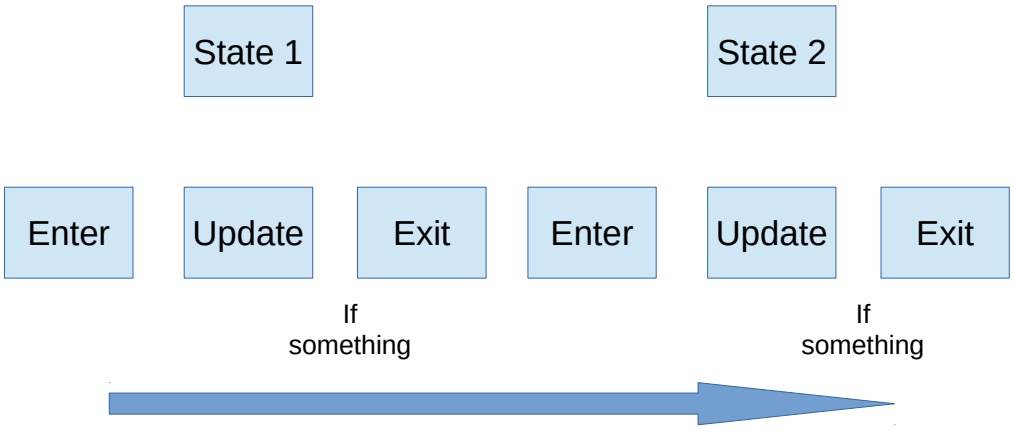
3-Behavior Tree

In this scene you have an example of a very simple behavior tree for our two monsters and how to reuse as much code as possible using delegates.

The game is simple , it's a top-down shooter, i'm using MasterPool and EventSystem so check their documentation in the (1-Event System) folder if you haven't already.

This isn't an AI project so I won't extend too much in talking about the AI behavior tree design nor I think it's a perfect one.

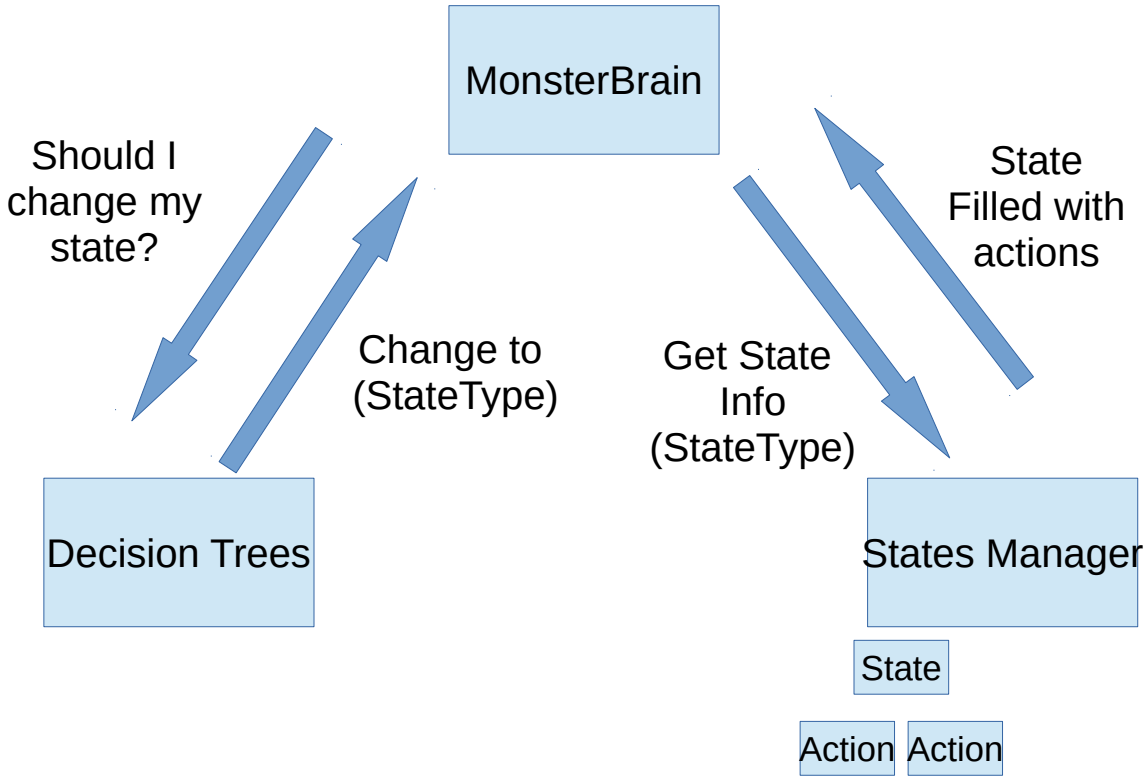
So to make our life easer every State will have Enter, Update, Exit delegate to store the logic that should happen when Entering, Staying, Exiting the state.
The states are being created and filled with actions in MonstersStates.cs, I prefer keeping me states separated to make adding and removing behaviors much easer.



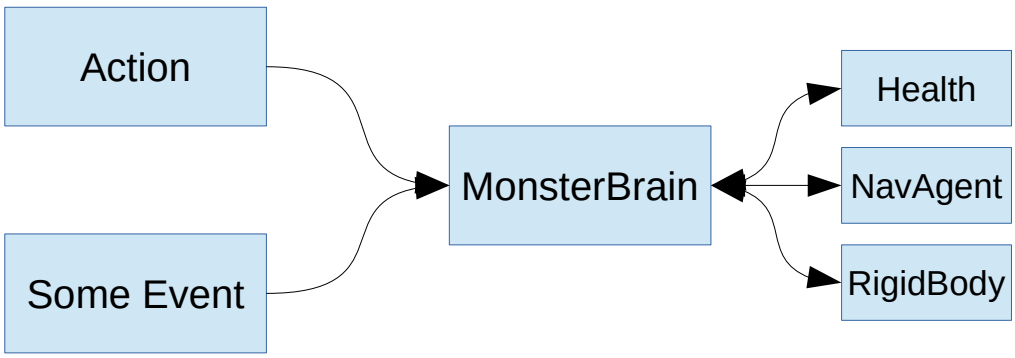
The only script that will be in the scene is MonsterBrain.cs and it will ask DecisionsTrees.cs to update it's state based on it's variables.
All the behavior trees logic will be in DecisionsTrees.cs again to keep adding and removing behavior easy, so to recap:

-MonsterBrain.cs: is the monster instance variables holder.
-MonstersStates.cs: is the States holder.
-DecisionsTrees.cs: is the monster logic on what state is should be in.

Also MonstersStates.cs is using MonsterActions.cs as a namespace to fill the states



So the variable that's being pass is MonsterBrain.cs so our three delegates in State.cs and all our Actions in MonstersAction.cs takes MonsterBrain as parameter to apply the action to the MonsterBrain.cs instance.



So by using Delegates to store state behavior we released the MonsterBrain and it's BehaviorTree from dealing with actions.