**Behavior Tree Editor Window with Runtime Execution Support**

**Summary:**

To build a custom Behavior Tree Editor Window for Unity that allows content integrators or developers to create behavior trees visually via the Editor. The trees are built all without writing code for the tree building hirearchy. This setup allows rapid prototyping of AI behavior without hardcoding trees every time new tree or nodes need to be created or added.

**Problem:**

In traditional Unity setups, behavior trees are often hardcoded or require third-party visual editors. This made it time-consuming to test or iterate AI behaviors during development. I wanted a faster way to prototype behaviors without constantly rewriting scripts like AIChar, and ideally, build everything visually inside Unity.

**My Approach:**

My algorithm:  
**Create a window → Create UI in the window → Add functionality using buttons.**  
Instead of attaching an AI character script and hardcoding behavior, I created constructors for each strategy. These are used to generate **leaf nodes**, which can then be attached to a **sequence** node. Sequences are added to **selectors**, selectors to **repeater decorators**, and finally the repeater node is assigned to the **tree**.

I built a full UI that allows:

* Selecting node types (leaf, sequence, selector, repeater)
* Selecting strategy types for leaf nodes and entering required parameters (like AI/player/destination GameObjects)
* Creating nodes and organizing them into a hierarchy using dropdowns
* Assigning the root to the tree and saving it into memory for runtime access

I designed the system to allow users to set and adjust values at runtime wherever possible. The tree is stored in a static memory holder BTMemoryStore.tree, which is referenced at runtime to avoid losing the structure on play.

**Features:**

* Used different node types (Leaf, Sequence, Selector, Repeater) and strategies (PlayerFollowCheck, PlayerAction, etc.)
* Created a menu entry under CustomTools > Behavior Tree to launch the custom window.
* UI to:
  + Input node names.
  + Choose node types and attach strategies for leaves.
  + Drag/drop GameObjects like AI and Player directly into the editor window.
  + Attach child nodes to parent nodes hierarchically.
  + Build a full behavior tree with runtime memory storage.
* Tree is stored using a static BTMemoryStore.tree variable for runtime access.

**Strategy Support:**

* PlayerFollowCheck
* DestFollowCheck
* PlayerAction
* FollowPlayer

**Additional Details:**

* Parameters like AI GameObject, Destination, Player, and conditional booleans can be assigned per strategy.
* Runtime execution confirmed using debug logs after assigning the built tree to an AI controller script.
* I optimized the scripts necessary for the task.

**Further extensions:**

* If new strategies are coded, they can be added to the window and used for the leaf.
* The trees can be built before runtime and retrieved during runtime using scriptable objects(optimal) or JSON. I did test this with JSON successfully but to keep the implementation light weighted, I removed this part.

