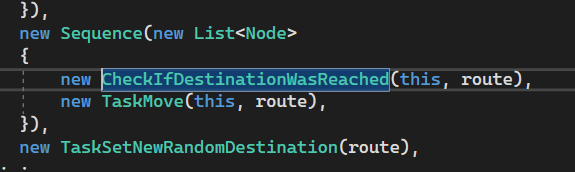
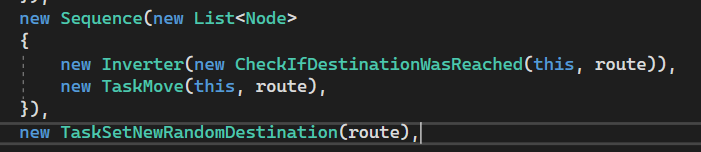
В общем, пробежался по твоей реализации.

Первое, что я заметил, была проблема с логикой последовательности проверки достигнутой точки (CheckIfDestinationWasReached) и задачи ходьбы (TaskMove).

Что было: проверяется, что точка достигнута -> выдаётся Failure, поскольку не достигнута ну и нет перехода к TaskMove.

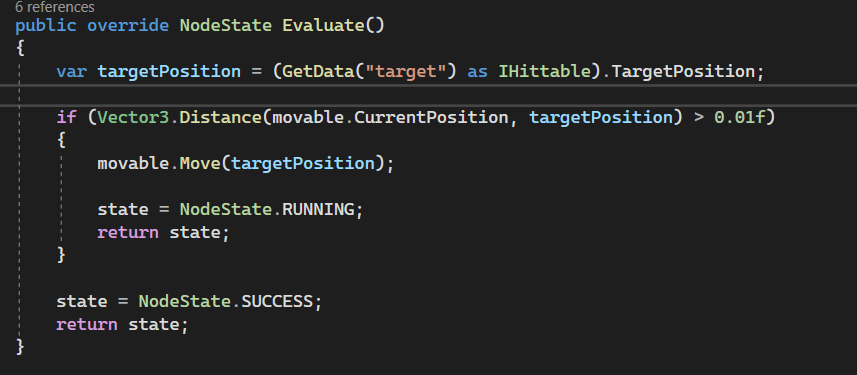


Что стало: В кодовую базу добавил Inverter. Думаю, что суть понятна из названия – инвертировать значение. Внизу показано, каким образом было исправлено.



Второе, что заметил, было следующее:

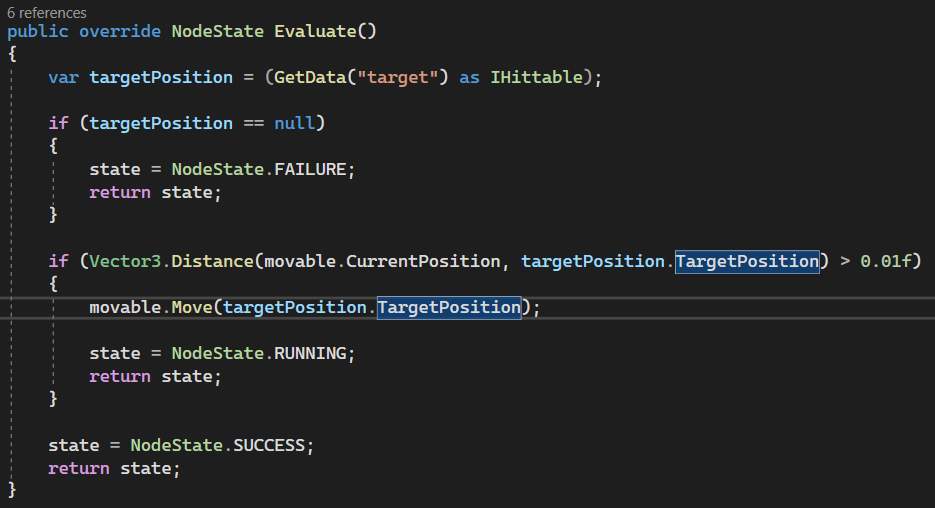
В классе TaskGoToTarget получение данных из ноды такое:



Проблема вот в чём. Ты получаешь данные, а потом берёшь оттуда позицию. А что будет, если данные = null?

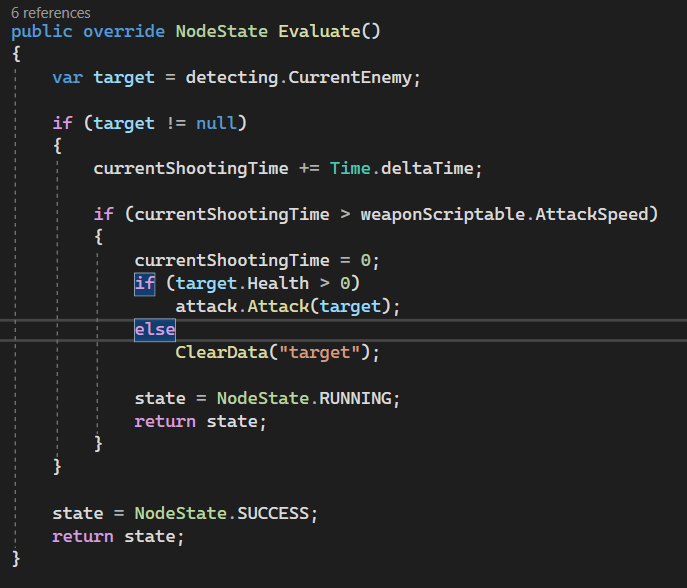
Что стало.

Когда писал на Verilog, проверки недопустимых ситуаций писались вначале, а потом уже дальнейшая логика .



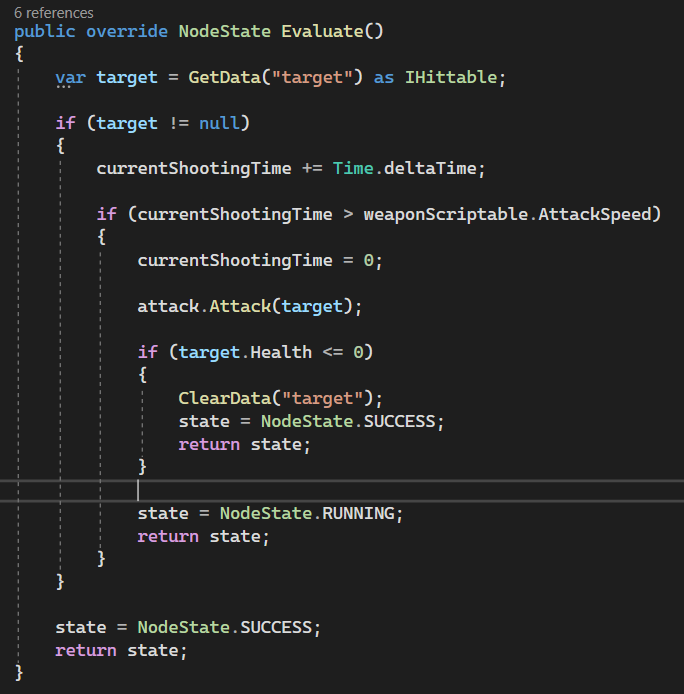
Третье, что заметил. Класс TaskAttack

Что было:

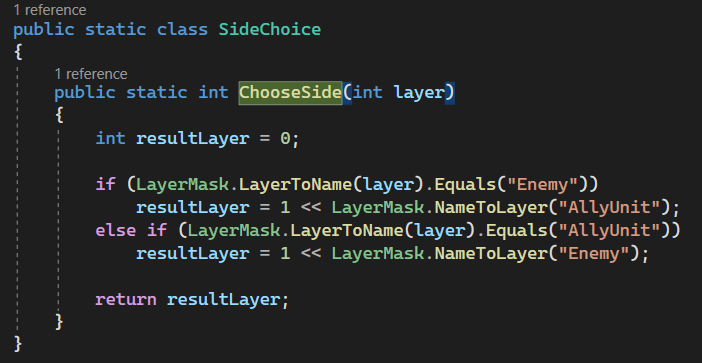


Тут, если честно, возникает путаница, когда пытаемся получить данные и от интерфейса, и от записанных данных в дереве. Давай так, если что-то и выбираем, то следуем до конца. А иначе потом будет путаница и поиск ошибок ☺ В логике проверки здоровья противника обнаружил очень интересный баг. Вместо живого существа, уже умерщвлённое тело, а референс на него остаётся, даже при проверке на null. Я бы тут предпочёл сделать следующее: ударить противника, а после проверить его здоровье. Ну и также, если мы его замочили и очищаем данные, то нам просто незачем снова залезать в эту проверку. Можем сразу выдать SUCCESS

Стало так



Остальное по мелочи. В классе SideChoice незачем при проверке каждого условия делать выход из метода. Можно просто присваивать переменной значение и в конце возвращать. Что-то в таком духе:



Ну и осталось добавить слои каждому существу

PS. Если хочешь, чтобы и PatrolMan помирал, то стоит добавить ему интерфейс IHittable