Behavior trees and state machines with LimboAI

LimboAI addon overview

We'll be using LimboAI for behavior trees and state machines

LimboAI docs | LimboAI on GitHub

LimboAI includes a demo project. It is **highly recommended** that you download it and check it out. When you run it, it has a nice little tutorial that walks you through the basics:

https://github.com/limbonaut/limboai/tree/master/demo

In its classes and nodes, LimboAI uses the following prefixes:

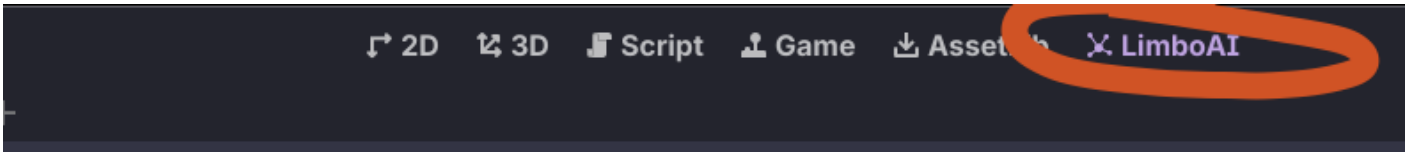
- `BT`: Behavior Tree
  - Nodes/classes that start with this have to do with parts of behavior trees
  - As an example, the `BTPlayer` node "plays" a behavior tree every frame and inside of that tree you have `BTSequence`, `BTSelector`, etc
- `BB`: BlackBoard
  - The blackboard is a shared place for agents to write values to.
  - These values can be read and written by any task in the tree.
- `LimboHSM`: Hierarchical State Machine
  - HSMs have `LimboState`s inside of them.
  - `LimboHSM` is itself a `LimboState`, and `BTState` can put states inside of behavior trees

Behavior trees

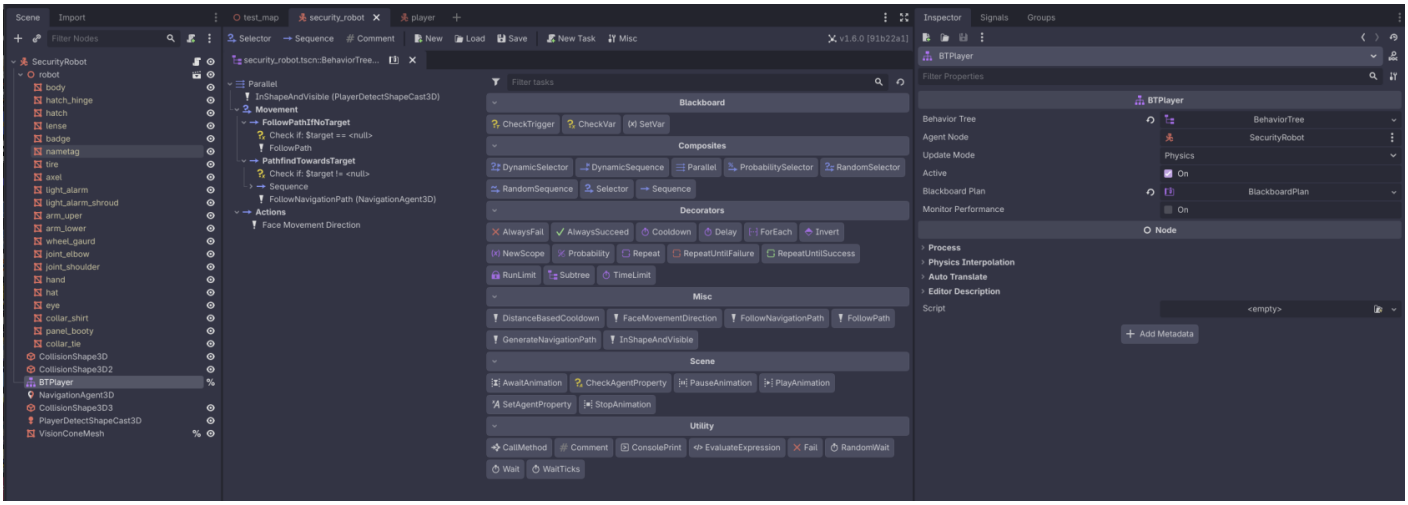
Read the intro to behavior trees here

Accessing LimboAI behavior tree editor

When installed, LimboAI adds a button to the top of the Godot editor, alongside the 2D/3D/Script buttons



When you click on a `BTPlayer` node, it has a `BehaviorTree` attached to it. When clicking that, it will automatically switch over to the LimboAI tab.



Building blocks: sequences and selectors

The main building blocks of a behavior tree are:

- `BTSequence`: Think of these like an AND operator on the child tasks.
  - As long as a task return `SUCCESS`, the next one will be executed.
- `BTSelector`: Think of these like an OR operator on the child tasks.
  - On the first task that returns `SUCCESS`, it will stop executing.

Writing custom tasks

Read the documentation on writing custom tasks.

Custom tasks go inside of `ai/tasks` in the project.

At the time of writing, we have:

- `follow_path`: Follows a `Path3D` node (i.e. for preset patrol paths)
- `generate_navigation_path` / `follow_navigation_path`: Using a `NavigationAgent3D`, generate and follow a path to a `target` in the blackboard
  - `follow_navigation_path` will follow the path until it gets close enough to the `target`, it will return `RUNNING` until it gets there

- `distance_based_cooldown`: Takes in a `Curve2D` and uses that to do a cooldown based on how close the agent is to the `target`
  - Useful for generating navigation paths more frequently the closer you are to the `target`, and less frequently the further you are
- `in_shape_and_visible`: Returns `SUCCESS` if there's a target in a `ShapeCast3D` that is visible (hit by a raycast, in "front" of the agent)
  - This also sets the `target` when it is visible

State machines

https://limboai.readthedocs.io/en/stable/hierarchical-state-machines/create-hsm.html

---

Revision #1
Created 2026-02-19 21:32:43 PST by TranquilMarmot
Updated 2026-02-19 22:46:53 PST by TranquilMarmot