Pedagogical Report

Github Link: https://github.com/Ziyuan-Yin/CSYE7270_Final_Exam

1. Teaching Philosophy

Target Audience

University student or Unity starters who have coding basic and knows how to build a simple unity project, but don't know about Behavior Trees.

Learning Objectives

After completing this lesson, students should be able to:

- Explain what a Behavior Tree (BT) is and why it is used in modern games.
- Understand the three node states: Success, Failure, and Running.
- Implement a simple Behavior Tree system, including Sequence, Selector, condition nodes, and action nodes
- .Build a working AI behavior such as patrolling and chasing the player.

Teaching Rationale

Behavior Trees are widely used in real games because they are modular and readable. They are also much easier for students to understand compared to state machines or reinforcement learning.

My teaching approach emphasizes:

- **Learning through action** (students implement the BT as they learn concepts)
- **Visual feedback** (students can see the enemy move and switch states)

2. Concept Deep Dive

Technical Explanation

A Behavior Tree is a hierarchical structure of nodes. Each node returns one of three states:

- **Success** – the node finished and the condition or action worked

- **Failure** – the node finished but the condition or action did not work
- **Running** – the node has not finished yet and will continue next tick

Composite nodes organize the behavior:

- **Sequence**
  Runs children from left to right.
  Fails immediately when one child fails.
  Succeeds only if all children succeed.
- **Selector**
  Tries children one by one.
  Succeeds when any child succeeds.
  Fails only if all children fail.

Leaf nodes do the real work:

- **Condition Nodes** (example: IsPlayerInRange)
- **Action Nodes** (example: MoveToPlayer)

The tree is evaluated every frame.
The evaluation can be described mathematically as a depth-first traversal
with early termination depending on node states.

Connection to Game Design

In real games:

- Designers want readable and flexible behavior.
- Programmers want AI logic that can grow without rewriting large parts of code.
- Behavior Trees allow both by splitting decisions into small reusable blocks.

3. Implementation Analysis

Architecture

The implementation uses a clean, modular structure:

- **BTNode** (abstract class)
- **SequenceNode** and **SelectorNode** (composite behavior)
- **Condition nodes** and **Action nodes** (patrol, chase, distance checks)
- **EnemyBTController** (builds and runs the Behavior Tree)

This architecture is easy to understand because each class has one responsibility.

It also follows common Behavior Tree patterns used in professional engines.

4. Assessment & Effectiveness

Validation Criteria

Students are considered successful if they can:

- Build the chase behavior using Sequence + condition + action nodes
- Make an enemy patrol and switch to chase when the player is close
- Explain in their own words how a Sequence and Selector work
- Modify the Behavior Tree to add new simple behaviors
- Debug incorrect behavior by reading the tree logic

Expected Student Challenges

Students may struggle with:

- Understanding why nodes must return Success / Failure / Running
- Predicting how Sequence and Selector combine node results
- Connecting the abstract tree diagram to actual movement in Unity