Artificial Intelligence for Games

|  |  |  |
| --- | --- | --- |
| **Assessment Task Number:** Task 4 – Design and Implement One Other A.I. Strategy for a Non-Player Character | | |
| **Unit Code(s):** | | **Unit Title(s):** |
| ICTGAM423 | | Apply artificial intelligence in game development |
| ICTICT525 | | Identify and manage the implementation of industry specific technologies |
| CUADIG517 | | Design digital simulations |
| **Items to Submit** | | **Evidence Criteria and Assessment Decision-Making Rules** |
| 1. | Other A.I. Strategy Implementation | A game containing an agent or non-player character implementing an A.I. strategy using any other A.I algorithm (i.e., not a State Machine) |
| 2. | Design Document | A 2-3 page detailed design of the state machine, including:   * A description of the real-world environment simulated, including any technical parameters * A description of the underlying functionality of the A.I strategy specifying essential settings, states, conditions, and parameters * How the agent interacts with the simulated environment * Difficulty levels and their controls, as appropriate |
| 3. | Record of Feedback | Seek and record feedback on your state machine design.  Record the feedback along with any changes you have made to your design or implementation as a result of the feedback you received.  At least one piece of feedback must be recorded. |
| **Submission Requirements:** | | |
| You will need to submit the following:   * A Release build of each application that can execute as a stand-alone program * Your complete Visual Studio project * A Design Document and Record of Feedback in MS Word or PDF format. These items may be in the same document.   Be sure to remove any temporary build folders (i.e., the Debug and Release folders). Only project files, source code files, and any resource files used should be included in your submission.  Package all files in a single compressed archive file (.zip, .7z, or .rar) | | |

Student’s own self-assessment of outcomes

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Competency | Description | Result |
| [Task 1](#Task1) | Implement one other **non-finite state machine** decision-making algorithm within your application (for example, decision trees, behaviour trees, blackboards, etc.). | Implemented a UtilityAI which selects a the more suitable of two Behaviours based on a comparison of numerical scores. |  |
| [Task 2](#Task2) | A detailed design document of the state machine. | A description of the real-world environment simulated, including any technical parameters |  |
| A description of the underlying functionality of the A.I strategy specifying essential settings, states, conditions, and parameters |  |
| How the agent interacts with the simulated environment |  |
| Difficulty levels and their controls, as appropriate |  |
| [Task 3](#Task3) | Record the feedback received and actions taken in response (minimum 1). | 6 topics recorded (combination of feedback given and received) including best practice changes, logic flaw bugs and ‘flow’ optimisations. |  |

1. Utility AI Implementation

Please refer to attached Visual Studio project.

1. Design document
   1. A description of the real-world environment simulated

The program simulates a physical space, inside of which are agents that navigate that space according to various behaviours that they have been assigned, such as wandering randomly or seeking other agents. The space is divided into a bordered ‘map’ of equally sized square tiles, with every tile containing a ‘node’ that defines the parameters of that tile.

Parameters of a tile include whether the tile is traversable by agents or not, visibility over which other tiles may be reached via this tile, and a set of numerical values that influence the degree of priority a given tile will receive when agents consider traversing it. These values include the number of tiles between the agent’s starting position and the tile (‘g’ score), the straight-line distance (irrespective of tile navigability) between the tile and the agent’s desired destination (the heuristic or ‘h’ score), and the sum of these two values (‘f’ score).

* 1. A description of the underlying functionality of the A.I strategy, and how the agent interacts with the simulated environment

**AI strategy chosen (combination wander & seek)**

There are several agents on the map. One agent (the player [yellow]) is controlled via left mouse click, at which time the player will navigate its way across the map to the tile clicked on if it is navigable, or if not, it will stop moving and remain where it is at the time of mouse click.

The behaviour of one non-player agent (the Utility AI [orange]) is governed by **wander** and **follow** behaviours. At the moment of each discrete behaviour decision, both behaviours calculate a numerical suitability score, and the AI autonomously selects the behaviour with the higher-evaluating score. In this demonstration, the behaviour with the higher score will always be selected – there is no weighted-average or random selection component. Both behaviours calculate their score according to the 2D Cartesian distance between the player and the AI, with **wander** becoming more suitable as the distance between the two increase and **follow** becoming more suitable as the distance decreases. The score of the **follow** behaviour only exceeds 0 (minimum suitability) when the distance between the two agents is 10 or fewer tiles. The suitability of behaviours crosses over at a distance of 5 cells.

Behaviour suitability is re-calculated approximately every 3 seconds. Movement paths of all agents are to be chosen from the possible routes according to an A\* pathfinding algorithm, with preference given to the route with the lowest ‘f’ score. Tiles are connected to one another both on cardinal compass directions and also diagonally. A\* pathfinding, once calculated, is visually ‘smoothed’. This program does have the capacity to adjust the cost of traversing from one tile to another (making certain paths more or less desirable). Smoothing results in agents no longer explicitly traversing the tile network paths themselves, and so the ‘cost’ of travelling from one tile to another is lost, resulting in a standard movement speed for agents.

|  |  |  |
| --- | --- | --- |
| Behaviour | Behaviour-induced action | Behaviour activation is conditional upon… |
| Wander | Move toward a randomly selected navigable tile on the map | The suitability score of one behaviour exceeding the suitability score of the alternative behaviour. Where the straight-line distance (irrespective of tile navigability) between the AI and the player is greater than or equal to 5 tiles, Wander is more suitable. Where the distance is less than 5 tiles, Follow is more suitable. |
| Follow | Move toward the player (or its last known location if follow condition no longer met) |

**Essential settings, states, conditions, parameters**

* The map of node connections must completely and accurately store the navigation information of connecting nodes in order for tiles to represent a network of navigable paths.
* The 2D Cartesian coordinates of an agent and its goal must be unambiguous to calculate the straight-line distance (the ‘h’ score).
* The pathfinding algorithm which calculates the preferred path of an agent to its goal must accurately calculate the relative ‘f’ score of each tile in each potential path.
  1. Difficulty levels and their controls, as appropriate

This simulation has no difficulty levels.

1. Task 3
   1. Record of feedback

On 03/08/2023 I provided my project to Michael Burford to review and provide feedback. This included:

1. Reporting back to me an error that arose on program closure originating from my Agent.cpp destructor.

Because my Agent classes are instantiated as objects, not pointers to objects, I cannot call delete on them at the end of the program, yet they themselves *have* instantiated pointer objects (Behaviour objects), and so I created a function which I invoke for each non-pointer Agent at the end of the program, which deletes the Behaviours of those Agents, while not calling ‘delete’ on the Agent itself.