Game Dev – Winter 2018 – Q4

Ian Sweeney

* 1. In modern games AI typically form the controllers for all NPC characters, AI can also find minor uses in other parts of games for intelligent generation of game maps, objects or stats
  2. Finite State Machines operate using a set of states and transitions which are manipulated by case-based programming to implement AI using a series of stages, these stages are described as follows:
* Sensing Stage – which determines the transitions which need to be applied to the thinking stage
* Thinking Stage – which uses the current state and the transition given by the sensing stage to determine the next state
* Action Stage – which uses the state to determine the next action to take
  1. The advantages of FSM’s are that they are very fast, not computationally intensive and easy to code
  2. Hierarchical FSM’s are FSM’s which are found within the states of another FSM, this gives the advantage of being able to make more complex and varied decisions.

Markov Models are similar to standard FSM’s but instead of adhering to rigid rules to determine transitions they use random number generation to similar dynamic and natural choices

* 1. Search Problems work by using a “Search Tree” paired with a Min/Max Algorithm.
     + The Search Tree is a tree of data which represents all the possible outcomes from a particular situation with a numeric value representative of whether that outcome is preferred or not.
     + The Min/Max algorithm works its way upwards from the bottom of the search tree, making decisions based on the numeric values of the outcomes.
  2. The Utility Function is the function which handles choosing which path to take based on its numeric value, these decisions can be beneficial (higher numbers) or detrimental (lower numbers) to the player.
  3. The Evaluation Function is the function which handles evaluating all potential outcomes, based on the current state of the problem.
  4. In a search tree problem the evaluation function can be used to form the tree itself, evaluating the potential outcomes which form the branches of the tree
  5. Creating an AI for a game such as Tic-Tac-Toe is far more reasonable than an AI for chess because Tic-Tac-Toe is a much smaller board with far less complex choices each turn, as a result the search tree for Tic-Tac-Toe would be far smaller than a search tree for chess.

An AI for chess would require immeasurably large amounts of storage space to store its search tree and a large amount of processing power to make its decisions based on that tree.

The performance of such an AI could be dramatically improved by only storing a search tree representing a number of moves ahead of the current game state, rather than storing all possible outcomes.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 6 | | | | | | | | | | | | | |
| 3 | | | | -2 | | | | | | 6 | | | |
| 3 | | 5 | | 8 | | 8 | | -2 | | 6 | | 9 | |
| -1 | 3 | 5 | 4 | 8 | 2 | 2 | 8 | -5 | -2 | 6 | 2 | 9 | -8 |