Generalized Search Space AI

Thank you for taking the time to look at the code for the search space AI I developed for an assignment at university. We were asked to use DFS and BFS to find solutions to the Towers of Hanoi game. Because these algorithms are completely uninformed, I was able to make my project use a completely generic version of the algorithms, meaning any person can define custom classes with my framework and use DFS or BFS to solve their desired problem.

In the following documentation, I hope to walk you through the basics of this framework.

ArcBase Class

This class is intended to hold the change between two states. In the case of games, think of this as the move that is being taken, like moving piece *a* from space *b* to another space *c*.

StateBase Class

This class holds an actual state of the game, and can be determined to be a goal state, meaning the AI will stop here as it is a desired outcome of the search space.

Both the ArcBase and StateBase<> Classes must be inherited and defined for the search space to work. I have included the classes for Towers of Hanoi as examples.

Graph Class

Graph is the core of the AI as it contains the entire search space and the DFS/BFS algorithms.

```
//Create a graph and a state to be the start state
HanoiState otherstate = new HanoiState();
Graph<HanoiArc> mygraph = new Graph<HanoiArc>();

//Pass the start state to the graph and initialize it.
mygraph.AddToStartStates(otherstate);

//[Optional]Initialize the state to be a start state as defined by the class.
mygraph.InitStartStates();

//Uses the respective algorithm to find goal states. The bool is if you want the algorithm to be Verbose via the classes defined print functions.
mygraph.DepthFirstSearch(false);
mygraph.BreadthFirstSearch(false);
```