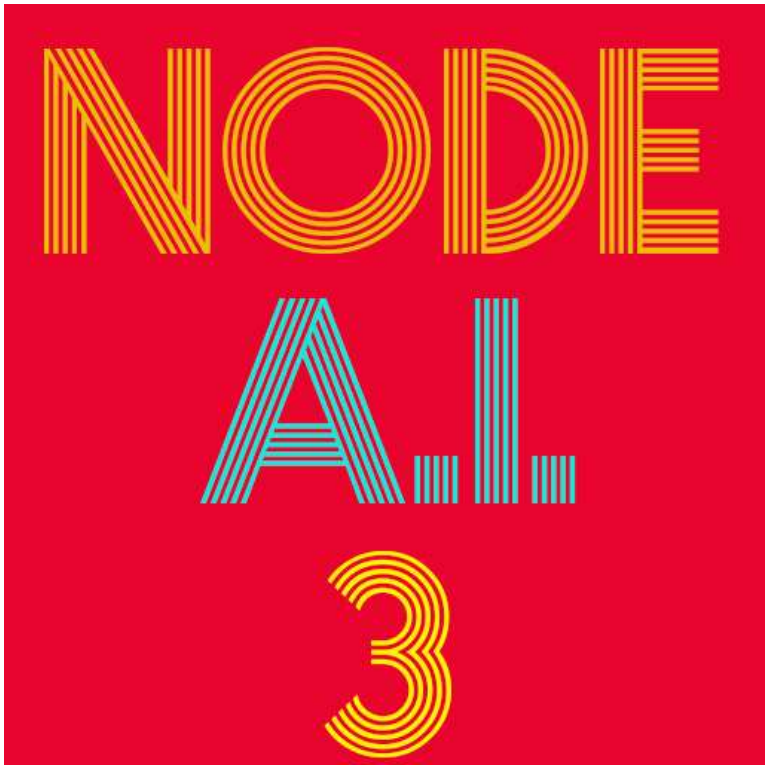


version 3.1

Thank you for purchasing Node A.I.! Reading this guide should equip you with all you need to know to get the most out of this system. If you have any other questions, you can contact me at nodeaiunity@gmail.com.

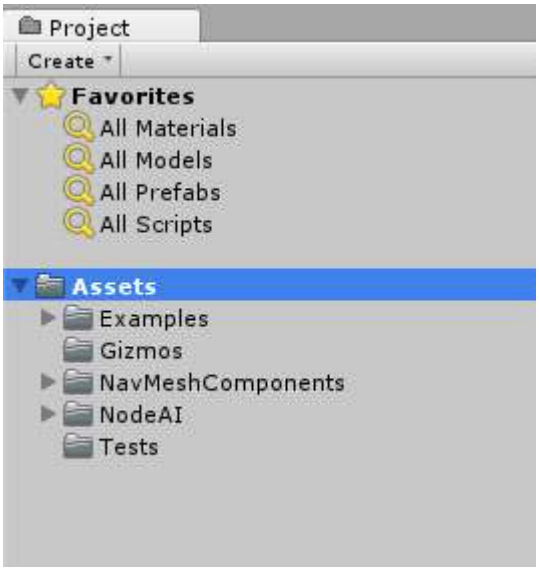
Quick Start Guide

1.Import the Node AI package from the asset store.

2. Download and import Unitys new navmesh components. I imagine that sooner or later they will be incoporated into the standard Unity package, but as of Unity 2017.1 they are located at

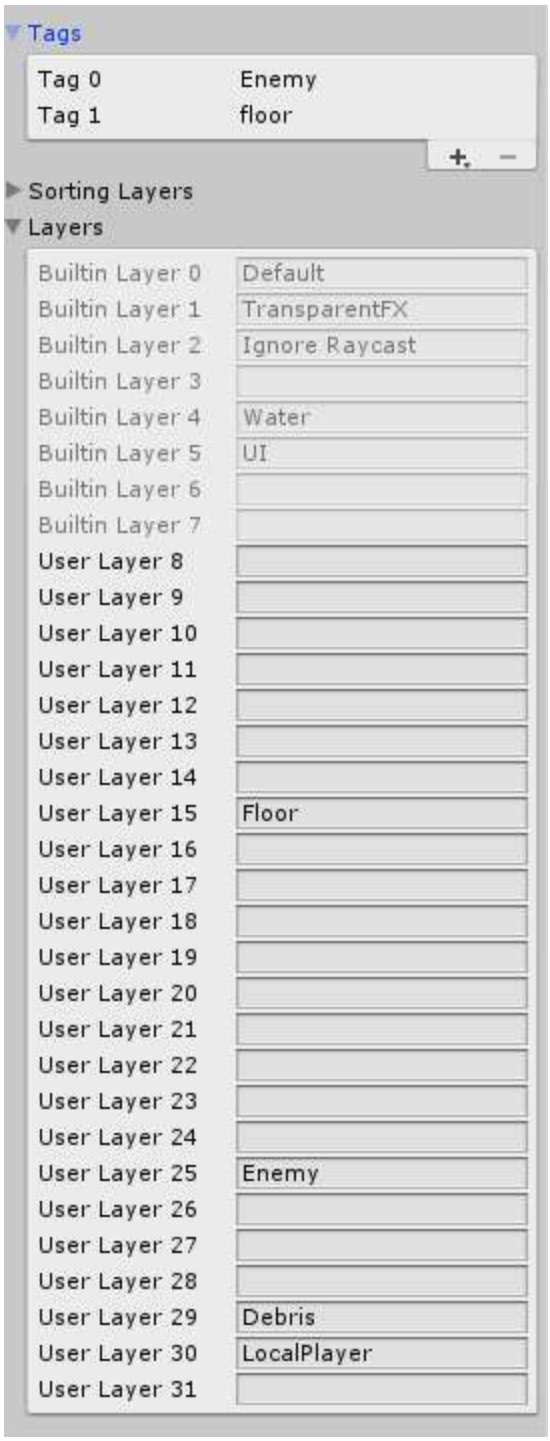
https://github.com/Unity-Technologies/NavMeshComponents

I suggest downloading the zip file, and copying everthing from the assets folder into your projects assets folder. At this point, your project should look like this:

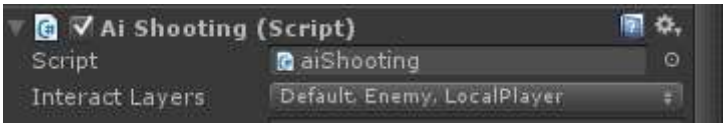


(note: you may want to delete the "Tests" folder. In my experience, it can cause problems when you try to build your project.)

3.Set up Tags and Layers. Here are the default assignments.

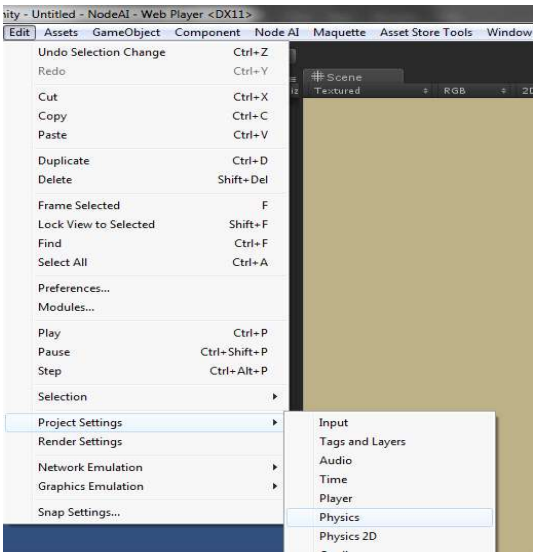

Tags
Tag 0 Enemy
Tag 1 floor
Sorting Layers
Layers
Builtin Layer 0 Default
Builtin Layer 1 TransparentFX
Builtin Layer 2 Ignore Raycast
Builtin Layer 3
Builtin Layer 4 Water
Builtin Layer 5 UI
Builtin Layer 6
Builtin Layer 7
User Layer 8
User Layer 9
User Layer 10
User Layer 11
User Layer 12
User Layer 13
User Layer 14
User Layer 15 Floor
User Layer 16
User Layer 17
User Layer 18
User Layer 19
User Layer 20
User Layer 21
User Layer 22
User Layer 23
User Layer 24
User Layer 25 Enemy
User Layer 26
User Layer 27
User Layer 28
User Layer 29 Debris
User Layer 30 LocalPlayer
User Layer 31

It is recommended that you use this same set up. It will reduce your setup time, however if you have an existing project or conflicting layer setups with other assets, you can assign these tags and layers to different slots. Just make sure your agent prefabs use the tag "Enemy" and the layer "Enemy", your player uses the layer "LocalPlayer" finally that the everything under the "_Level" gameobject in the "multiLevel" scene uses the "floor" tag. Also, on the node_AiShooting, make sure the interact Layers use these layers.

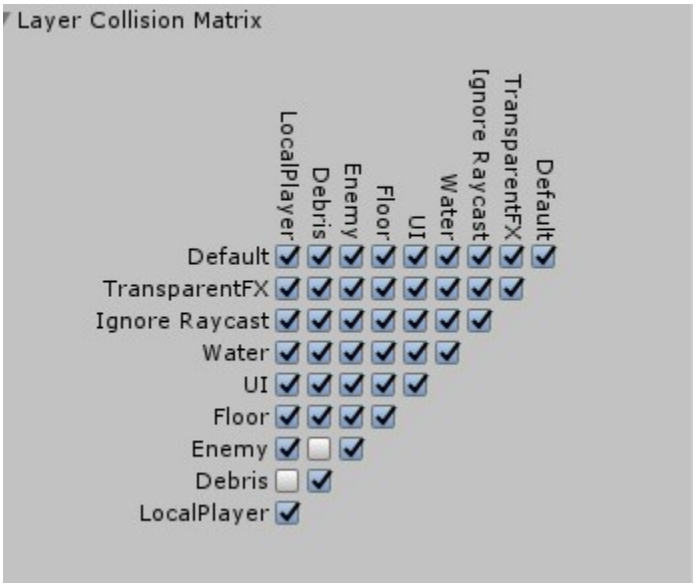


node_AiMovement interact layers are just "Default" and "LocalPlayer".

4.Setup physics



Select the physics project settings



and make sure it looks like this.

Lastly, if you want to use the included player controller, you need to unzip the "InputManager" zip file in the node AI folder, and use that to replace the InputManager file in your project/projectSettings folder. If you replaced this file correctly, you will see "Fire4" in the Input Manager in unity.

5. Open the "gamePlay" scene located in the nodeAI/scenes/ folder. Press play, and everything was imported/set up correctly, you should see your agents moving around and you can control the player.

Note: sometimes prefabs and/or the agents in the scenes will loose there layer assignments on import. If you are having issues, hover your mouse over a setting and it will tell you what to include on these lists. Typically interact layers are "default" and "localPlayer" player layers are just "LocalPlayer" and enemy layers are just "Enemy".

Coming from Node A.I. 3

Node AI 3.1 has a slightly different tag/layer setup, so double check all those settings. Also make sure your physics settings match the above settings.

AI Modes

Each A.I. agent has 4 modes: Wander, Node Wander, Waypoint or guard.

Wander Mode

if an agent is set to **wander mode**, the agent will randomly move around the scene ignoring nodes and wayPoints. In the inspector you can set the minimum amount of time an agent will wander (Wander Time Low), the maximum amount of time an agent will wander (Wander Time High).

The amount of time an agent will wander is randomly generated between wander time low and wander time high, so if you want the agent to wander for a set time, simply set these as the same number. If you don't want the agent to stop at all, set both settings to 0.

Node Wander Mode

Node wander is a better, more natural version of wander. Instead of wandering to completely random spots and staring at walls and nothing in particular, in this mode the agent will wander to particular destinations that you designated with nodes around your level.

Use Node Group

> If true, the agent will only wander to nodes with the group name defined in the '**Node Group Name**' setting.

Node Group Name

> If '**Use Node Group**' is true, the name of the node group the agent can wander through. The Node Group Names are case sensitive and must exactly match the name of the nodes in the scene to work.

Follow Sequence

> If true, the agent will wander through its list of nodes in order, good for patrol type behaviour. If false, the agent will pick a random node each time, good for search behavior.

Way Points Mode

In way points mode, an agent will patrol through a number of wayPoints assigned to it in the inspector, or use custom wayPoint scripts. If no wayPoints are assigned, the agent will default to **guard** mode.

Default wayPoints

Default wayPoints are easy to setup. Simply drag the wayPoint prefab from the prefabs folder into your scene and place them where you would like your agent to patrol. Make sure your wayPoints are easily visible, usually 1

unit above the ground.

Once you have your wayPoints set up, you need to drag them into your agents wayPoints list. The list is located under "**wayPoint Settings/wayPoints**" drag them in the order you would like them patrolled. When the Agent reaches the last wayPoint, it will move towards the first.

Conditional wayPoints

If you want an agent to use this system, click "**use way point script**" in the agents wayPoint settings. Conditional wayPoints are not for beginners, and require scripting to use properly. A demo can be found in the "**conditionalWaypoint**" scene.

Guard Mode

An agent in guard mode doesn't move until chase mode is triggered. If it looses the player, it will return to its initial guard position and rotation.

Chase Mode

Chase mode is activated when an agent notices a player. An agent can notice a player in multiple ways:

-The most common way chase mode is activated is when an agent "sees" the player. Agents wont notice players right away (unless you want them to) but react to a player sighting based on distance, and reaction time settings. The minimum amount of time it takes an agent to react to a player sighting can be adjusted under "**detection settings/reaction time low**" and the maximum under "**reaction time high**". Use "**reaction time smooth**" to adjust the reaction time based off of distance. A lower reaction time smooth will make the agent react slower at long distances, while a higher setting will make it react faster. Set **reaction time low** and **reaction time high** to the same number if you want a set reaction time (this could be set to 0 for instant reaction). The marker above the head of the default enemy

prefab lets you visualize the reaction time.

A quick note about the marker above the default enemy prefab

marker

The marker above the default enemy prefab is a very useful tool for tuning your agents behavior. When the marker is green, the player is out of range. The agent won't notice the player, even if the player is directly in front of the agent. If the marker is yellow, it means the player is within the agents visibility range, and the agent will notice the player if the player enters the agents line of sight. This setting can be adjusted under "**detection settings/normal vision distance**". When the marker starts fading from yellow to red, this represents the agents reaction time. The player can avoid detection by breaking line of sight with the agent before the marker turns red. If the marker turns red, the agent has noticed the player, and chase mode is activated. If you don't want to use the marker system, simply delete the marker.

Chase mode continued

-If a player gets to close to an agent, regardless if an agent can "see" the

player, the agent will notice the player. You can adjust this setting under "**distance settings/personal awareness distance**".

-Agents can alert other agents when chase mode has been activated. You can turn this on or off with the "**detection settings/alert other agents**" option.

-You can activate chase mode via code any time you want. For instance, you might want agents to "hear" a gunshot and investigate the location it came from. Or maybe an agent set off an alarm. All you need to do is send the message "wakeUp" to an agent to trigger chase mode. For an example, check out the "fireBullet" script located in the "scripts/player scripts" folder.

Chase type

New for Node AI 3.1 is the "chase type" option. "Direct" mode makes the agent run directly towards the player, which is more suitable for melee, while "surround" mode makes the agent pick a space around the player, and change position (find this setting under advanced settings, position change low and high) from time to time. This is more suitable for ranged agents, and provides a much more dynamic AI type.

Chase Mode - Attacking the player

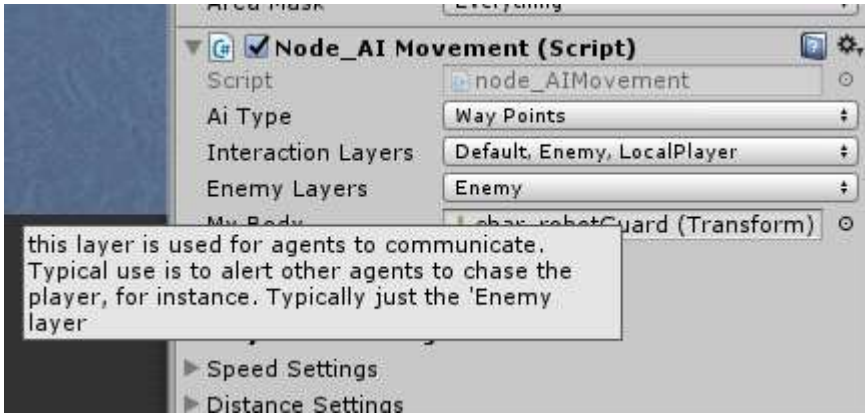
In chase mode, the agent will trigger a mode where it can attack the player (if you are using the included prefabs/shooting script). The attack range can be adjusted under "distance settings/attacking distance". Agents can attack and chase at the same time, the agent will only stop chasing once it reaches its stopping distance. The stopping distance is a random number between the "distance settings/stopping distance low" and "distance settings/stopping distance high" settings. To make a set stopping distance, assign the same value to both of these fields.

If the agent looses line of sight of the player, it will still chase the player for how many seconds you specify under "**detection Settings/search time**".

The agent will stop chasing the player after this time, and remain cautious in the area for how many seconds you set under "**detection Settings/camp time**".

If the agent hasn't seen the player in this amount of time, it will resume its previous behavior.

Node_AI Movement Settings



To get more information about the various settings on the node_AIMovement script (and most other scripts) simply hover your mouse over a setting you are curious about and you will see a tool tip explaining its function.

Nodes

With the move to version 3 of Node AI, nodes have taken a bit of a back seat. They are no longer required at all, but they can still be quite useful in making your ai more dynamic and natural.

The Node Manager

The node manager is a Gameobject for storing a list of nodes in your scene. If any of your agents are using node wander mode, you must have this script in your script in your scene, and it must use the "GameController" tag. Otherwise they will default to guard mode.

You can add nodes to the list in two ways. First, just drag and drop them on

the node lists. or second, dont drag them in, just make them a child of the nodemanager gameobject and tick "nodes are children". Nodes will be updated and added at runtime. Check out the gameplay demo for an example.

Setting up Nodes

The node prefab has a very simple script attached that allows you to visualize its placement, and assign a group name (not required). Place the nodes anywhere you like, typically at points of interest or other important areas of your scene. If you use Nodes and Node Wander mode, your agent will wander between these smartly placed nodes, as opposed to plain wander mode where the agent will wander mindlessly.

Setting up the Player

Node AI 3.1 comes with an all new, top down shooter player controller. The player controller supports both mouse+keyboard and gamepad. Check out the gameplay scene to see it in action. If you want more info about a particular settings, simply hover your mouse over it for a tool tip pop-up.

Scene Summary

conditionalWaypoint- This scene is an example use of conditional wayPoints.

gameplay - This a small example scene featuring agents who use node wander, waypoints, and guard. The main demo scene, and a good scene to get an idea of how the different agent modes can work together.

multiLevel - This scene features a click to move Agent.

runTimeExample - this is a scene showing how to use node AI to generate a node graph dynamically at runtime. Check out the "**Procedurally generated level setup"** section for more info.

Prefab Summary

Agents

_RagDollAI_Guard This agent will not move unless it sees the player. If it chases the player and looses its trail, it will return to its initial position and rotation.

_RagDoll_Node_Wander This agent requires nodes. An agent setup and ready for gameplay. In a scene with nodes, you can drag a couple of these dudes in to quickly test out your scene.

_RagDoll_Wander_Melee - Basic ai setup. uses the default wander mode, and doesn't require nodes. Only uses melee attacks.

_RagDoll_Wander_Ranged - Basic ai setup. uses the default wander mode, and doesn't require nodes. Only uses ranged attacks.

_RagDoll_Wander_Ranged_Melee - Basic ai setup. uses the default wander mode, and doesn't require nodes. Uses ranged and melee attacks.

_RagDoll_WayPoints An agent setup to use waypoints. Requires waypoints (of course).

Demo Prefabs

Camera - The camera prefab to be used with the runTime3rdPersonController prefab. Contains a script for following the player around.

Canvas - HUD elements for the new player control. Includes health bar, damage flash, and control information.

hitFlare - This is a particle effect that plays when a defaultEnemy "dies" in the demo scenes.

HUD - Canvas gameobject with on screen controls for the multiheight demo scene.

marker - Marker model featuring a custom shader that allows it to be scene behind objects.

moveableWayPoint- wayPoint used in multiLevel click to move demo.

SoldierModel - This is an ancient unity model, used in many projects. I included it for demonstrating how to change character models in the tutorial videos.

Navigation prefabs

_NodeManager - Required for any scene that uses nodes. Required to use the tag "GameController". Stores list of nodes in scene.

node - basic Node.

wayPoint - WayPoint prefab to set up patrols for your agents.

Player Prefabs

Player_TwinStick - The new twin stick controller. The model is taken from the unity "adventure game" asset available for free on the asset store.

Script Summary

Please note that most of these scripts contain many tool tips, comments and notes within them, describing most of the variables. This is just a general overview, for more in depth understanding of the scripts open them up and take a look. Even if you aren't a programmer, it might be

useful to look at what exactly certain variables and settings do.

3rd Party Scripts

Includes zip files containing scripts for use with ufps and dungen. Check out there respective sections of this manual for more info.

AI Scripts

node_AIAnimation - A simple script to control the defaultEnemys animation. Mecanim compatible.

node_aiHealth - This script controls your agents HP, and what (if anything) to spawn on death. This script is not required.

node_aiMovement - The most important script in this package. This is the AI "brain" and required on all agents using this system.

node_aiShooting - A simple script used to allow agents to attack the player. Eventhough it's called aiShooting, it also contains melee attack settings.

Demo Scripts

demoLevelSelect - Used in the nodeDemo Scene. When the player enters the red sphere in the level, this script resets the level.

followCam - script used to control the "camera2" prefab.

levelControl - This simple script allows the player to exit the game by pressing the escape key, or to restart the level by pressing the p key.

node_conditionalText- used to disable/enable on screen text in the conditional wayPoint demo scene.

node_conditionalWayPointHandler - attach this script to agents using advanced wayPoints.

node_demoSpawner - script used in the proceduralExample scene. Check out the procedural generation section of this manual for more info.

node_Gizmo a simple script to give your transforms a visualization. useful

for wayPoints.

node_moveWayPoint - This script is used in the multiLevel scene to move the waypoint around.

node_wayPointConditional- Script used in the conditional wayPoint demo scene that allows you to select the next wayPoint destination with the number keys.

Misc Scripts

quickDisable - Use this script on "Loading" text for scenes that take a bit to load.

selfDestruct - Attached this to items you want to destroy in a certain amount of time.

Node Scripts

node_node - use to visualize nodes and assign them group names.

node_nodeManager - Every scene that uses nodes requires a gameobject with this script in order to work.

New Player Scripts

For more in depth coverage of the new player controller, please check out the tutorial video at: https://www.youtube.com/playlist?list=PLGrMvTMQkr2M6gpn1pQtDBs9Tw4cYgouX

node_IKHandling - This script is used to position the player characters hands correctly on a ranged weapon.

node_objectPooler - This is a simple pooling script, and can actually be used for any number of things. But in this case, it is used to pool the bullet impact particle fx.

node_PlayerHealth - Keeps track of your players health.

node_playerMovement - Controls the players movement.

node_playerShooting - Controls how the player attacks.

Procedurally generated level setup

Open up the runTimeExample scene to follow along with this section.

The basic workflow for using Node AI with procedurally generated scenes goes like this:

1. Create level geometry

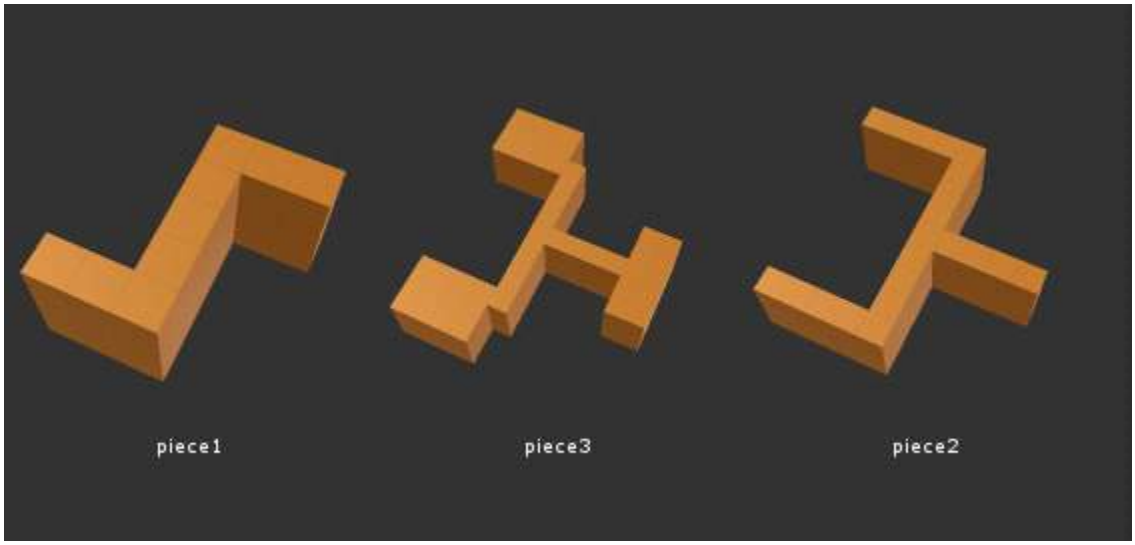
2. build navmesh

3. spawn Agents

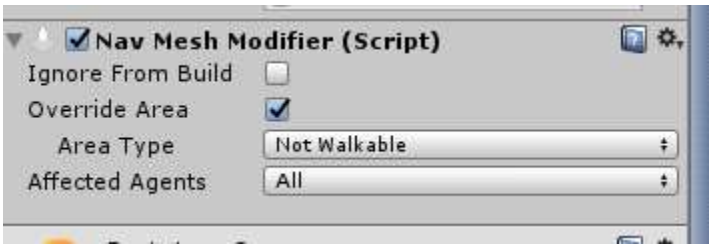
And that is the same in this simple example scene as it is using more complex packages like dungen. So I will break each step down one at a time and explain what is going on.

Step 1 - Create Level Geometry

In the scene view you will see spawn points for level geometry and an AI agent. At the level geometry spawn locations, one of these three blocks will be spawned:

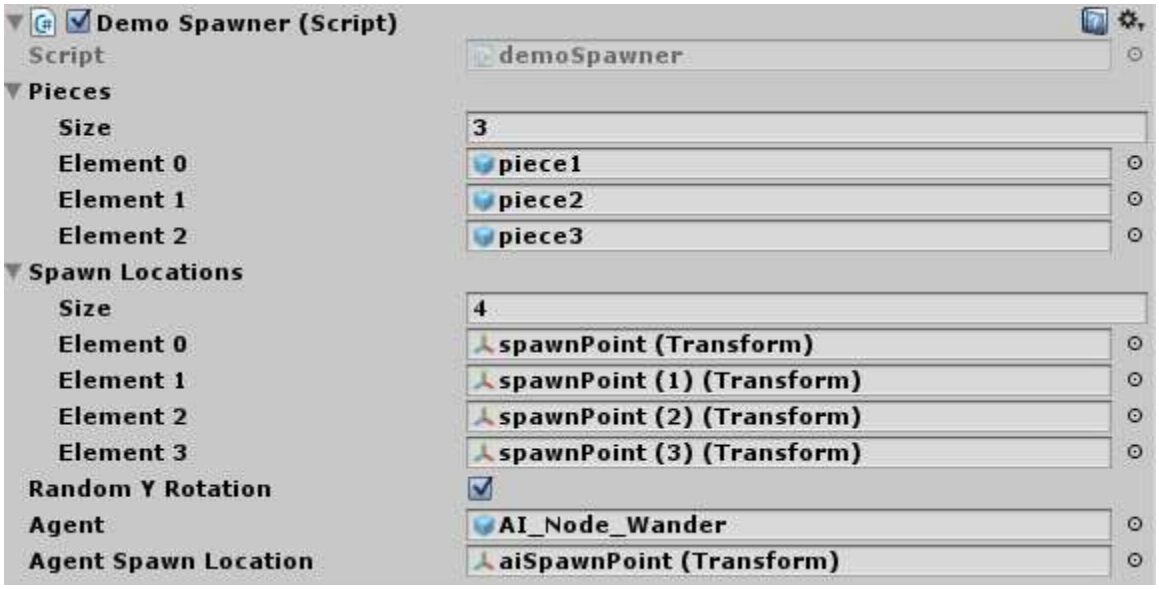


each one of these pieces has a nav mesh modifier script on it:



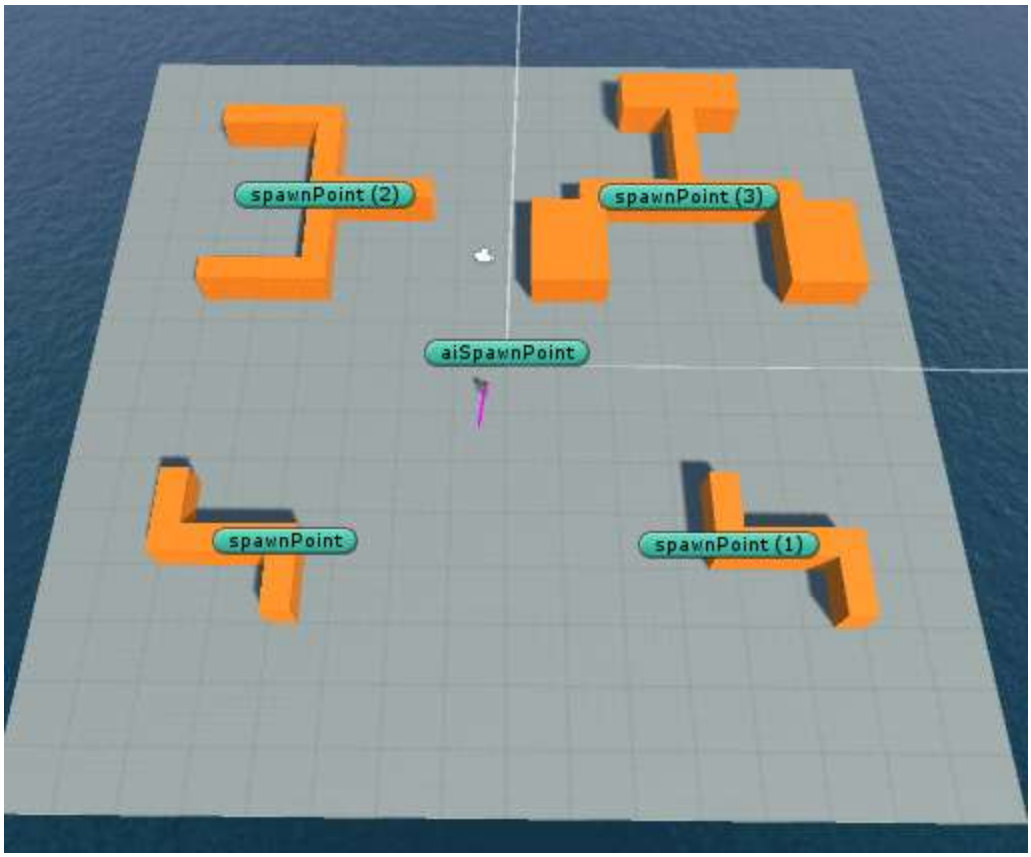
More on this in a bit.

The first script to do anything when you press start is the "node_demoSpawner" script:



notices that the pieces setting contains the blocks pictured above, the spawn locations contain the spawn locations located in the scene and the agent spawn location contains the agent spawn location contained in the scene.

When you press play in this scene, one of those blocks is spawned at each one of those spawn locations with a random y rotation. It looks something like this:



Step 2 - Build navmesh

Unity now uses a component based system to build its navmesh, and its why I have updated node AI to version 3. It is pretty cool, and easy to use in my opinion. So in this scene, notice a gameobject called "NavMesh Surface". This gameobject is required in every scene that uses a navmesh (and therefore every scene that uses node AI) and you can build your navmesh at anytime by pressing the bake button. But for this example, we will not be doing that.

After the demoSpawner script spawns in the level pieces, it sends a message to the navmesh surface to build the navmesh. It builds around the area where the pieces are now located thanks to the modifier script, which also prevents navmesh from being built on top of those pieces.

Step 3 - spawn agent

After the navmesh is built, the agent is spawned in and the scene begins.

Tips on integration with DUNGEN

As of this writing, dungen now has native support for runtime navmesh baking. Information on dungen and navmesh baking can be found in the dungen manual (page 29 as of this writing).

UFPS Setup

Node AI and UFPS are a great combination! For this guide I am using Node AI 3.1, Unity 2017.1, and UFPS 1.7.1 **UFPS IS SOLD SEPERATELY!**

1.Start a new project.

2. Import UFPS.

3.Import the unity navmesh components from

https://github.com/Unity-Technologies/NavMeshComponents

I suggest downloading the zip file, and copying everthing(except the "Tests" folder) from the assets folder into your projects assets folder.

4.Import and setup Node AI. Your Tags, layers and physics should be ok as is.

5. Open up the scene called "**gamePlay**" located in the "scenes" folder.

6.Delete the "**Player_TwinStick**", **"Canvas"** and "**camera**" objects in the scene. Remove the "**LevelControl"** script from the "**_GameControl"** gameobject.

7. Open up the "**aiShooting**" script located in "nodeAI/scripts/aiScripts"

8. replace line 154 with this:

hit.collider.SendMessageUpwards("Damage", new vp_DamageInfo(baseDamage, transform), SendMessageOptions.DontRequireReceiver);

It should look like this:

And line 166 with:

hitColliders[i].SendMessageUpwards("Damage", new vp_DamageInfo(baseDamage, transform), SendMessageOptions.DontRequireReceiver);

9. save the script.

10. Find the ufps addon on scripts located at "nodeAI/scripts/3rdPartyScripts", right click it, then click "show in explorer" (I am not sure if this option is the same on Mac. Basically unzip the zip file in this same folder). Unzip that folder, then close the file explorer window and return to unity.

11. Select all the AI agents in your scene and remove the "**aiHealth**" script.

12. Click "add component", and add the "**node_ai Health UFPS**" script.

13. (optional) With all the ai agents still selected, drag the "**roboRagDoll**" prefab located in the nodeAI/prefabs/demoPrefabs folder on to the "spawn on death" slot located on the "**node_ai Health UFPS**" script.

14. With all the ai agents still selected, click "**detection settings**" on the aiMovment Script, and change "**height offset**" to 0.5.

15. Select the "**HeroHDWeapons**" prefab located in "UFPS/Base/content/Prefabs/Players" and drag it into your scene. I would recommend putting him where the old 3rd person controller was.

16. Click "Add component" select scripts, then "**ufps_reloadLevelOnDeath**". This script will simply restart your scene

when the player dies.

17. Select all the game objects located in the folder "UFPS/Base/Prefabs/Projeciles" and change the "**Damage Mode**" setting of the "**vp_FX Bullet**" script to "**Both**"

18. If you want AI Agents to react to the sound of gun shots (so you may want to deselect "knife attack" and "melee attack" or other silent weapons) click Add Component, scripts then "**gunSound**". Set "Enemy Layers" to only "Enemy". Shot distance is the radius in which this weapon being fired will alert enemies.

19. Thats it! Press play and everything should be working. Drag in some more weapons and ammo, duplicate some enemies and move them around, and have fun :) I recommend saving this scene as something unique, and creating new prefabs based off the agents/player in this scene.

Some tips: You may want to adjust the damage paramater on the ai Shooting scripts. 1.0 is a little high, something like .2 is more suitable. Likewise, you may wan to drop there HP to something like 5.

With UFPS, you may want to increase the stopping distance under distance settings on the aiMovement script. Otherwise the agents can push your character around. Also, as you increase the stopping distance, if you are using melee you will also want to increase the melee attack radius on the Node AI shooting script. trying changing both stop distances to 2, melee distance to 4, and melee attack radius to 3.

Ifyou want bullet holes to show up in the level, highlight all the children of the "_Level" gameObject in the scene, click add component then "scripts" then "**vp_Surface Identifier**". With all the level objects still selected, click "surface type" and then select "default" from the assets tab. Now your gunfire should leave holes in the level.

Credits

Special thanks to Miodrag "j0linar" Sejic for providing the melee animation.
His website can be found at http://j0linar.github.io/

-Thanks again for purchasing Node A.I.!