# Worksheet – Behavior Trees - Unity

## Overview

### Objectives:

1. Understand Unity Script Code
2. Code Priority Selector
3. Implement Priority Selector

## Preliminary

This tutorial will make use of the Unity game engine and the project linked with this worksheet has been building in Unity 2020.1.17.1f.

The versions on campus or your own machine may not match exactly, but the project should import without issues.

## Getting Started – Understanding the Scirpts

Use the link on MyLearningSpace to download the file called **Behaviour Tree – Unity.** Unzip the file, and open in Unity.

### Scripts

The Unity project makes use of eight scripts:

**RootNode** - This script allows for us to call and access the tree. There is only one root node and it should be attached to the GameObject that represents the root node in the scene.

**TreeNode** - Script that starts as the basis for the rest of the tree. All nodes inherit from this class. Contains a virtual function that runs the nodes code.

**SelectorNode** - Contains a list of child nodes and sets up capturing them when the node is first initialised. Extends into all other types of selector nodes. When reset, resets all child nodes.

**SequenceSelector** - Runs the child nodes in sequence, with the first in the list being the one run first. If a node fails, it also fails. Succeeds when all child nodes have succeeded.

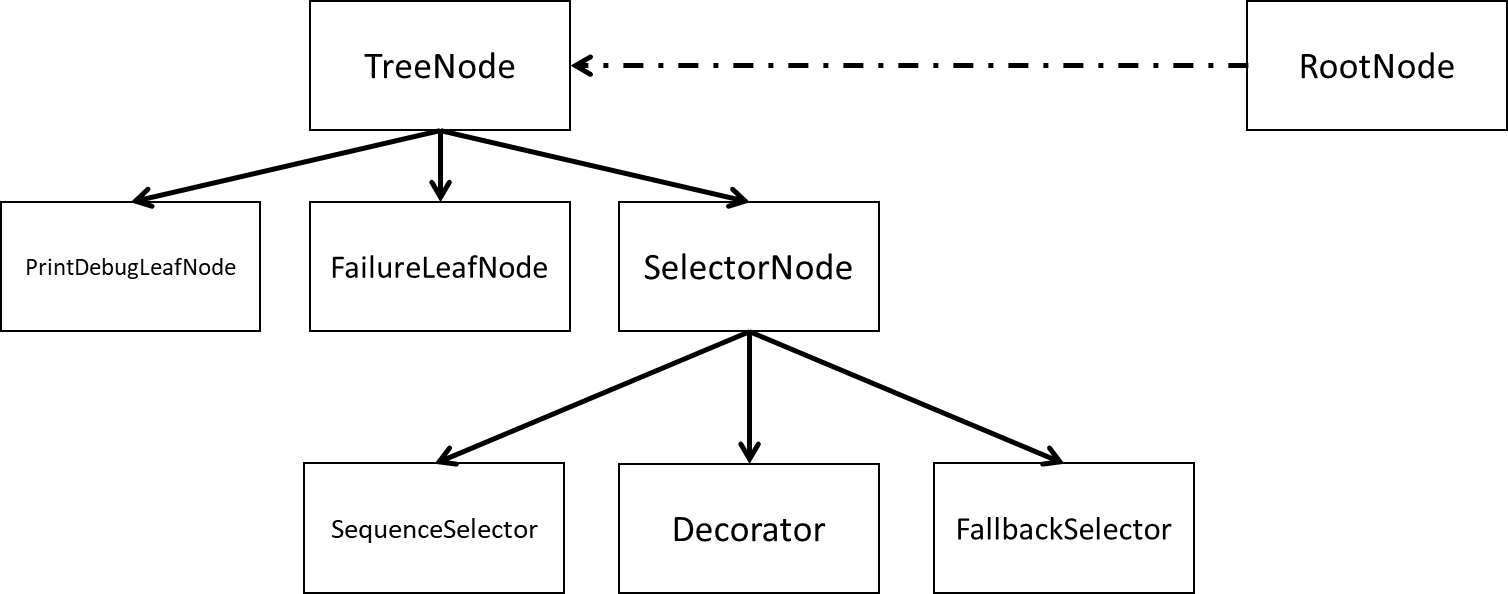
**FallbackSelector** - Runs the child nodes in the same sequence as the Sequence Selector. However, if a single node succeeds, it succeeds. Only fails if ALL child nodes fail.

**Decorator** - Meant for managing time access to a single node. Runs the node for a set amount of time then considers itself and the child node successful. Fails if the child node fails.

**PrintDebugLeafNode** - Prints a debug message then waits for a specified amount of time. Can only succeed.

**FailureLeafNode** - Runs for a set amount of time then forces itself to enter a failure state.

The structure of classes and inheritance is like so:



1. Read through the scripts and run the base project. The nodes should change colour as they run and some of the leaf nodes will output messages into the Debug view in the Unity Editor.

Make sure you understand how the classes work, which inherited functions are overwritten and why.

Press the SPACEBAR to run the tree. Once the tree has finished, you can press SPACEBAR again to restart the tree.

1. Currently the Decorator node finishes before its child leaf node. Edit the Decorator or its attached leaf nodes timing variable so the leaf node finishes before the decorator. See how that changes the tree.
2. Remove the script from one of the leaf nodes on the sequence selector branch and replace it with a **FailureLeafNode**. The entire tree should now fail.

## Code a Priority Selector

1. Create a new script for the Priority Selector, make sure it inherits from **SelectorNode**
2. You can design the priority selector to order the child nodes via priority as you wish. A suggestion would be to have public list with names of nodes and a priority value. Enter the name of the child nodes and give them each a priority number.
   1. The selector will choose the node with the highest priority
      1. Said node cannot be in the SUCCESS or FAILURE state
   2. Keep track of which node is currently being run
   3. The priority can either stay the same, or can change during the running of the Tree.

You can either:

* + 1. Implement input controls to change the priority of items
    2. Have an algorithm that changes the priorities over time (increase over time, random increase, etc.)

1. Add the script to a GameObject.

## Implement Priority Selector

1. Replace one of the leaves within the tree with the priority selector and give it three leaf nodes, each which will print a message to the Debug console.
2. Run the project and make sure it works properly.

## Extra

1. (Optional) Create another selector class that replicates the operator of a Concurrent Selector. When the selector is run, you want it to create a thread or subroutine that runs all of its leaf nodes.
   1. If one fails, the selector fails
   2. If all succeed, then the selector succeeds
   3. Make sure to wait for all the threads to finished before the selector can set its own state to success or failure.