement responds to relevant inputs |  |
| * Player interaction responds to relevant inputs |  |
| * AI path finding (using the A\* algorithm) has been successfully integrated |  |
| * The AI behaviour is defined by four states – idle, chase, stun, and either patrol or wander. |  |
| * The logic of all AI behaviour states aligns with the game brief |  |
| * AI successfully and appropriately transitions between behaviour states |  |
| * Chase state becomes the default state for the AI when the objective item is interacted with |  |
| * AI transitions to stun state in response to player interaction |  |
| * AI 3-D model and animations have been successfully integrated for each behaviour state |  |
| * AI audio clips have been successfully integrated for each behaviour state |  |
| * Game win conditions have been successfully implemented |  |
| * Game loss conditions have been successfully implemented |  |
| * UI prompt displays appropriate message based on game loss or victory |  |
| * UI widget responds to relevant keyboard inputs |  |
| * UI widget responds to relevant mouse inputs |  |
| * UI widget responds to relevant controller inputs |  |
| * UI set to scale with a full HD resolution (1920x1080) |  |
| * UI objective text updates appropriately |  |
| * UI ‘new game’ button successfully loads game scene |  |
| * UI ‘info’ button successfully displays a window with some game information |  |
| * UI ‘quit’ button successfully closes the application |  |
| * Appropriately compatible with Windows |  |
| * Appropriately compatible with Google Chrome web browser |  |
| * Appropriately compatible with Mozilla Firefox web browser |  |

**4.3 Final Client Sign-Off**

*Insert a screenshot of your email communications with the client, providing evidence of their endorsement to finish the production of the project.*

**Repository Link:**

# References

GeeksforGeeks. (2022, February 6). *A\* Search Algorithm*. Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/a-search-algorithm/

Programiz. (n.d.). *Priority Queue*. Retrieved from Programiz: https://www.programiz.com/dsa/priority-queue

Red Blob Games. (2014, May 26). *Introduction to the A\* Algorithm*. Retrieved from Red Blob Games: https://www.redblobgames.com/pathfinding/a-star/introduction.html

Red Blob Games. (2017, October 19). *Heuristics*. Retrieved from Red Blob Games: https://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html

Roy, B. (2019, September 29). *A-Star (A\*) Search Algorithm*. Retrieved from Towards Data Science: https://towardsdatascience.com/a-star-a-search-algorithm-eb495fb156bb

Thaddeus Abiy, H. P. (n.d.). *A\* Search*. Retrieved from Brilliant: https://brilliant.org/wiki/a-star-search/

Wikipedia. (2022, February 23). *A\* search algorithm*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/A\*\_search\_algorithm