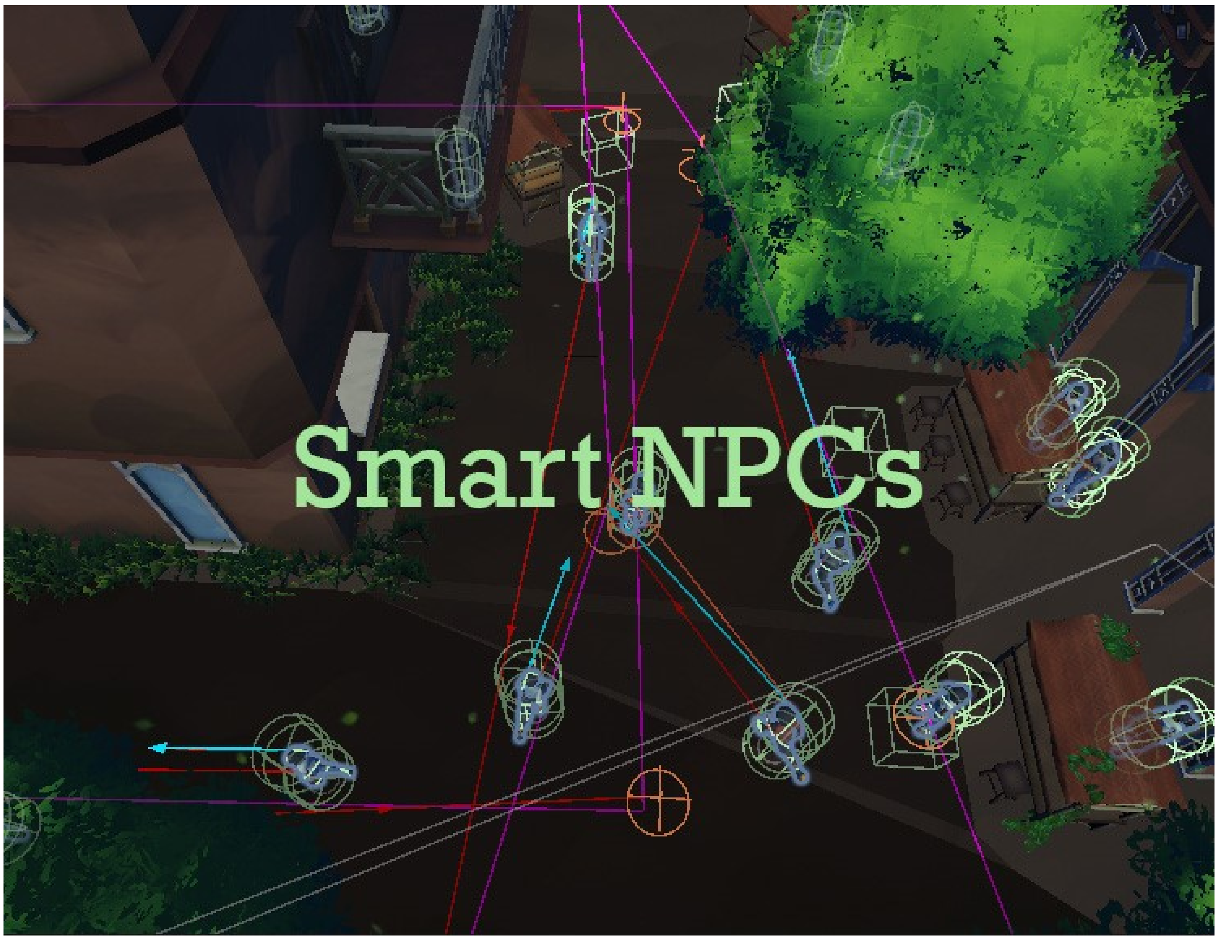


Version 0.7.1

Last updated 2nd of November 2024

Thank you kindly for purchasing this asset pack. Each purchase helps me to further develop this asset pack in order to make it the perfect solution for every game development company.

This tool is created by Ravi Bechoe. For issues or business queries you can reach out to me by mail at rbechoe@outlook.com, please include in your subject at least the following "Smart NPCs – [purpose]" e.g. "Smart NPCs - collaboration".

There is a roadmap for this asset pack. You can view it on the asset store at "Technical Details" or in this document.

Table of Contents

Important information

This asset pack works with Tooltips and has a shortcut key for the tool itself. You can open the base tool with CTRL + T.

In every scene where you would like this asset pack you must bake the navmesh on the surface of where the agents should navigate on. If you have your own navigation solution that is of course fine as well.

This entire pack is modular, nothing in this pack is mandatory or a must, you can easily swap systems with your own systems in order to get the best out of it.

Most of the fields (if not all) within the tool contain tooltips. You can see the tooltip when you hover over the field.

This asset pack assumes that you have a general understanding of Unity the game engine. Upon expanding on the tool this asset pack assumes that you have a general understanding of software development and/or system design.

This asset pack has a coding standard and folder structure standard.

This asset pack includes source code to help you customize it to your needs.

The quick start guide will help you getting started with the asset pack, in order to expand the other chapters can be mandatory depending on your technical level of knowledge.

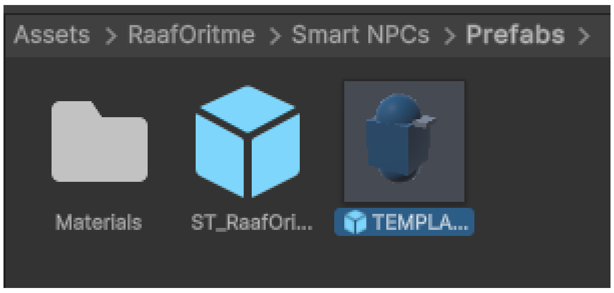
While this asset pack serves as a strong baseline for NPCs within your game, it does require tweaking and fine-tuning in order to achieve what your project requires.

All blue marked words in the guide sections are literal names within the asset pack.

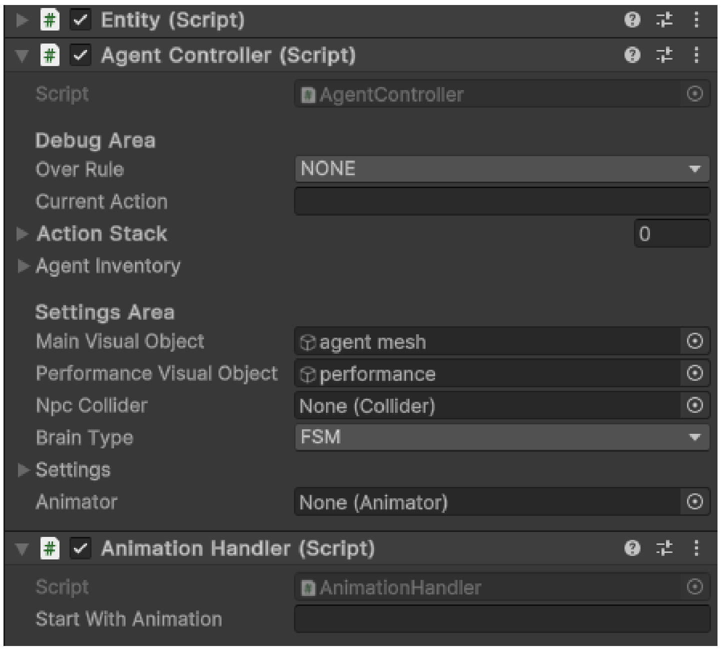
Quick Start Guide

When your goal is to quickly see behaviour within your project in order to make everything feel more alive then this section will have you sorted in under 10 minutes! In order to remain fast, please ensure to use the FSM Brain Type. The GOAP is a lot more advanced and requires more knowledge about this tool. Please refer to the main systems section of this documentation.

1. Go to the scene you want to implement the NPCs in. Go to the prefabs folder and drag the ST_RaafOritme in your scene. In the same folder you will also find the TEMPLATE_NPC prefab. You can edit this prefab by replacing the agent mesh and performance objects with what you have in your project, or you can leave it as is. You can ignore the settings within all components.

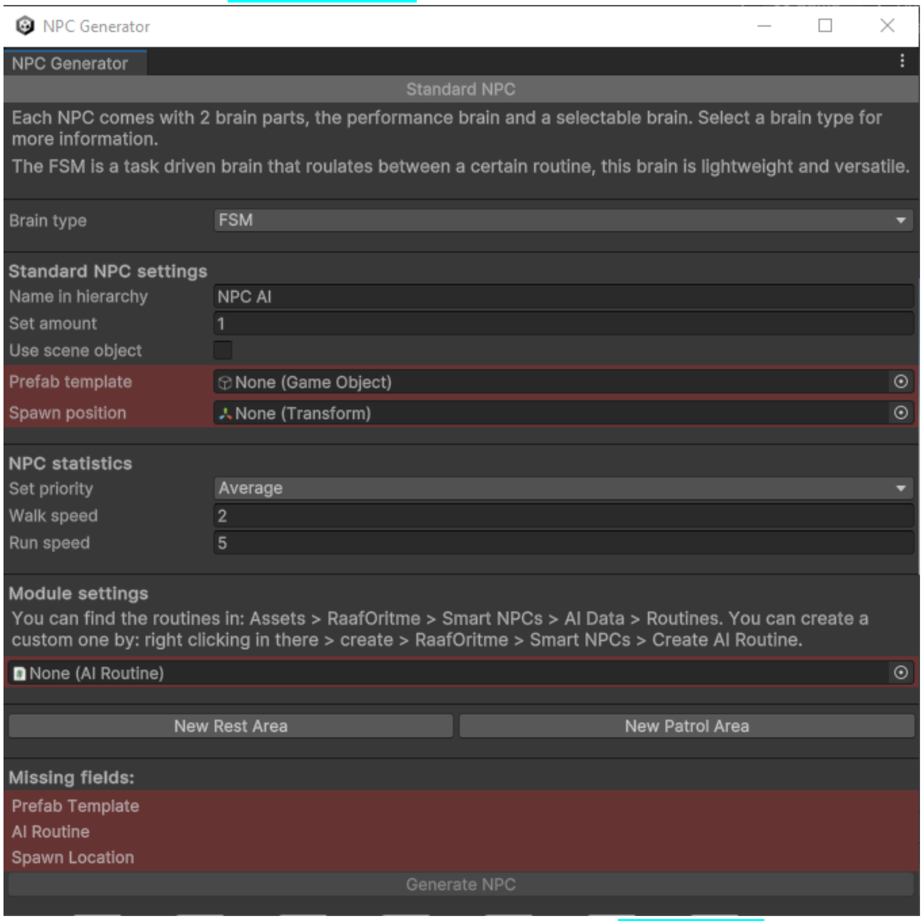
   The only thing you don't have to ignore is if your mesh has an animator with animations. In the Agent Controller you can set the Animator field with the animator. Please go to the animations section if you would like to learn more about how you can trigger the right animations. Don't forget to set in the Settings Area of the agent controller the Main Visual Object and Performance Visual Object. You can drag the new game objects in those fields.

   Entity (Script)
   Agent Controller (Script)
   Script: AgentController
   Debug Area
   Over Rule: NONE
   Current Action:
   Action Stack: 0
   Agent Inventory
   Settings Area
   Main Visual Object: agent mesh
   Performance Visual Object: performance
   Npc Collider: None (Collider)
   Brain Type: FSM
   Settings
   Animator: None (Animator)
   Animation Handler (Script)
   Script: AnimationHandler
   Start With Animation:

2. After preparing your basics you can open the tool with CTRL + T, or alternatively by going to "Tools > RaafOritme > SmartNPCs > NPC Generator" in the top bar.

This will open the NPC Generator, the main tool of this asset pack, for you.



Each NPC comes with 2 brain parts, the performance brain and a selectable brain. Select a brain type for more information.
The FSM is a task driven brain that roulates between a certain routine, this brain is lightweight and versatile.

Brain type: FSM

Standard NPC settings
Name in hierarchy: NPC AI
Set amount: 1
Use scene object:
Prefab template: None (Game Object)
Spawn position: None (Transform)

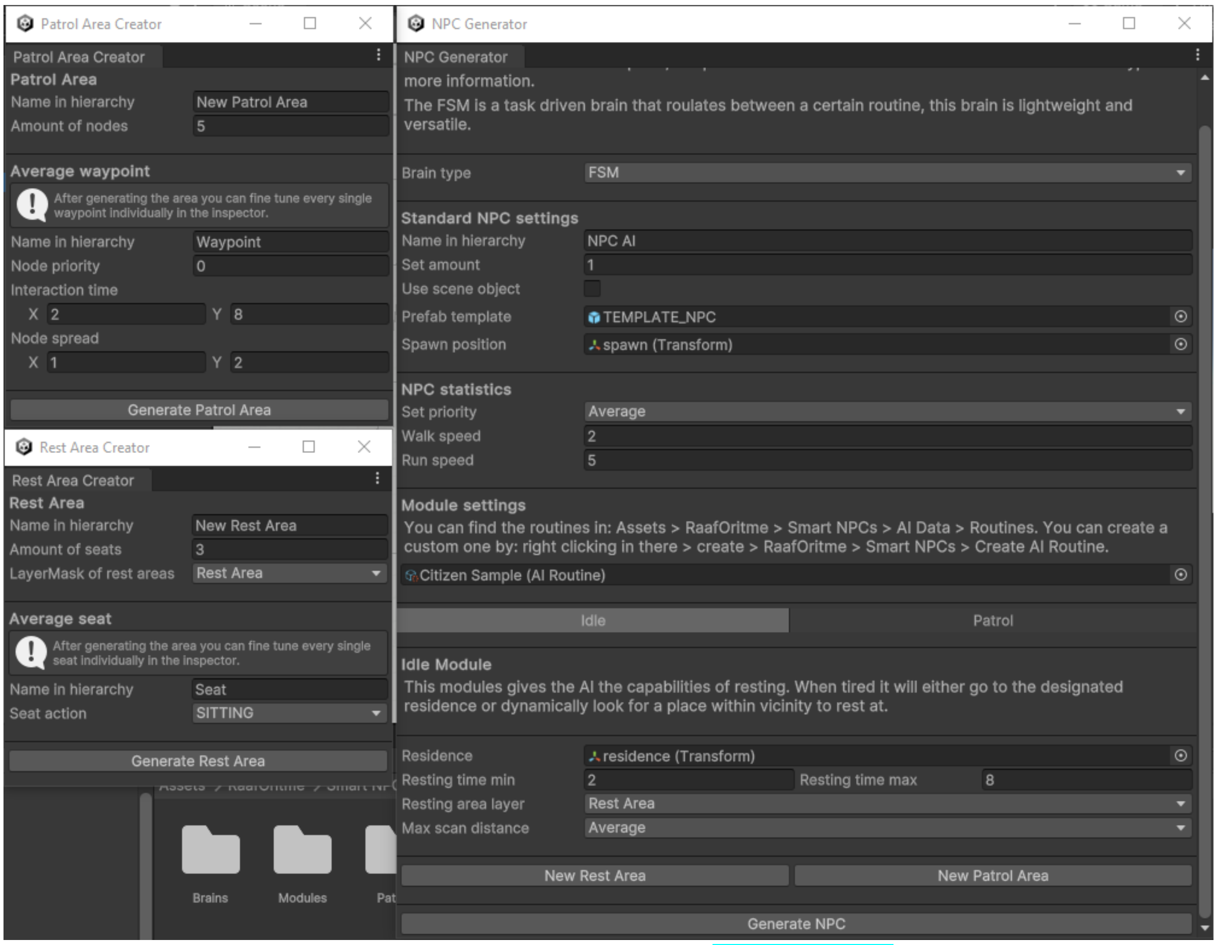
NPC statistics
Set priority: Average
Walk speed: 2
Run speed: 5

Module settings
You can find the routines in: Assets > RaafOritme > Smart NPCs > AI Data > Routines. You can create a custom one by: right clicking in there > create > RaafOritme > Smart NPCs > Create AI Routine.
None (AI Routine)

Missing fields:
Prefab Template
AI Routine
Spawn Location

3. Each field show a tooltip (hover over for example Set amount to learn what it is used for). Fill in at least all the red marked fields, these are mandatory. The default settings are based on what generally is used, feel free to adjust them accordingly.
4. Modules are explained at the modules part of this documentation, in a nutshell it encapsulates what type of behaviour your NPC should have. By default the asset pack comes with a few predefined AI Routines to choose from. Each AI routine is a mixture of Module combinations. These all have individual settings as well.
5. In order to properly assign the Patrol and Idle modules you can use the buttons at the bottom names New Rest Area and New Patrol Area. Click on them to open mini tools that will guide you through the creation of those. After that hit generate and assign them to the NPC Generator. Pro tip: in the Hierarchy you can customize the newly generated rest, and patrol area.

6. Once you have everything filled in it might look something like this.



Patrol Area Creator

Patrol Area

Name in hierarchy	New Patrol Area
Amount of nodes	5

Average waypoint

After generating the area you can fine tune every single waypoint individually in the inspector.

Name in hierarchy	Waypoint
Node priority	0
Interaction time	X 2 Y 8
Node spread	X 1 Y 2

Generate Patrol Area

Rest Area Creator

Rest Area

Name in hierarchy	New Rest Area
Amount of seats	3
LayerMask of rest areas	Rest Area

Average seat

After generating the area you can fine tune every single seat individually in the inspector.

Name in hierarchy	Seat
Seat action	SITTING

Generate Rest Area

NPC Generator

more information.

The FSM is a task driven brain that roulates between a certain routine, this brain is lightweight and versatile.

Brain type	FSM

Standard NPC settings

Name in hierarchy	NPC AI
Set amount	1
Use scene object	
Prefab template	TEMPLATE_NPC
Spawn position	spawn (Transform)

NPC statistics

Set priority	Average
Walk speed	2
Run speed	5

Module settings

You can find the routines in: Assets > RaafOritme > Smart NPCs > AI Data > Routines. You can create a custom one by: right clicking in there > create > RaafOritme > Smart NPCs > Create AI Routine.

Citizen Sample (AI Routine)

Idle | Patrol

Idle Module

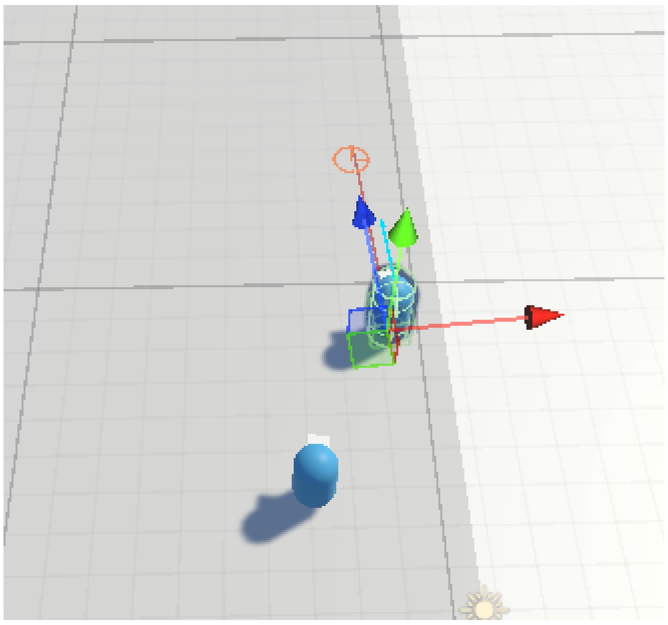
This modules gives the AI the capabilities of resting. When tired it will either go to the designated residence or dynamically look for a place within vicinity to rest at.

Residence	residence (Transform)		
Resting time min	2	Resting time max	8
Resting area layer	Rest Area		
Max scan distance	Average		

New Rest Area | New Patrol Area

Generate NPC

7. If you are happy with the results, please make sure to hit Generate NPC before closing the window.
8. When you have done everything properly you can simply hit the play button and see the magic unfold! **Congratulations, you are done!**



9. Each generated NPC can be customized further by clicking on them in the hierarchy. The inspector shows plenty of settings for you to mess around with. Refer to section fields of this documentation to understand what each setting encapsulates.

Advanced Start Guide

This section will teach you how to properly adjust the values and how to make the asset pack more your own. This is more orientated towards game designers. You should visit the sections covering the systems if you want to learn more about the technical details.

This asset pack contains 3 main systems powering your NPC. This will be referred to as “brain”. So each brain type serves a different purpose, based on the different results you can achieve with it. In order to understand the brains you need to understand the Mother AI.

Mother AI

The whole purpose of this brain is to ensure that the system load will be non-existent. It works with a 3 circle principle. The inner circle has for example a radius of 10 meters. The middle one has a radius of 20, and the outer one has a radius of 50 meters. The centre point of the player is the centre of all the circles. The Mother AI switches between main brain (FSM or GOAP), performance, and inactive. You don’t need to simulate what you don’t see or what you don’t immediately see. The ST_RaafOritme contains the Mother AI in which you can set the distances.

Performance Brain

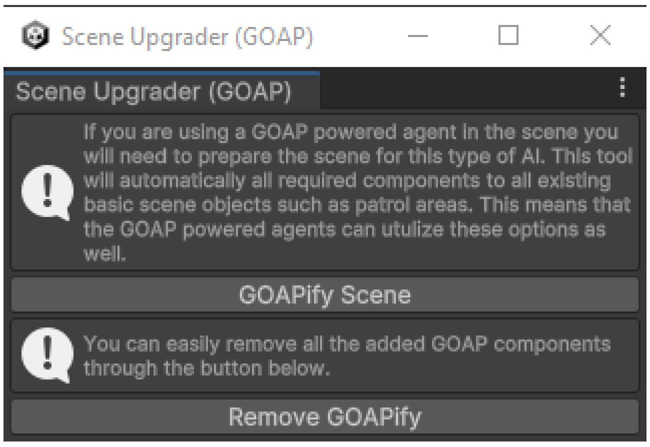
You can choose between FSM or GOAP, each agent automatically comes with the performance part. The performance brain disables animations, sounds, the main mesh (hence the performance mesh), and heavy calculations. It estimates what it should do and where it should go based on a patrol area.

FSM Brain (lightweight)

FSM stands for Finite State Machine. It is a system that rotates between a set of actions and resets when done in order to infinitely rotate between the actions. Each action requires a Module and a singular module comes with a lot settings for fine tuning and customization. If you want your NPC to explore your game by walking around and doing various objectives then you will need the Patrol Module. The benefit of this system is that it is only “heavy” during start-up or when it needs to figure out an action. This designed system also leaves opportunities for dynamic reactions. This means that the NPC can be interrupted as long as it meets the conditions before it resumes what it was doing. Let’s say the same NPC has the Dialogue Module then it means that you can talk to it. So it stops patrolling to entertain you and after the conversation it resumes patrolling.

GOAP Brain (hero NPCs)

GOAP stands for Goal Orientated Action Planner. This system is a lot more complex than the FSM Brain and it utilizes a lot different sub-systems in order to achieve unique results. Fine tuning your NPCs will take significantly more time, but it allows you to create truly unique NPCs that feel alive behaviour wise. As the name suggests, it plans what it will do by scanning the environment for all possible actions. It has needs and emotions which give a certain bias to some actions. Let’s say that your NPC is tired from all the patrolling it has done and is now scanning the environment for what it can do next and it finds a restaurant to relax at, it will most certainly go there until it feels well rested. The NPC also keeps track of what it has done so far and how it impacted its own mental and physical well-being. Let’s say it did a patrol route, but was interrupted by an aggressive player that attacked it, but didn’t manage to kill it. The NPC will remember it. The NPC will have a negative bias due to this towards patrolling, where it will try to choose things different from patrolling, since it has bad memories about that activity. When you use the GOAP you also need to use the tool GOAPify Scene, you can find this in Tools > RaafOritme > Smart NPCs > GOAPify Scene. Since each agent now requires new data in order to figure out where it can do what, your entire scene needs to be upgraded. This tool does that automatically for you for out-of-the-box assets. Through the same tool you can also undo it.


Scene Upgrader (GOAP)

Scene Upgrader (GOAP)

If you are using a GOAP powered agent in the scene you will need to prepare the scene for this type of AI. This tool will automatically all required components to all existing basic scene objects such as patrol areas. This means that the GOAP powered agents can utulize these options as well.

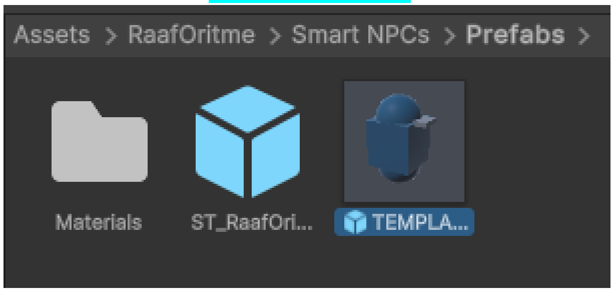
GOAPify Scene

You can easily remove all the added GOAP components through the button below.

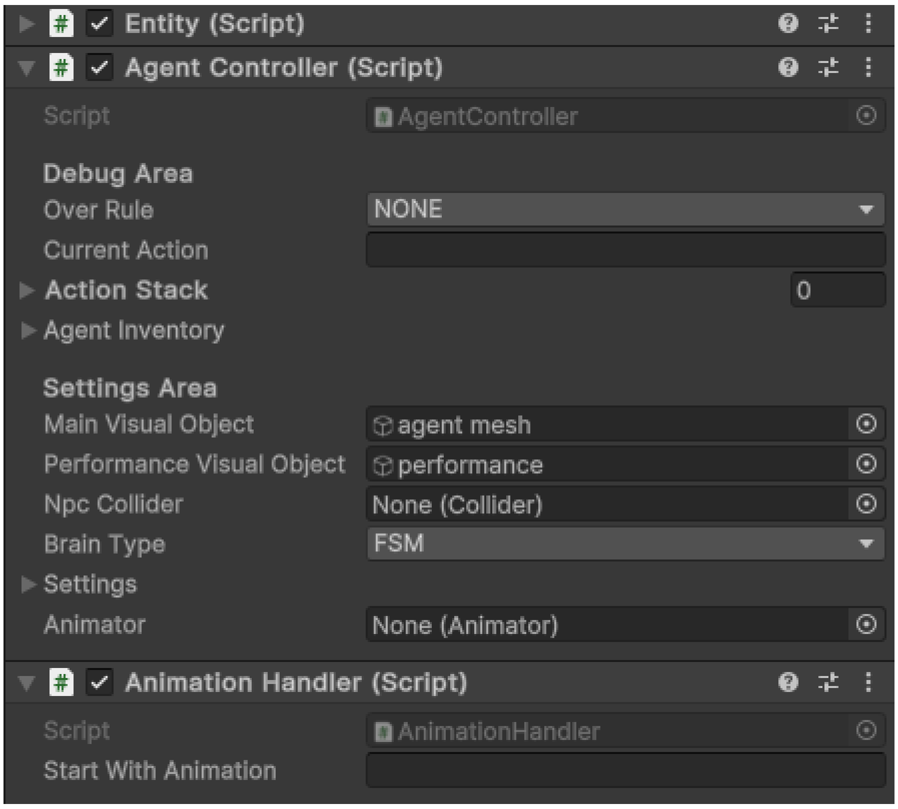
Remove GOAPify

Alright with all of that out of the way, let's create a new NPC through the tool. Each step will contain more information compared to the quick start guide so that <u>you can customize as much as you want without touching a single line of code</u>!

1. Go to the prefabs folder of this asset pack (RaafOritme > Smart NPCs > Prefabs) and open the prefab TEMPLATE_NPC.



In here you can customize the performance and agent mesh. The agent mesh should be your character with all the juicy details. The performance object should be something extremely low-poly and simple. For example a quad with a 2D image of your character, or a low poly silhouette.



Entity (Script)

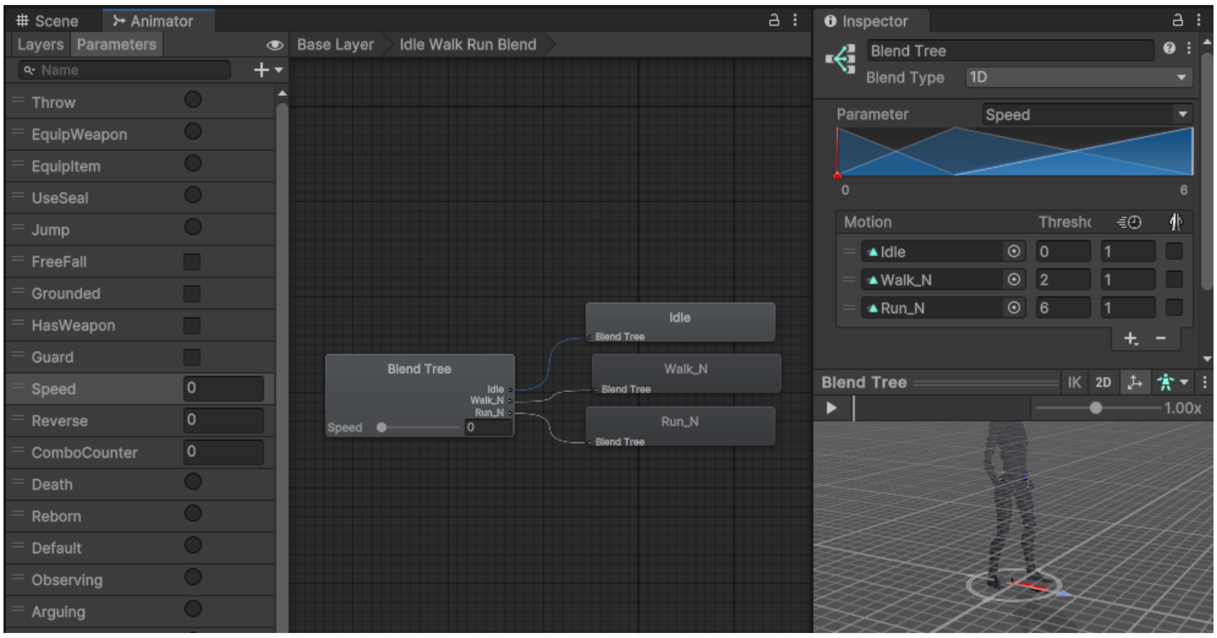
Agent Controller (Script)

Script	AgentController
Debug Area	
Over Rule	NONE
Current Action	
Action Stack	0
Agent Inventory	
Settings Area	
Main Visual Object	agent mesh
Performance Visual Object	performance
Npc Collider	None (Collider)
Brain Type	FSM
Settings	
Animator	None (Animator)

Animation Handler (Script)

Script	AnimationHandler
Start With Animation	

You can ignore all other settings for now. These are used to further customize or fine tune your NPC once it has been generated. If your character model has animations with an animator than you can assign the animator to the Animator in the agent controller. This will allow you to utilize animations. Out-of-the-box you will have movement based animations. The asset pack automatically updates the Speed float parameter for your animator. In your animator you should assign the walking, running, and idling based on this parameter for the quickest results. In the example below I used a Blend Tree for this. Keep in mind that this is not included in the asset pack.



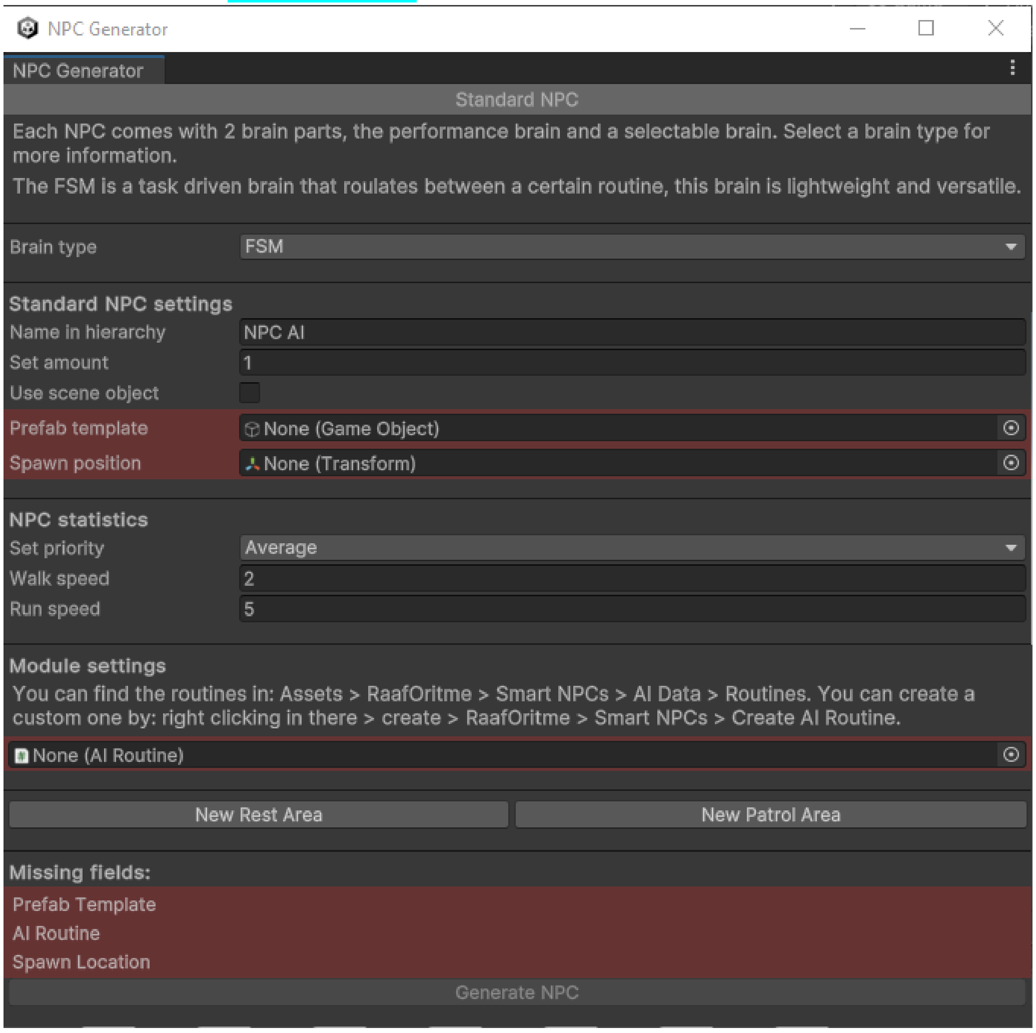
Parameters	
Throw	
EquipWeapon	
EquipItem	
UseSeal	
Jump	
FreeFall	
Grounded	
HasWeapon	
Guard	
Speed	0
Reverse	0
ComboCounter	0
Death	
Reborn	
Default	
Observing	
Arguing	

Blend Tree
Blend Type 1D
Parameter Speed

Motion	Threshold	
Idle	0	1
Walk_N	2	1
Run_N	6	1

2. After preparing your basics you can open the tool with CTRL + T, or alternatively by going to “Tools > RaafOritme > SmartNPCs > NPC Generator” in the top bar.

This will open the NPC Generator, the main tool of this asset pack, for you.



NPC Generator

NPC Generator

Standard NPC

Each NPC comes with 2 brain parts, the performance brain and a selectable brain. Select a brain type for more information.
The FSM is a task driven brain that roulates between a certain routine, this brain is lightweight and versatile.

Brain type: FSM

Standard NPC settings

Name in hierarchy: NPC AI
Set amount: 1
Use scene object: ☐
Prefab template: None (Game Object)
Spawn position: None (Transform)

NPC statistics

Set priority: Average
Walk speed: 2
Run speed: 5

Module settings

You can find the routines in: Assets > RaafOritme > Smart NPCs > AI Data > Routines. You can create a custom one by: right clicking in there > create > RaafOritme > Smart NPCs > Create AI Routine.

None (AI Routine)

New Rest Area | New Patrol Area

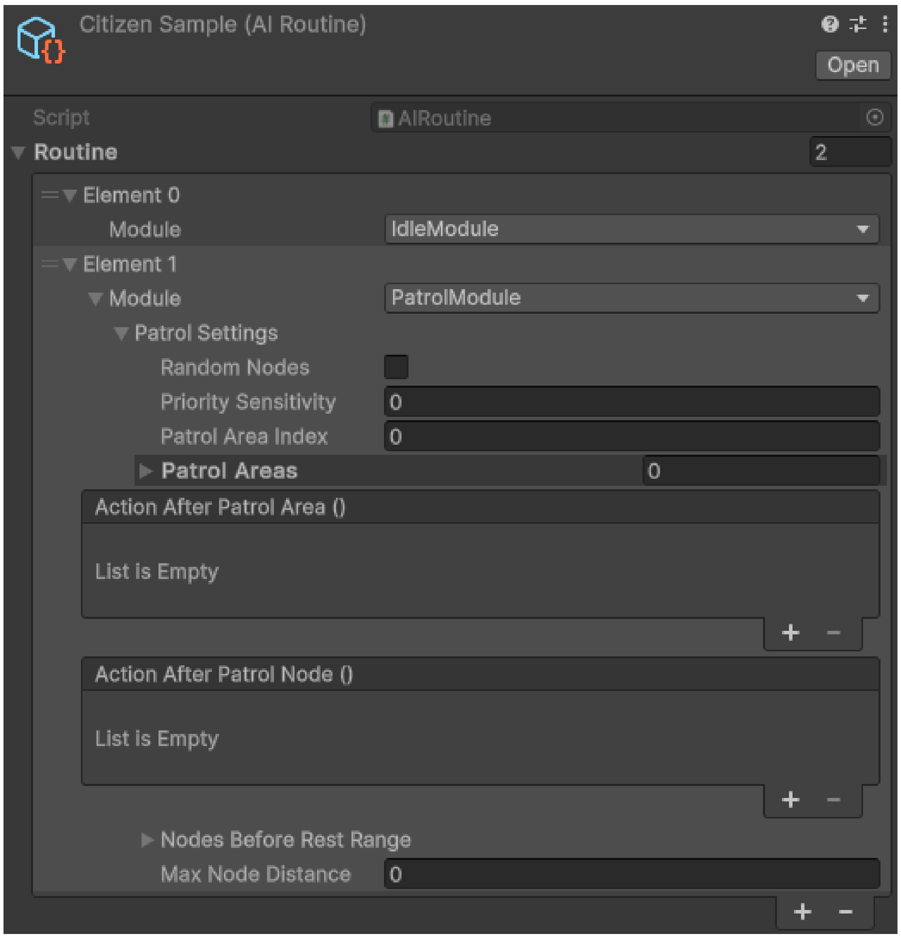
Missing fields:

Prefab Template
AI Routine
Spawn Location

Generate NPC

3. At first this might feel intimidating, but keep in mind that each field contains a tool-tip to make things as self-explanatory as possible. The Brain type GOAP and FSM both have the same settings. If you decide to use GOAP, please don't forget to use the GOAPify tool once you are done with your scene. Each field show a tooltip (hover over for example Set amount to learn what it is used for). Fill in at least all the red marked fields, these are mandatory. The default settings are based on what generally is used, feel free to adjust them accordingly. In your scene you can create an empty game object and drag it to a place where you want your NPC to spawn at. You can manually place the created NPC where-ever you want, or maybe there is a special building where they all come from. The prefab template is the template that you adjusted earlier on.
If you use the setting Use scene object then it means that all logic will be applied on an existing object in your scene. This can be especially useful if you already have an existing scene with all the characters at specific locations.
4. Modules are explained at the modules part of this documentation, in a nutshell it encapsulates what type of behaviour your NPC should have. By default the asset pack comes with a few predefined AI Routines to choose from. Each AI routine is a mixture of Module combinations. These all have individual settings as well. For a GOAP powered brain it is important to add a routine that contains all the modules you want it to utilize. It completely ignores the order of the modules, so you want each module to be there once. Leaving a module out essentially means leaving a sense out. If it has no patrol module

then it will have no means of navigation.
You can also create your own AI Routine. It is recommended to do this in the project folder RaafOritme > Smart NPCs > AI Data > Routines. Right click on there and select RaafOritme > Smart NPCs > Create AI Routine.



Citizen Sample (AI Routine)

Open

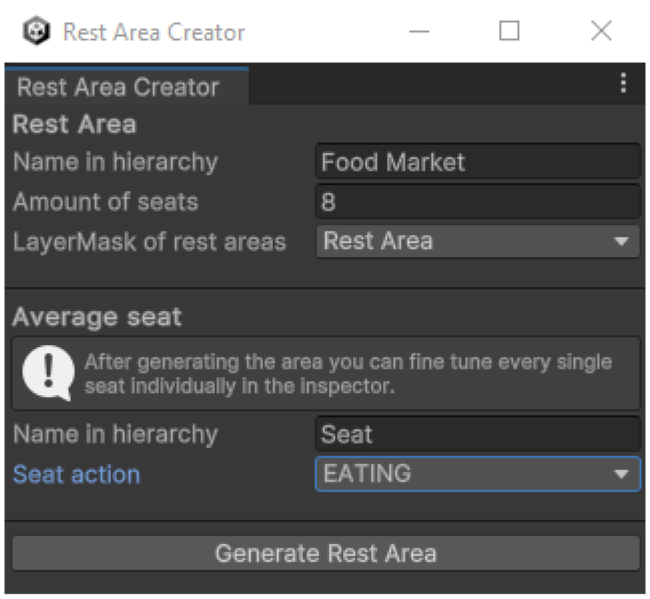
Script: AIRoutine

Routine: 2

- Element 0
  - Module: IdleModule
- Element 1
  - Module: PatrolModule
    - Patrol Settings
      - Random Nodes: ☐
      - Priority Sensitivity: 0
      - Patrol Area Index: 0
      - Patrol Areas: 0
    - Action After Patrol Area ()
      - List is Empty
    - Action After Patrol Node ()
      - List is Empty
    - Nodes Before Rest Range
    - Max Node Distance: 0

You can add as many modules as you want in any order that you want, at least for the FSM brain. Each module comes with settings that will be overwritten by the tool itself. You can fine-tune this in the generated NPC. Why would you add for example 2x idle, and 2x patrol in any order? Maybe you want the NPC to patrol in different areas with different settings when you fine tune it. **This is a scriptable object, so changes made during run-time are kept!**

5. Back to the main tool. In order to properly assign the Patrol and Idle modules you can use the buttons at the bottom names New Rest Area and New Patrol Area. Click on them to open mini tools that will guide you through the creation of those. After that hit generate and assign them to the NPC Generator. Pro tip: in the Hierarchy you can customize the newly generated rest, and patrol area. The steps 6 and 7 do a deep dive on these 2 mini tools.
6. A rest area is used for the NPC to relax at and to give your environment a more dynamic look and feel. Maybe you have some food stalls, restaurants, sleeping places, etc. Mark all of them as a rest area and the NPC will utilize them! Keep in mind that it always requires the Rest Area layer mask. Create one if you don't have it already.


Rest Area Creator

Rest Area

Name in hierarchy: Food Market

Amount of seats: 8

LayerMask of rest areas: Rest Area

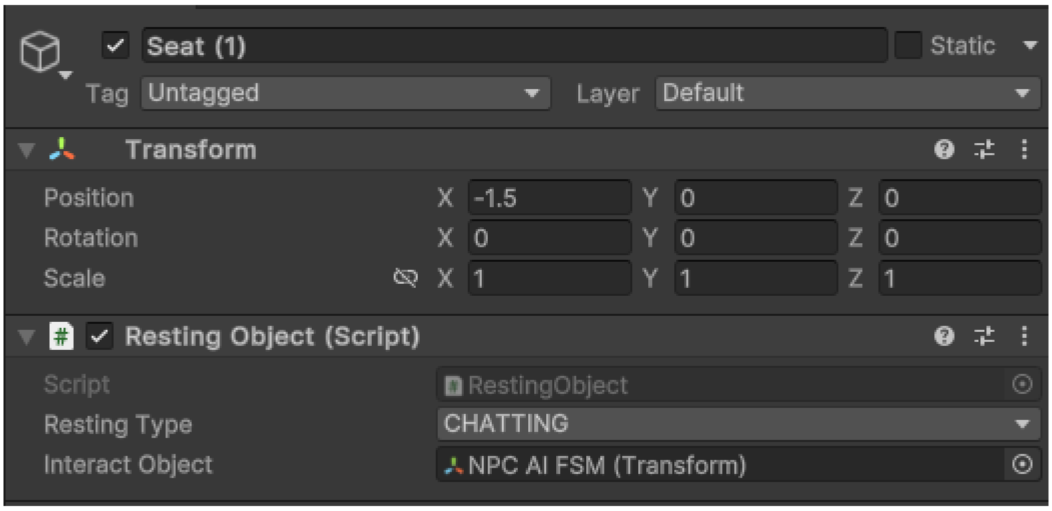
Average seat

After generating the area you can fine tune every single seat individually in the inspector.

Name in hierarchy: Seat

Seat action: EATING

Each seat has an unique action you can assign to it. This action directly translates to animations that are fired based on a trigger. Each animation name is translated to the Animator name which in this case would be “Eating”.
After hitting generate you will notice that you will have made a game object with various children. The entrance serves as literally the entrance. This is where all NPCs will be coming from. This asset pack doesn’t support multiple entrances currently. You can easily expand on it through the RestingArea script. In the method BookSeat you can request a parameter that is the transform of the agent and then see which entrance is the most nearby and return that.
Each Seat can only be occupied by 1 NPC. We don’t want the classic bug where 2 NPCs are glitching in the lap of one another. Here comes the magic where you can fine tune 2 things with major outcomes, the resting type and the Interact Object. Let’s say that you have a vendor at a stall, the object would be the vendor. The outcome? Your NPC always looks at the vendor when performing the selected Resting Type. Resting essentially means what type of animation should be used.


Seat (1) — Static

Tag: Untagged — Layer: Default

Transform

	X	Y	Z
Position	-1.5	0	0
Rotation	0	0	0
Scale	1	1	1

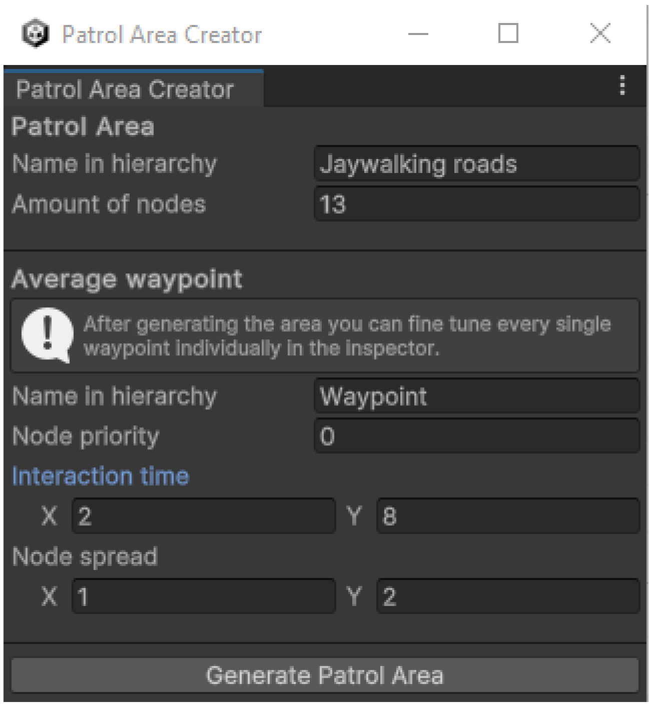
Resting Object (Script)

Script: RestingObject

Resting Type: CHATTING

Interact Object: NPC AI FSM (Transform)

7. Creating a patrol area is mandatory if you want to give your NPCs somewhere to walk to, at least for the FSM brains. The GOAPs will figure something out if you have created your scene properly. Opening the Patrol Area Creator through the main tool will give you an interface with a couple of settings.


Patrol Area Creator

Patrol Area

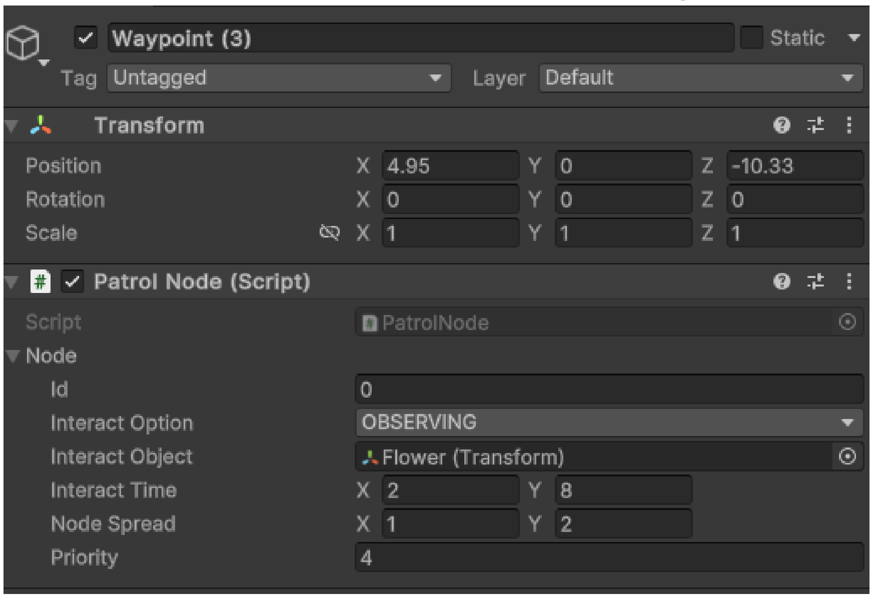
Name in hierarchy	Jaywalking roads
Amount of nodes	13

Average waypoint

After generating the area you can fine tune every single waypoint individually in the inspector.

Name in hierarchy	Waypoint
Node priority	0
Interaction time	X 2 Y 8
Node spread	X 1 Y 2

The nodes are the point that the NPC will be walking to. A priority decides how high the chances are for a NPC to go to a certain node more often. The interaction time indicates how much time a NPC should spend at a node with an action tied to it. The node spread decides how close in range the NPC should roughly be before it has reached its destination. Make sure that it is always at least 1 to remain reachable.
Clicking on your generated patrol area will show the route in the scene. The Patrol Area component has a Validate button. This button will check for you if the route is completely reachable and not obstructed. The performance brain will basically navigate the magenta lines based on hard calculations. So when it switches back to GOAP or FSM it might get stuck in a building or something else if it is within that path.
Each waypoint has its own settings. You can give it an Interact Option and Interact object similar to the seats. This will decide what animation it should perform and where it should look at. This can also be as simple as observing a flower on the ground.


Waypoint (3) — Static
Tag Untagged — Layer Default

Transform

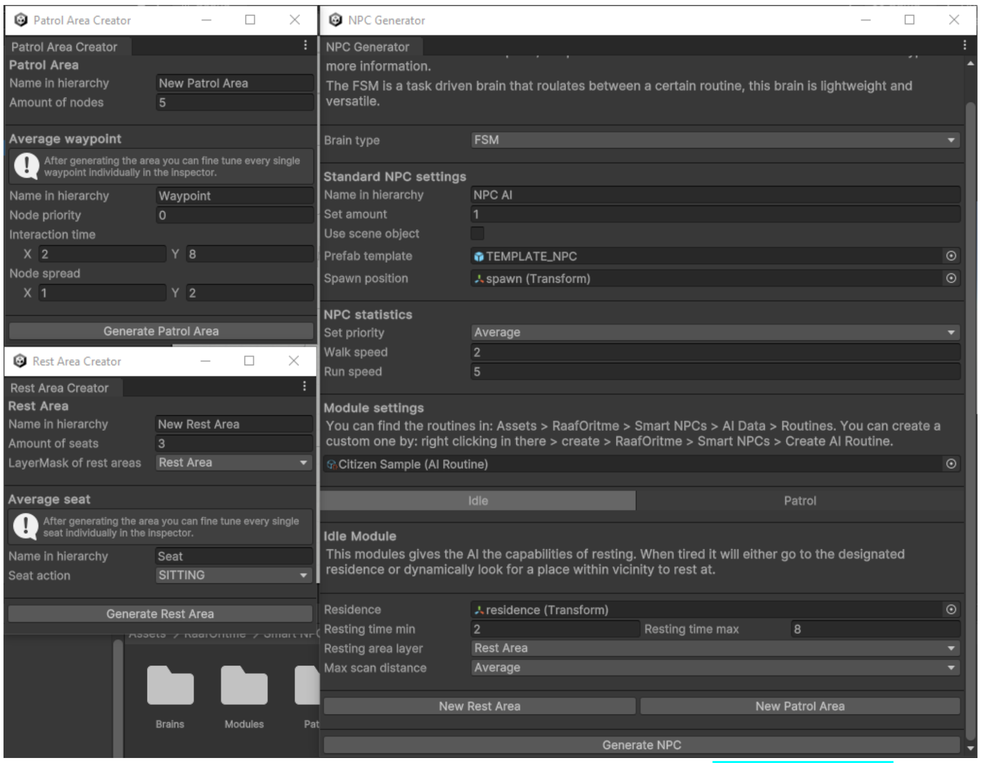
	X	Y	Z
Position	4.95	0	-10.33
Rotation	0	0	0
Scale	1	1	1

Patrol Node (Script)

Script	PatrolNode
Node	
Id	0
Interact Option	OBSERVING
Interact Object	Flower (Transform)
Interact Time	X 2 Y 8
Node Spread	X 1 Y 2
Priority	4

Each waypoint can be made as unique or as simple as you want. When the option for interact is set to none it means that the NPC will ignore the time and just move from Node A to Node B.

8. After having created your rest areas and patrol areas make sure to assign the right parameters to the fields for these corresponding modules. Set the resting area layer in the rest module and assign a patrol area in the patrol module. These settings are mandatory. You can open each module by clicking on the buttons with the names. These serve as tabs that you can collapse when clicking on it again.
9. Once you have everything filled in it might look something like this.



Patrol Area Creator

Patrol Area
Name in hierarchy: New Patrol Area
Amount of nodes: 5

Average waypoint
After generating the area you can fine tune every single waypoint individually in the inspector.
Name in hierarchy: Waypoint
Node priority: 0
Interaction time: X 2 Y 8
Node spread: X 1 Y 2

Generate Patrol Area

Rest Area Creator

Rest Area
Name in hierarchy: New Rest Area
Amount of seats: 3
LayerMask of rest areas: Rest Area

Average seat
After generating the area you can fine tune every single seat individually in the inspector.
Name in hierarchy: Seat
Seat action: SITTING

Generate Rest Area

NPC Generator

more information.
The FSM is a task driven brain that roulates between a certain routine, this brain is lightweight and versatile.

Brain type: FSM

Standard NPC settings
Name in hierarchy: NPC AI
Set amount: 1
Use scene object:
Prefab template: TEMPLATE_NPC
Spawn position: spawn (Transform)

NPC statistics
Set priority: Average
Walk speed: 2
Run speed: 5

Module settings
You can find the routines in: Assets > RaafOritme > Smart NPCs > AI Data > Routines. You can create a custom one by: right clicking in there > create > RaafOritme > Smart NPCs > Create AI Routine.
Citizen Sample (AI Routine)

Idle | Patrol

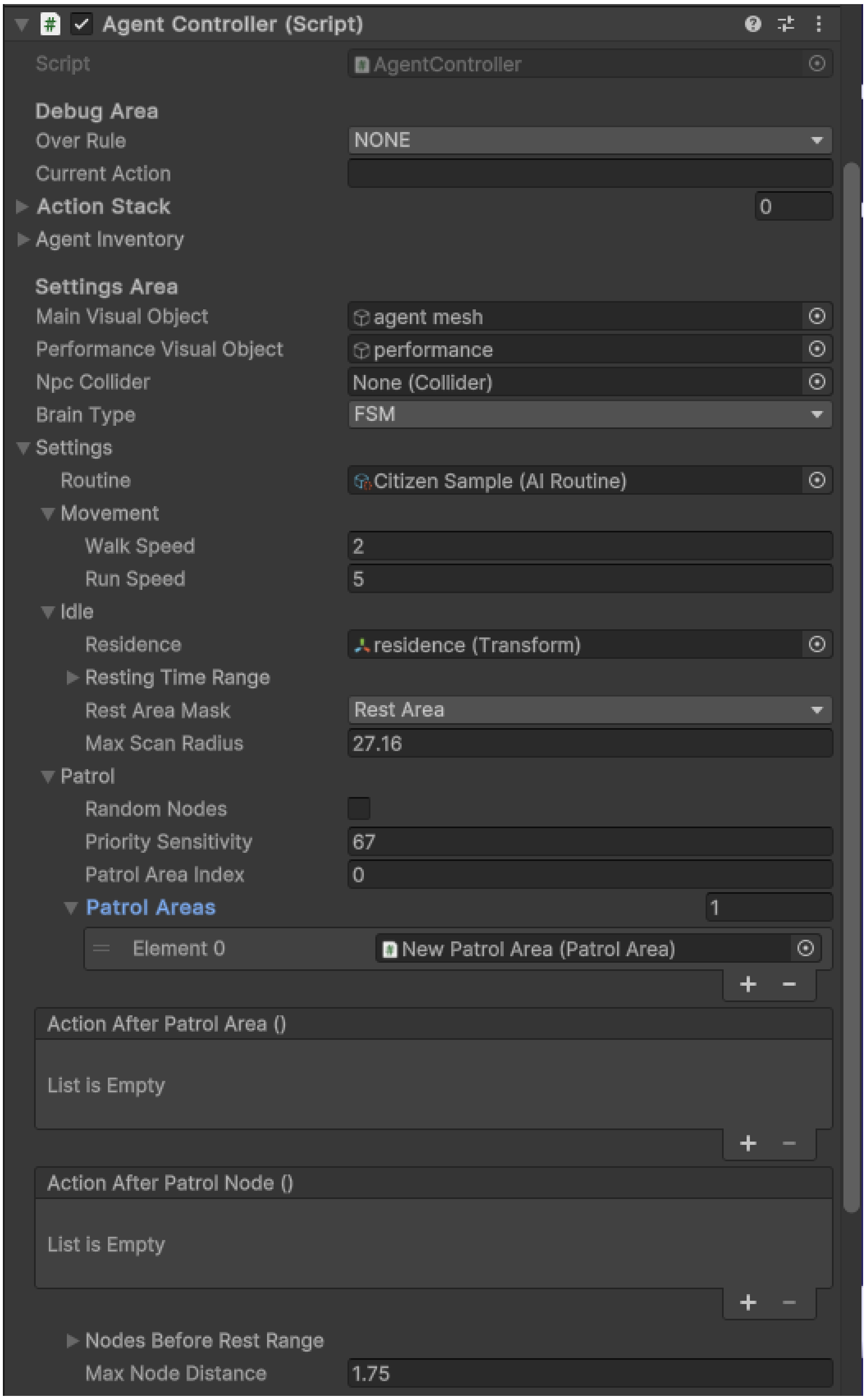
Idle Module
This modules gives the AI the capabilities of resting. When tired it will either go to the designated residence or dynamically look for a place within vicinity to rest at.

Residence: residence (Transform)
Resting time min: 2 Resting time max: 8
Resting area layer: Rest Area
Max scan distance: Average

New Rest Area | New Patrol Area

Generate NPC

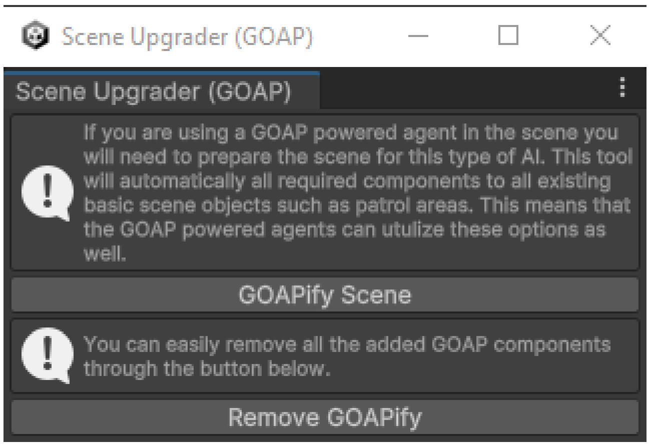
10. If you are happy with the results, please make sure to hit Generate NPC before closing the window. If you have created multiple NPCs then you can further fine tune them by individually selecting them. Go to the Settings Area in the Agent Controller and change things as you wish. You can of course add multiple Patrol Areas to a single NPC so it can choose a new area to patrol once it is done.



Agent Controller (Script)

Script	AgentController
Debug Area	
Over Rule	NONE
Current Action	
Action Stack	0
Agent Inventory	
Settings Area	
Main Visual Object	agent mesh
Performance Visual Object	performance
Npc Collider	None (Collider)
Brain Type	FSM
Settings	
Routine	Citizen Sample (AI Routine)
Movement	
Walk Speed	2
Run Speed	5
Idle	
Residence	residence (Transform)
Resting Time Range	
Rest Area Mask	Rest Area
Max Scan Radius	27.16
Patrol	
Random Nodes	☐
Priority Sensitivity	67
Patrol Area Index	0
Patrol Areas	1
Element 0	New Patrol Area (Patrol Area)
Action After Patrol Area ()	List is Empty
Action After Patrol Node ()	List is Empty
Nodes Before Rest Range	
Max Node Distance	1.75

11. When you have done everything properly you are almost done! If you have used GOAP then you should go to Tools > RaafOritme > Smart NPCs > GOAPify Scene. Click on GOAPify Scene and that's it!


Scene Upgrader (GOAP)

Scene Upgrader (GOAP)

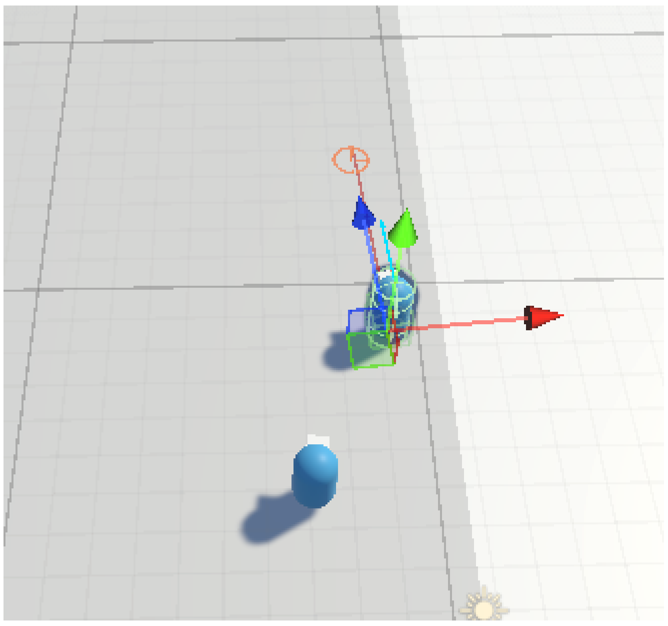
If you are using a GOAP powered agent in the scene you will need to prepare the scene for this type of AI. This tool will automatically all required components to all existing basic scene objects such as patrol areas. This means that the GOAP powered agents can utulize these options as well.

GOAPify Scene

You can easily remove all the added GOAP components through the button below.

Remove GOAPify

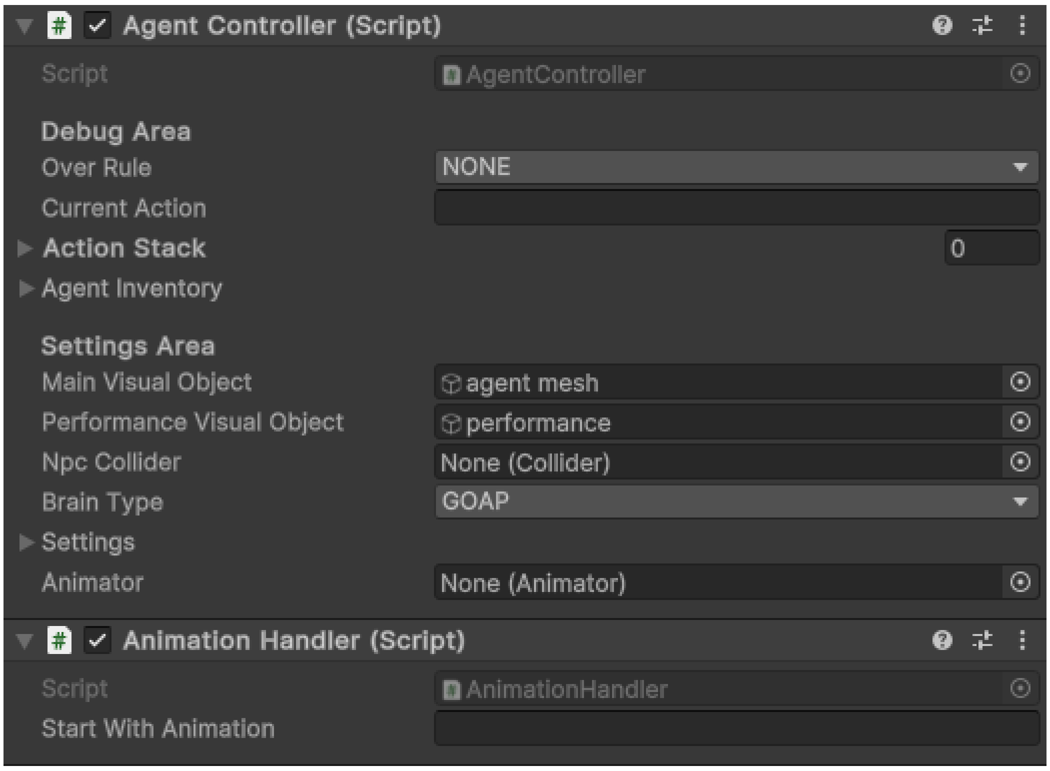
Now you can simply hit the play button and see the magic unfold! **Congratulations, you are done!**



Project Integration

Animations

Each NPC after being generated has the Agent Controller and Animation Handler as components. These components have settings related to the animations.



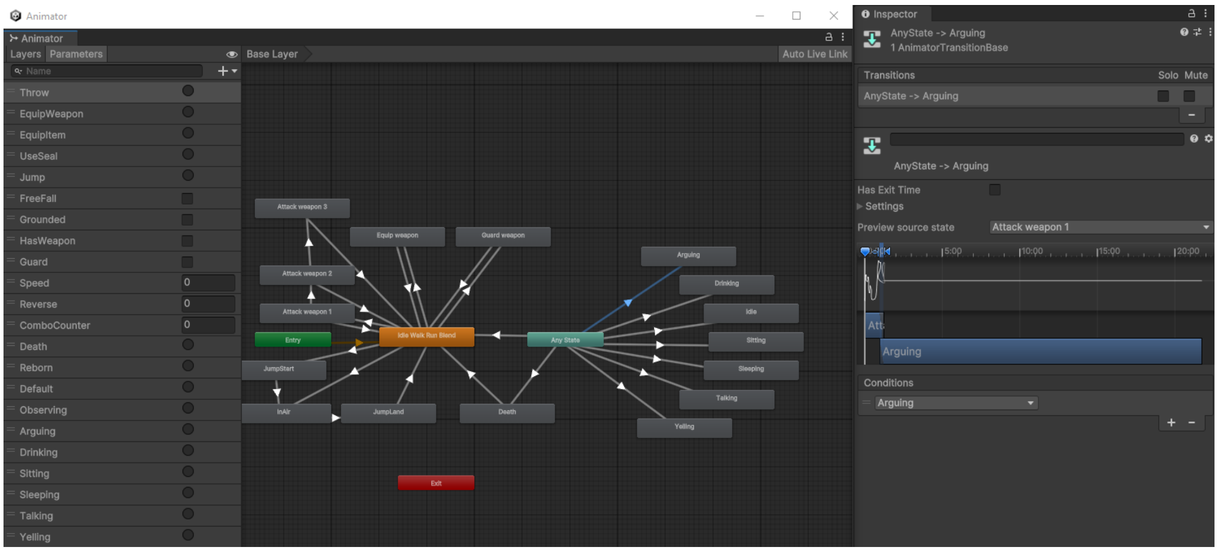
Agent Controller (Script)

Script	AgentController
Debug Area	
Over Rule	NONE
Current Action	
Action Stack	0
Agent Inventory	
Settings Area	
Main Visual Object	agent mesh
Performance Visual Object	performance
Npc Collider	None (Collider)
Brain Type	GOAP
Settings	
Animator	None (Animator)

Animation Handler (Script)

Script	AnimationHandler
Start With Animation	

Most work through a trigger. If you have a character with an animator, make sure to use triggers for your animation as well to invoke specific animations. An example of an animator setup can be seen below.



Throw	●
EquipWeapon	●
EquipItem	●
UseSeal	●
Jump	●
FreeFall	☐
Grounded	☐
HasWeapon	☐
Guard	☐
Speed	0
Reverse	0
ComboCounter	0
Death	●
Reborn	●
Default	●
Observing	●
Arguing	●
Drinking	●
Sitting	●
Sleeping	●
Talking	●
Yelling	●

AnyState -> Arguing
1 AnimatorTransitionBase

Transitions	Solo	Mute
AnyState -> Arguing	☐	☐

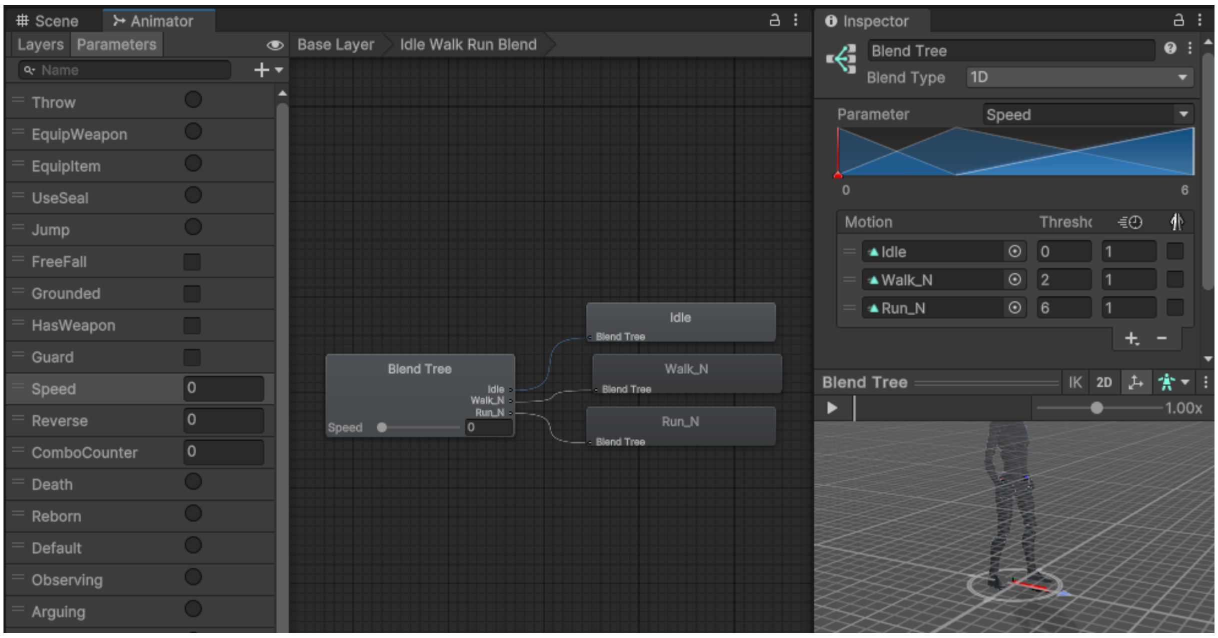
AnyState -> Arguing

Has Exit Time	☐
Settings	
Preview source state	Attack weapon 1

Conditions

Arguing

Other than that a float value such as movement speed is updated as well, this allows your blendtree to blend between the corresponding animations as required.



Throw	●
EquipWeapon	●
EquipItem	●
UseSeal	●
Jump	●
FreeFall	☐
Grounded	☐
HasWeapon	☐
Guard	☐
Speed	0
Reverse	0
ComboCounter	0
Death	●
Reborn	●
Default	●
Observing	●
Arguing	●

Blend Type 1D

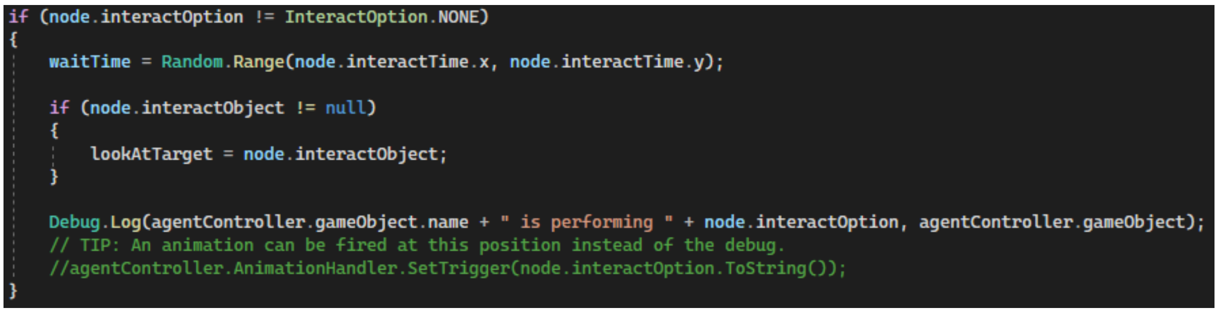
Parameter Speed

Motion	Threshold		
Idle	0	1	☐
Walk_N	2	1	☐
Run_N	6	1	☐

In the Agent Controller the Animator has to be me manually set. There are simply too many situations where various animators can be in a nested mesh, therefor you should assign it yourself. You can also do this through a script that assigns it on run time. If you do make sure to do this during the Awake method.

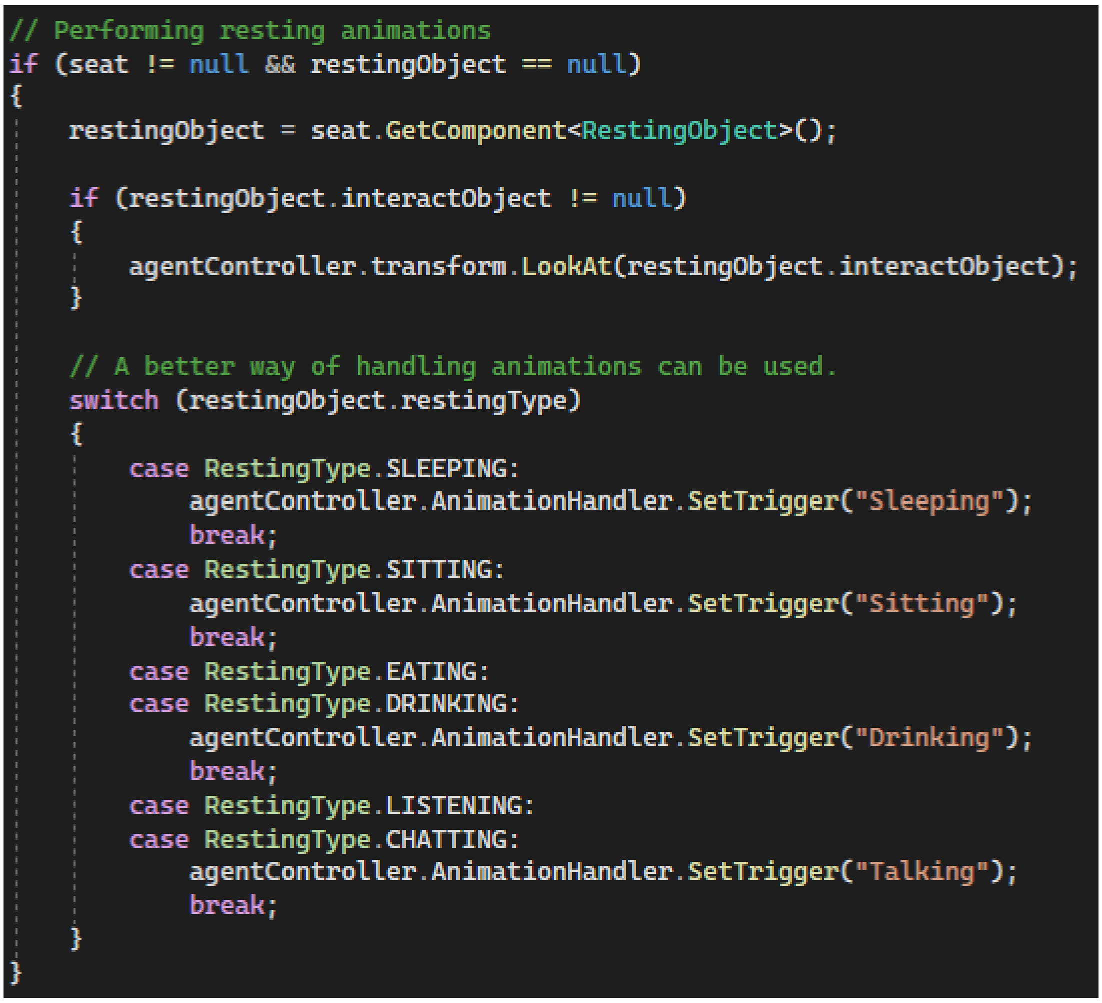
In the Animation Handler you will notice a field named Start With Animation. This field takes a string that it uses as parameter to fire a specific animation. Your animation settings decide whether it should loop or not.

If you have an animation heavy project you most likely have various world objects with animation parameters. In the PatrolModule.cs script you can find in the enumerator for PerformNodeAction a place where you can integrate your trigger. The commented line of code is fully functional although it might not be the best solution there is.



```
if (node.interactOption != InteractOption.NONE)
{
    waitTime = Random.Range(node.interactTime.x, node.interactTime.y);

    if (node.interactObject != null)
    {
        lookAtTarget = node.interactObject;
    }

    Debug.Log(agentController.gameObject.name + " is performing " + node.interactOption, agentController.gameObject);
    // TIP: An animation can be fired at this position instead of the debug.
    //agentController.AnimationHandler.SetTrigger(node.interactOption.ToString());
}
```

The IdleModule also has a section for animations at the RestingPlace method. This can be further fine-tuned or expanded upon.



```
// Performing resting animations
if (seat != null && restingObject == null)
{
    restingObject = seat.GetComponent<RestingObject>();

    if (restingObject.interactObject != null)
    {
        agentController.transform.LookAt(restingObject.interactObject);
    }

    // A better way of handling animations can be used.
    switch (restingObject.restingType)
    {
        case RestingType.SLEEPING:
            agentController.AnimationHandler.SetTrigger("Sleeping");
            break;
        case RestingType.SITTING:
            agentController.AnimationHandler.SetTrigger("Sitting");
            break;
        case RestingType.EATING:
        case RestingType.DRINKING:
            agentController.AnimationHandler.SetTrigger("Drinking");
            break;
        case RestingType.LISTENING:
        case RestingType.CHATTING:
            agentController.AnimationHandler.SetTrigger("Talking");
            break;
    }
}
```

These 2 modules show different examples and way of implementing animations in your project according to your needs. Keep in mind that this is not mandatory, but simply an addition to bring your NPC to the next level.

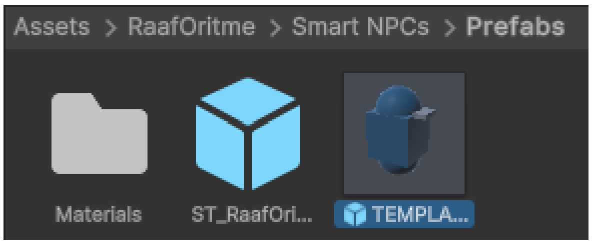
Sounds

Similarly to how animations are triggered / updated you can do the same thing for anything audio related. As each project heavily varies in audio system there is currently nothing in place for it. Every animation section however is a perfect place to implement an audio trigger for your own audio system, whether it is Unity’s built in audio, FMOD, or Wise.

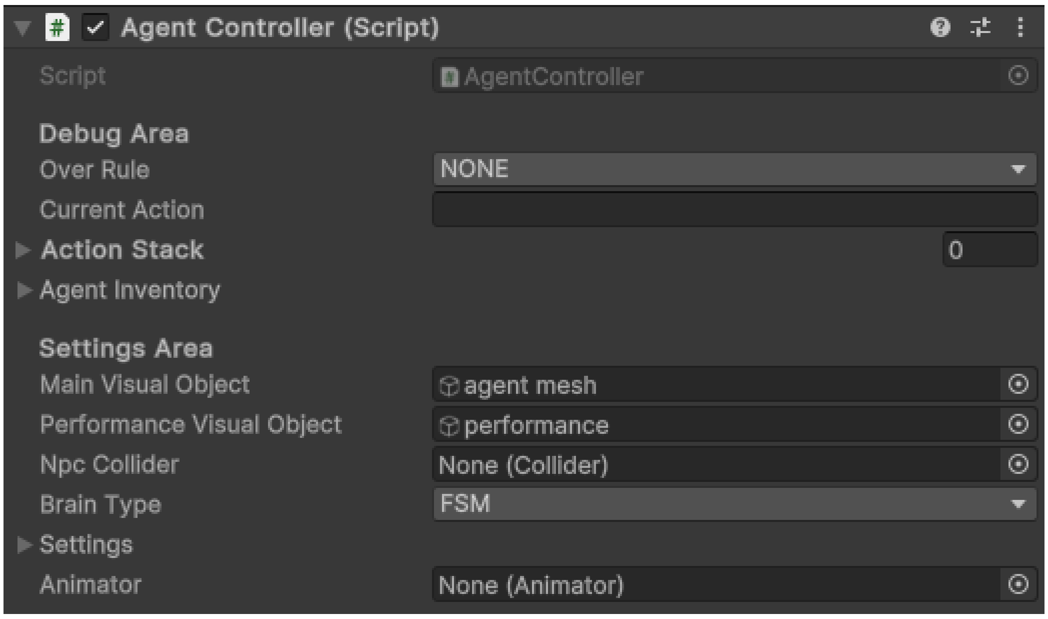
Characters

What really makes your NPCs stand out from the crowd is your own custom model. Whether it is 2D or 3D doesn’t really matter, although it might affect some modules and require further fine tuning. The basic premise of this asset pack is build around the philosophy that 2D and 3D games can utilize and customize this asset pack to their needs.

In “Assets > RaafOritme > Smart NPCs > Prefabs” you will find TEMPLATE_NPC. This prefab is what will be used whenever you create a NPC through the tool. You can create as many variations from this as you want.



When you open this in prefab mode you will notice that the basic components on it might or might not have fields assigned. Other than the Main Visual Object, Performance Visual Object, and Animator you can ignore everything else.



Agent Controller (Script)

Script	AgentController
Debug Area	
Over Rule	NONE
Current Action	
Action Stack	0
Agent Inventory	
Settings Area	
Main Visual Object	agent mesh
Performance Visual Object	performance
Npc Collider	None (Collider)
Brain Type	FSM
Settings	
Animator	None (Animator)

A character prefab is build up from various objects. First you have the main (animated) mesh. You can replace the child “agent mesh” with your character. This should have an animator if it is animated, and this will be used as the main visual object. It is what you will always see when the visibility conditions are met. Please do read the Main Systems > Performance area from this manual if you wish to learn more about it.

The prefab has another child, the performance object. You can make this an empty game object or maybe you have a 2D silhouette that you can use for your NPCs when they are a bit too far. This object is enabled when it doesn’t need to simulate animations and heavy meshes. It is used for performance.

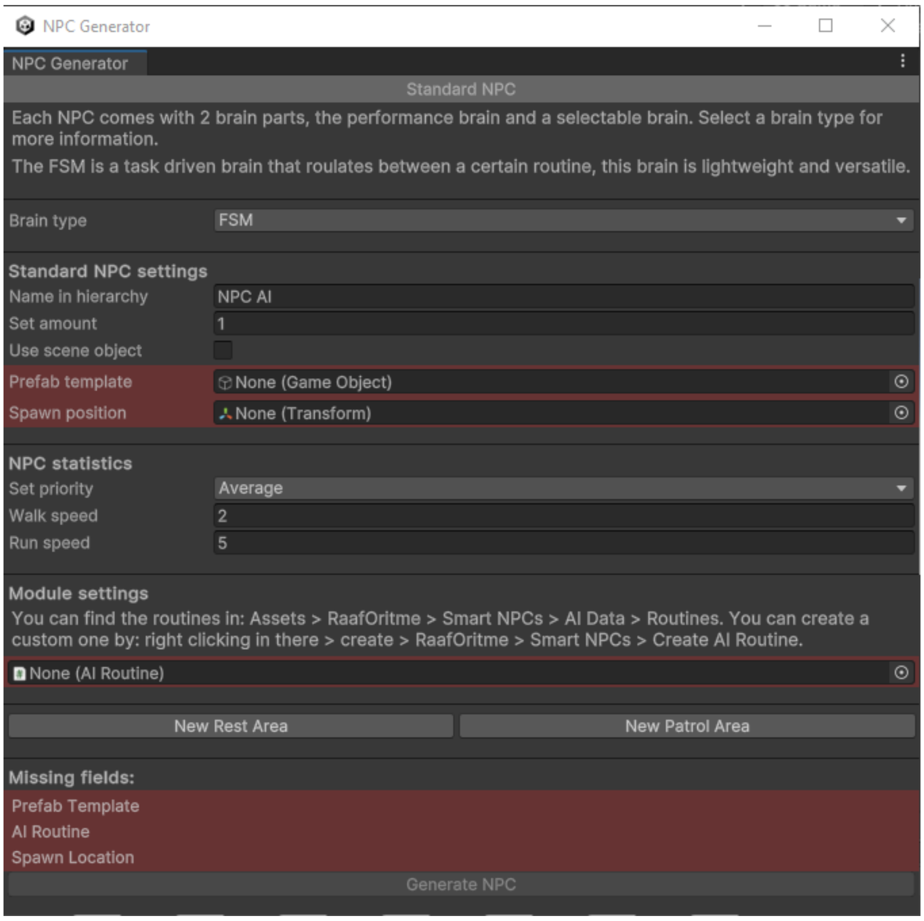
After having set the fields with your new additions your template will be good to go. All the other settings will simply be set during the use of the tool itself.

Tool Overview

NPC Generator

The main attraction of this asset pack is the NPC Generator. It works out of the box as is. It can be fine tuned with the settings it offers and it can be fully customized with it's modular foundation.

You can access the tool with CTRL+T or by going in the top bar to Tools > RaafOritme > Smart NPCs > NPC Generator.



NPC Generator

Standard NPC

Each NPC comes with 2 brain parts, the performance brain and a selectable brain. Select a brain type for more information.
The FSM is a task driven brain that roulates between a certain routine, this brain is lightweight and versatile.

Brain type: FSM

Standard NPC settings

Name in hierarchy: NPC AI
Set amount: 1
Use scene object: ☐
Prefab template: None (Game Object)
Spawn position: None (Transform)

NPC statistics

Set priority: Average
Walk speed: 2
Run speed: 5

Module settings

You can find the routines in: Assets > RaafOritme > Smart NPCs > AI Data > Routines. You can create a custom one by: right clicking in there > create > RaafOritme > Smart NPCs > Create AI Routine.

None (AI Routine)

Missing fields:

Prefab Template
AI Routine
Spawn Location

Each field within the generator is self-explanatory due to the tooltips they contain or the descriptive information boxes. The buttons will open other tools for you that are most likely necessary to achieve the best results possible.

The main principle behind this tool is to only show what is needed when it is needed, hence why there are various smaller windows and mini tools to assist you in the creation of your NPC and the environment for your NPC.

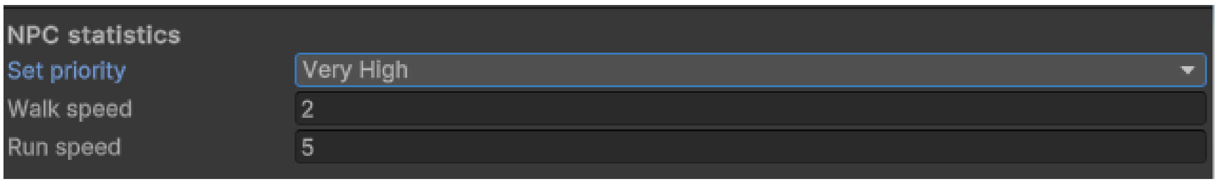
When you add a routine you will notice that the windows, for example Idle, is collapsable by clicking on it again after opening it. This ensures that you won't get overwhelmed with all the possible information.

If you want to further customize the tool itself you can find the scripts for this tool in Assets > RaafOritme > Smart NPCs > Scripts > Editor > Smart NPCs Tool. You can also create your own

NPC generator by inheriting the BaseNPCGenerator. This ensures that it can be used with the existing tool itself.

The NPCGenerator script is the script that draws the main tool. In here you can also find a sample about how you could add for own windows.

In the tool you will often also find field where it asks you to set something, which then has options between Extremely Low and Extremely High. These options are enums that are mathematically transformed into values. For example, when you are talking about priority it is easier to understand that Very High is more important than High. But numbers wise, which number would be actually high and which would be actually low? When it comes to speed it is much easier to understand numbers rather than words, keep this in mind when using and designing.



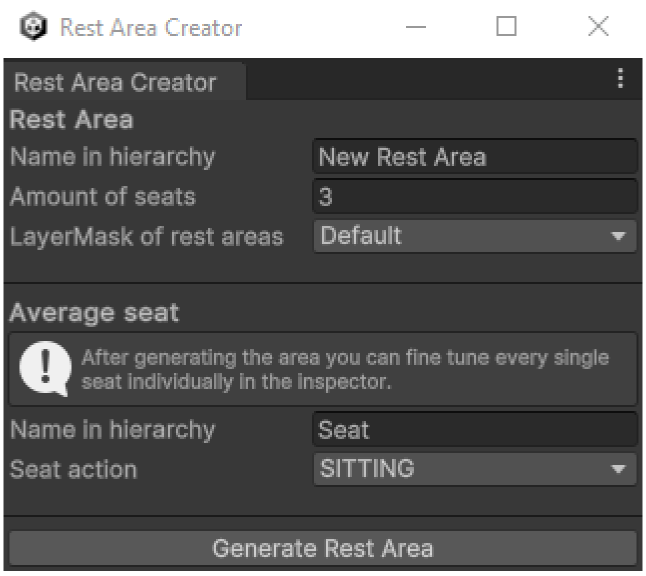
NPC statistics

Set priority	Very High
Walk speed	2
Run speed	5

After having created your NPC you can further fine tune it in the Agent Controller component on the NPC itself.

Rest Area Creator

Similar to the NPC Generator you can find this tool as a button in the NPC Generator or in the top bar; Tools > RaafOritme > Smart NPCs > Create Rest Area.



Rest Area

Name in hierarchy	New Rest Area
Amount of seats	3
LayerMask of rest areas	Default

Average seat

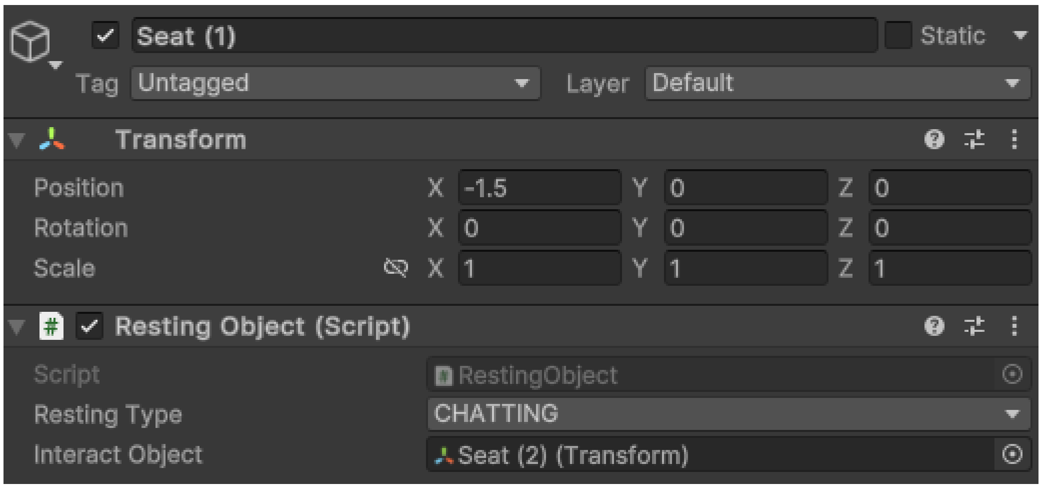
After generating the area you can fine tune every single seat individually in the inspector.

Name in hierarchy	Seat
Seat action	SITTING

Creating a rest area allows your NPCs to rest at specific locations while performing specific actions. For example going to a café through the main entrance and then sitting at a table to consume a stew. Once it is done consuming the stew (animation is finished) it will then leave the building and continue with its basic routine. It is more meaningful if each seat is placed on an actual Game Object, for example a chair. This leaves a more believable impression on the end user.

After having created your rest area, you can always select each seat in there to even further fine tune the interactions. Where should the NPC be looking at?

Should each seat have the exact same animation? Maybe the NPC should perform a drinking animation at the bar, chatting animation near the bathroom, and an eating animation at the table. Of course this isn't the limit! The Resting Type parameter is an enum which you can further expand upon. This enum is used for animations through a trigger, so it is only fired once.



Seat (1) — Static
Tag: Untagged — Layer: Default

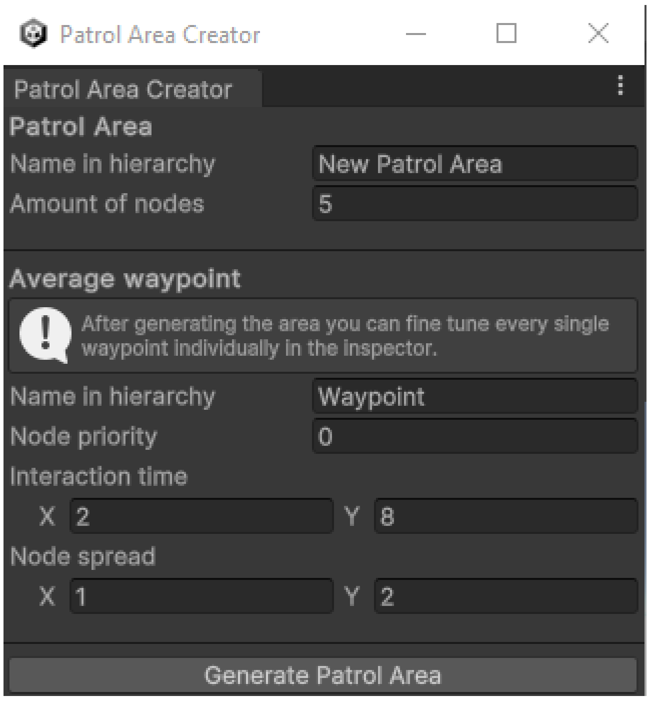
Transform
- Position: X -1.5, Y 0, Z 0
- Rotation: X 0, Y 0, Z 0
- Scale: X 1, Y 1, Z 1

Resting Object (Script)
- Script: RestingObject
- Resting Type: CHATTING
- Interact Object: Seat (2) (Transform)

The rest area works in such a way that each seat has to be booked by a NPC before it can be occupied. This prevents multiple NPCs from sitting on the same chair. Once a NPC has finished its resting activity it returns the seat to the resting area so that it becomes available for a new NPC to utilize. This ensures a dynamic flow. You can learn more about the idle system at Modules > Idle in this documentation.

Patrol Area Creator

Similar to the NPC Generator you can find this tool as a button in the NPC Generator or in the top bar; Tools > RaafOritme > Smart NPCs > Create Patrol Area.



Patrol Area Creator

Patrol Area
- Name in hierarchy: New Patrol Area
- Amount of nodes: 5

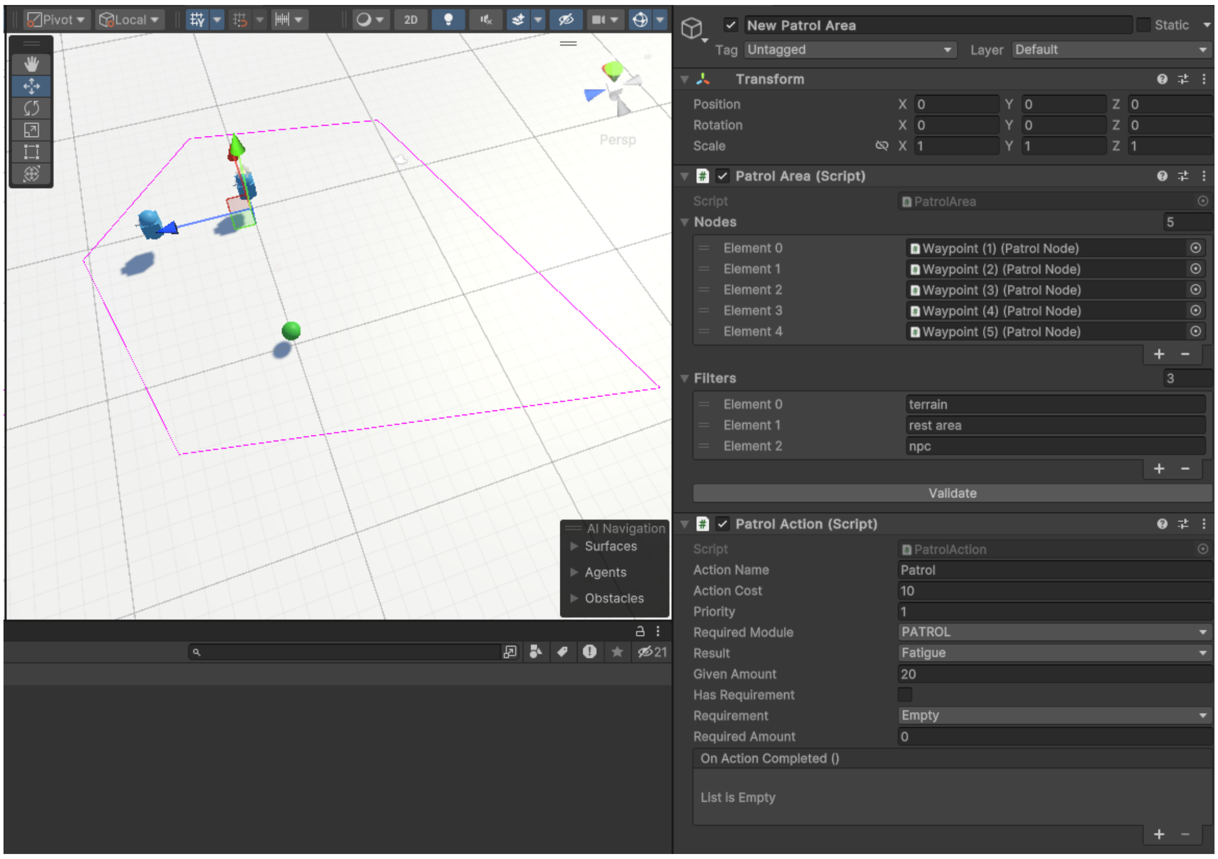
Average waypoint

After generating the area you can fine tune every single waypoint individually in the inspector.

- Name in hierarchy: Waypoint
- Node priority: 0
- Interaction time: X 2, Y 8
- Node spread: X 1, Y 2

Generate Patrol Area

The Patrol Area Creator is a small tool that helps you to create patrol areas in a certain environment, for example a market street. You can set the amount of nodes to get an amount to start with when you hit generate.



New Patrol Area — Static

Tag: Untagged — Layer: Default

Transform

	X	Y	Z
Position	0	0	0
Rotation	0	0	0
Scale	1	1	1

Patrol Area (Script)

Script	PatrolArea
Nodes	5
Element 0	Waypoint (1) (Patrol Node)
Element 1	Waypoint (2) (Patrol Node)
Element 2	Waypoint (3) (Patrol Node)
Element 3	Waypoint (4) (Patrol Node)
Element 4	Waypoint (5) (Patrol Node)
Filters	3
Element 0	terrain
Element 1	rest area
Element 2	npc

Validate

Patrol Action (Script)

Script	PatrolAction
Action Name	Patrol
Action Cost	10
Priority	1
Required Module	PATROL
Result	Fatigue
Given Amount	20
Has Requirement	☐
Requirement	Empty
Required Amount	0

On Action Completed ()

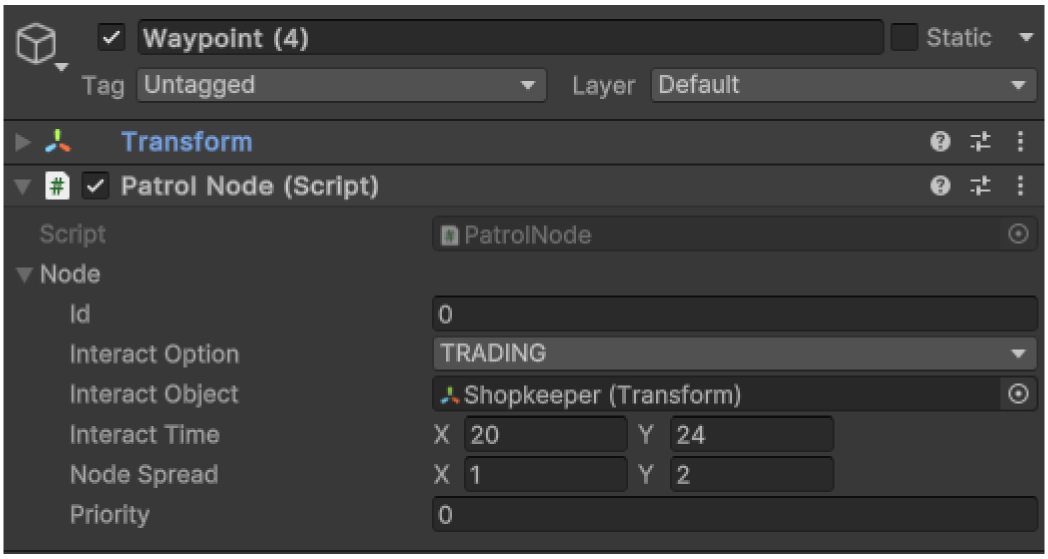
List is Empty

When you select the Patrol Area game object you will notice in the Scene overview that there will be lines drawn connecting each node. This should give you an idea of the path that will be patrolled. These lines will be used as a path for the performance brain.

When you press the button Validate in the Patrol Area component it will check for you if the nodes are reachable without obstruction. This is a simple check by checking for any obstruction in between the nodes. There is also a string array field named filters. You can add or remove strings in this list. All strings in this list are game objects that will be excluded from the obstruction check. So any object with the word “npc” in it will be excluded from the check.

The validate button is not mandatory, but an easy way to ensure that you don’t have to deal with minor, but tedious issues. Any obstruction will be printed as a message, and it highlights the object in the hierarchy.

Each patrol node, waypoint in this case, can be tweaked further for animations and realistic orientations. This is similar to the Rest Area system. You can change the interact option and extent them further in the code, in order to make the NPCs behave more dynamic and feel more alive.


Waypoint (4)
Static
Tag Untagged
Layer Default
Transform
Patrol Node (Script)
Script PatrolNode
Node
Id 0
Interact Option TRADING
Interact Object Shopkeeper (Transform)
Interact Time X 20 Y 24
Node Spread X 1 Y 2
Priority 0

Just like the resting system the selectable collection is an enum which fires animations as a trigger. You can expand upon this further with your own additions to make this more dynamic and custom.

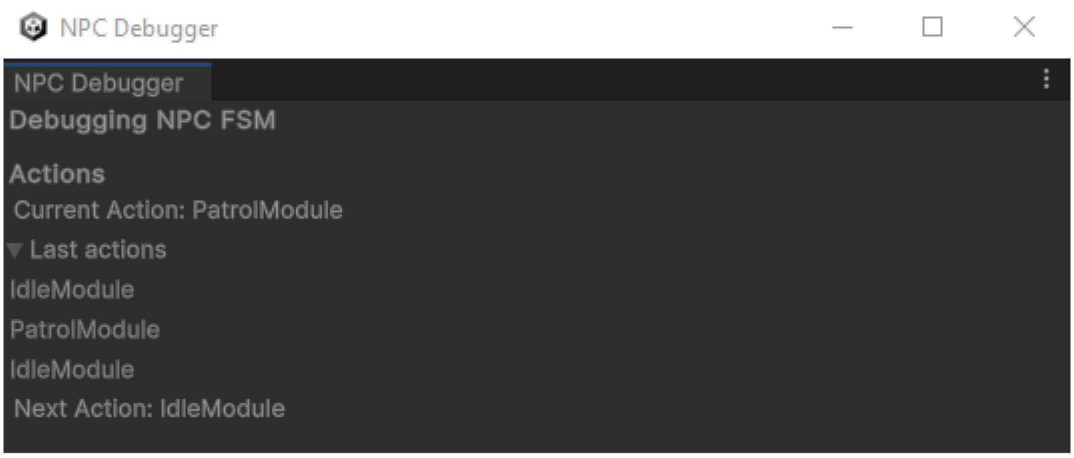
Debugger

The debugger is a work in progress and will be updated with every major update. The main philosophy behind the debugger is to have a peak in the brain of the AI. This means that the FSM powered brain will be showing a lot less data, since there isn't much going on in there. The GOAP contains the most, as there are various sub-systems being part of the decision-making process.

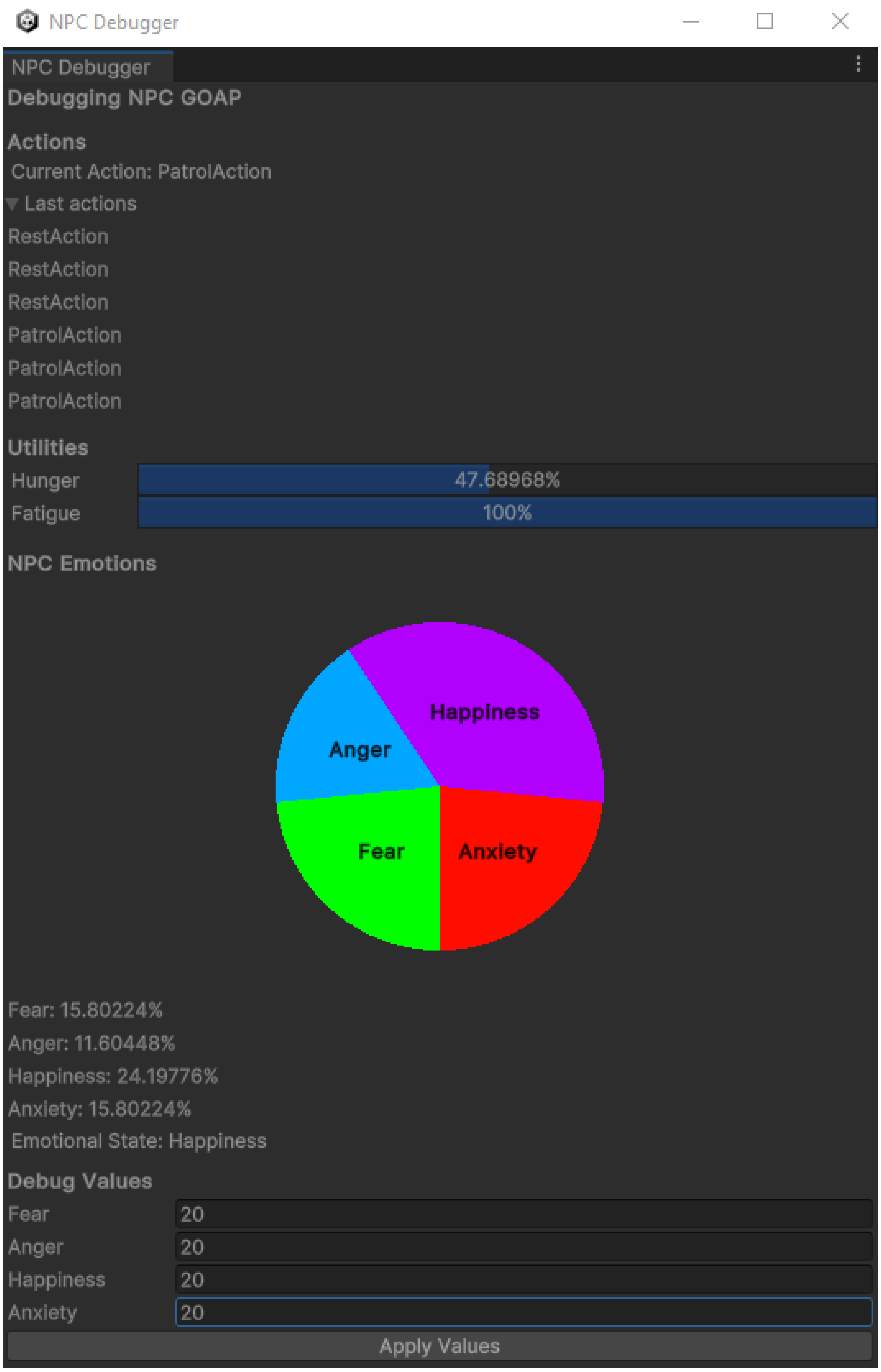
This debugger is only active during play mode and can be accessed through Tools > RaafOritme > Smart NPCs > NPC Debugger.

In order to see data you need to select the NPC which you would like to inspect. It must have an agent controller component, otherwise it won't be able to pull data to inspect. Keep in mind that selecting the NPC in the hierarchy also allows you to see plenty of things in the inspector.

The FSM powered NPC simply shows what it is currently performing, what it has done in the past, and what the next task is going to be.


NPC Debugger
NPC Debugger
Debugging NPC FSM
Actions
Current Action: PatrolModule
Last actions
IdleModule
PatrolModule
IdleModule
Next Action: IdleModule

The GOAP powered NPC uses a ton more statistics in order to make a decision, it has an utility system and emotions which create a certain bias.


NPC Debugger

NPC Debugger

Debugging NPC GOAP

Actions

Current Action: PatrolAction

Last actions

RestAction
RestAction
RestAction
PatrolAction
PatrolAction
PatrolAction

Utilities

Hunger	47.68968%
Fatigue	100%

NPC Emotions

Fear: 15.80224%
Anger: 11.60448%
Happiness: 24.19776%
Anxiety: 15.80224%
Emotional State: Happiness

Debug Values

Fear	20
Anger	20
Happiness	20
Anxiety	20

Apply Values

In the debugger you can force hard values to see how it impacts the current state of emotions and potentially the decisions it makes. This also comes with a stack of actions it did in the past. This NPC type requires a lot more fine tuning so that it makes decisions based on the values that you want.

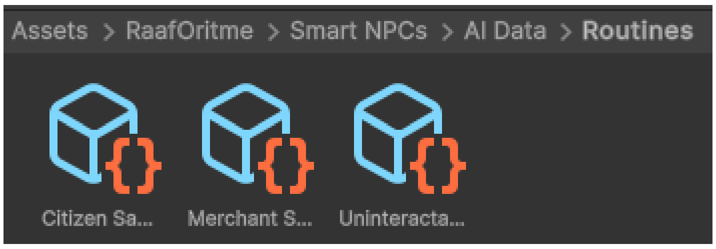
Even though the debugger is a work in progress, you can easily expand upon it. You can find the script in Assets > RaafOritme > Smart NPCs > Scripts > Editor > Other tools > NPCDebuggerWindow.cs.

Since debugging and information insights will vary per studio and project, the base debugging tool will be pretty abstract while it covers the basics.

Modules

A module is a component of behaviour that allows your NPC to be able to perform something. You can look at it as if it are the senses or limbs of your NPC. For example if you lost your arms you would no longer be able to interact with objects, without legs you can't walk, etc.

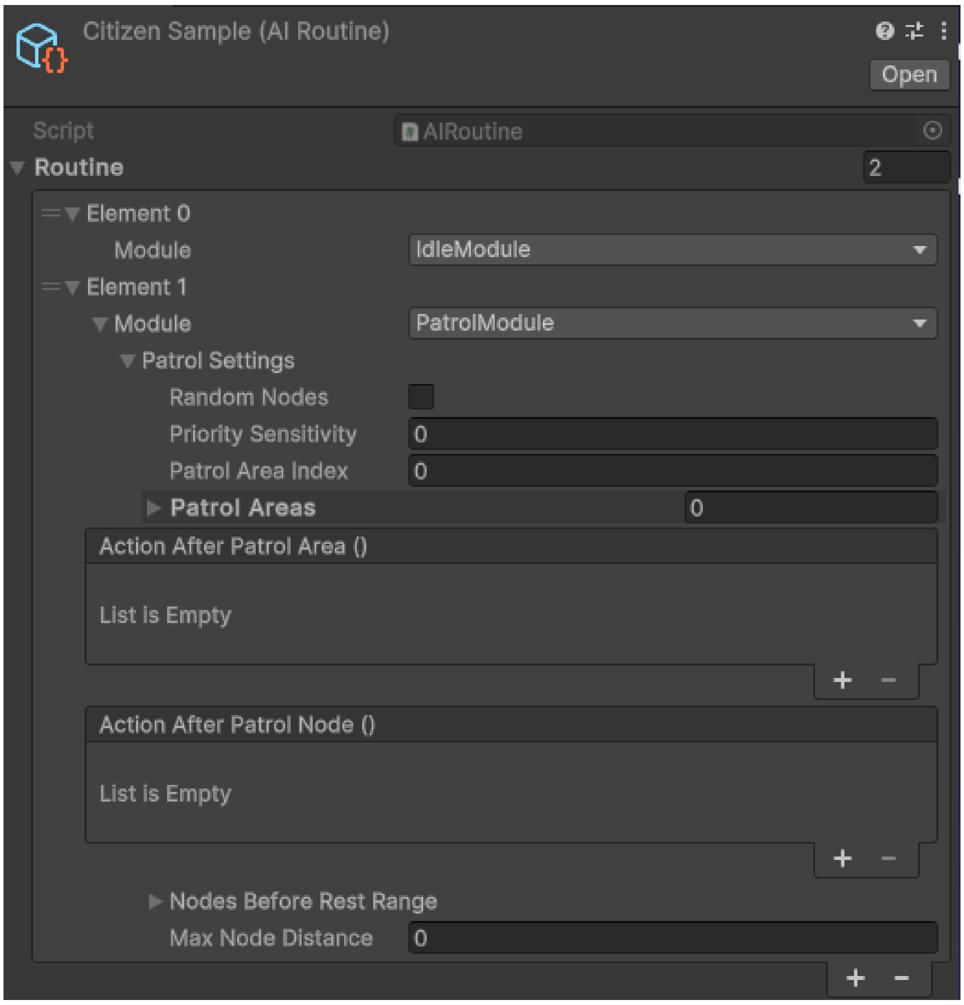
Each routine stack can also be seen as how many of those limbs / senses the AI will have to work with. More capabilities come at the cost of performance.



You can create your own routine by right clicking and going to Create > RaafOritme > Smart NPCs > Create AI Routine. The AI routines are scriptable objects, meaning that you can easily edit / modify them for all NPCs in one place, but changes between play and edit more are kept.

In each routine you can add any amount of modules that you want. You can have the same module over and over again as well, as it can be useful for the FSM to have a certain degree of variations in, for example, the patrol areas it should patrol at.

It is important to keep in mind that when you edit the AIRoutine class that you have to create all the ScriptableObject again. They will exist, but their data will be reset every time you hit play. Screenshot your data before you change the class, after that delete the old objects and create new ones.



Citizen Sample (AI Routine)

Open

Script: AIRoutine

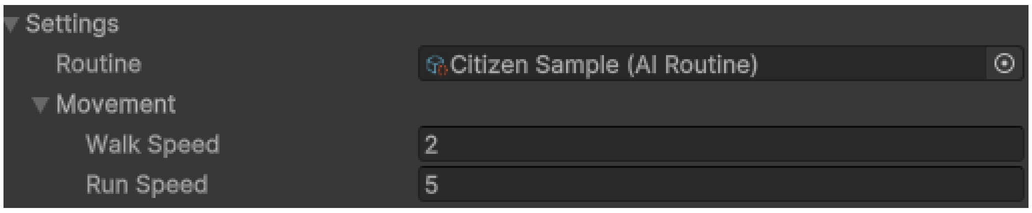
Routine: 2

- Element 0
  - Module: IdleModule
- Element 1
  - Module: PatrolModule
    - Patrol Settings
      - Random Nodes: ☐
      - Priority Sensitivity: 0
      - Patrol Area Index: 0
      - Patrol Areas: 0
    - Action After Patrol Area ()
      - List is Empty
    - Action After Patrol Node ()
      - List is Empty
    - Nodes Before Rest Range
    - Max Node Distance: 0

In the routine you only need to select which modules you want, where, and how many. The settings are set through the NPC Generator. Keep in mind though that multiple of the same will all get the same values through the tool, so you will need to manually alter the fields for the NPCs generated with this routine. Since you can bulk select (CTRL + LMB) in the hierarchy I doubt that this will cause much problems.

Each of the module (both existing and yet to come) are explained on the following pages.

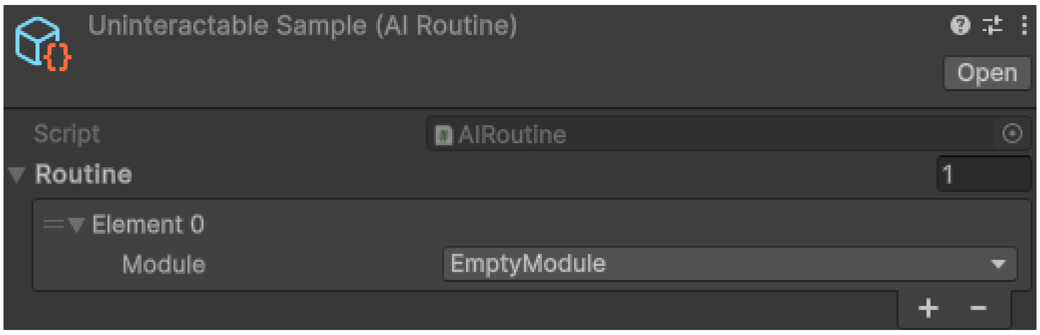
The agent controller contains settings for each module. These can be find in the inspector at the settings. You can further expand upon this by editing the SettingsAI.cs script.



- Settings
  - Routine: Citizen Sample (AI Routine)
  - Movement
    - Walk Speed: 2
    - Run Speed: 5

All agents require at least these 3 settings. The agents come by default with the Nav Mesh Agent from Unity in order to do basic path finding and obstacle avoidance. The script Pathfinding.cs is the hook between the navigation logic and the communication from the modules. Replacing the NavMeshAgent with your own solution should mainly be done here and partially in the AgentController.

Empty


Uninteractable Sample (AI Routine)

Open

Script: AIRoutine

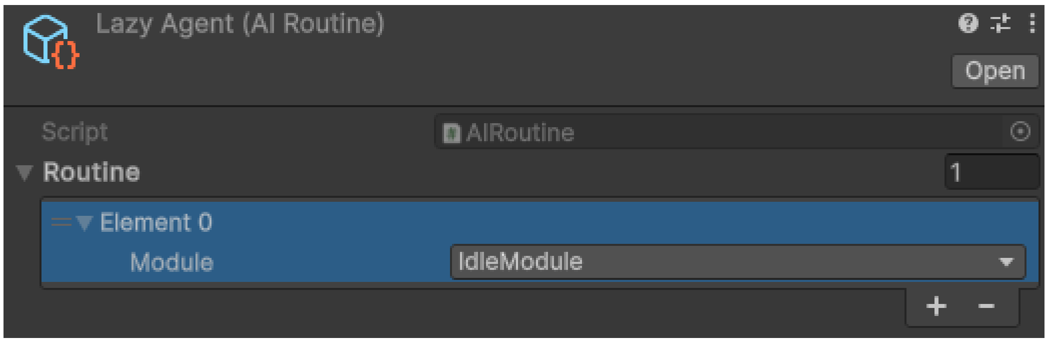
Routine: 1

Element 0

Module: EmptyModule

You might want to have a NPC in your game that is not interactable and just standing there. Assign this module to it. The only purpose of this is to have NPC dummies essentially that have no behaviour nor interact options. Think of it such as a corpse, that still might have other data that might be mandatory.

Idle


Lazy Agent (AI Routine)

Open

Script: AIRoutine

Routine: 1

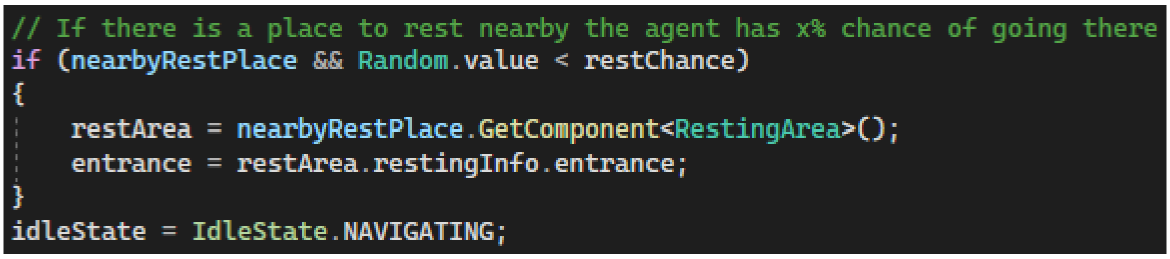
Element 0

Module: IdleModule

The idle system is a very complex system that ensures that your NPCs will perform routines that look and feel logical without introducing odd bugs such as multiple NPCs sitting on the same chair within each other.

This module (when activated) scans the surroundings to see if the NPC can rest somewhere, e.g. a restaurant. If there is such a place it will keep it in mind before choosing. It makes a choice between going "home" to it's residence or to go to the resting area. You can alter the code in order to for example force that it always goes to a rest area instead of the residence, whenever it finds one.

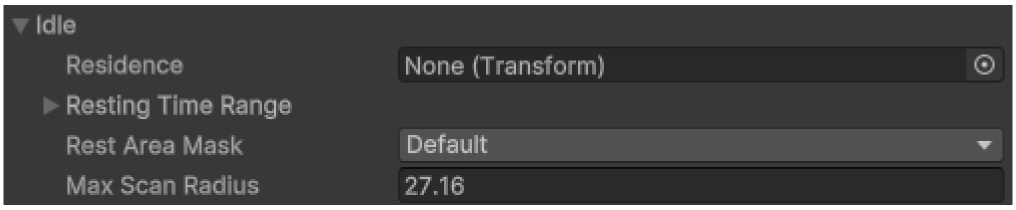
In the IdleModule.cs you can alter the restChance in the ChooseRestingPlace method. The variable can be found at the collection with the others at the first few lines of the class.


```
// If there is a place to rest nearby the agent has x% chance of going there
if (nearbyRestPlace && Random.value < restChance)
{
    restArea = nearbyRestPlace.GetComponent<RestingArea>();
    entrance = restArea.restingInfo.entrance;
}
idleState = IdleState.NAVIGATING;
```

When it finds a spot to rest at and decides to go there it will trigger the restaurant booking system of the rest area, where it has to book a seat in order to sit on it.

A seat can only be used by 1 entity. Once it leaves the place it has to return the seat to the restaurant, making space for a new NPC to book it.

Idle has some settings in the AgentController. These settings are where the residence are, how long it should rest for (random between 2 values), on which layer it can detect the objects for rest areas and what the maximal scan radius should be.

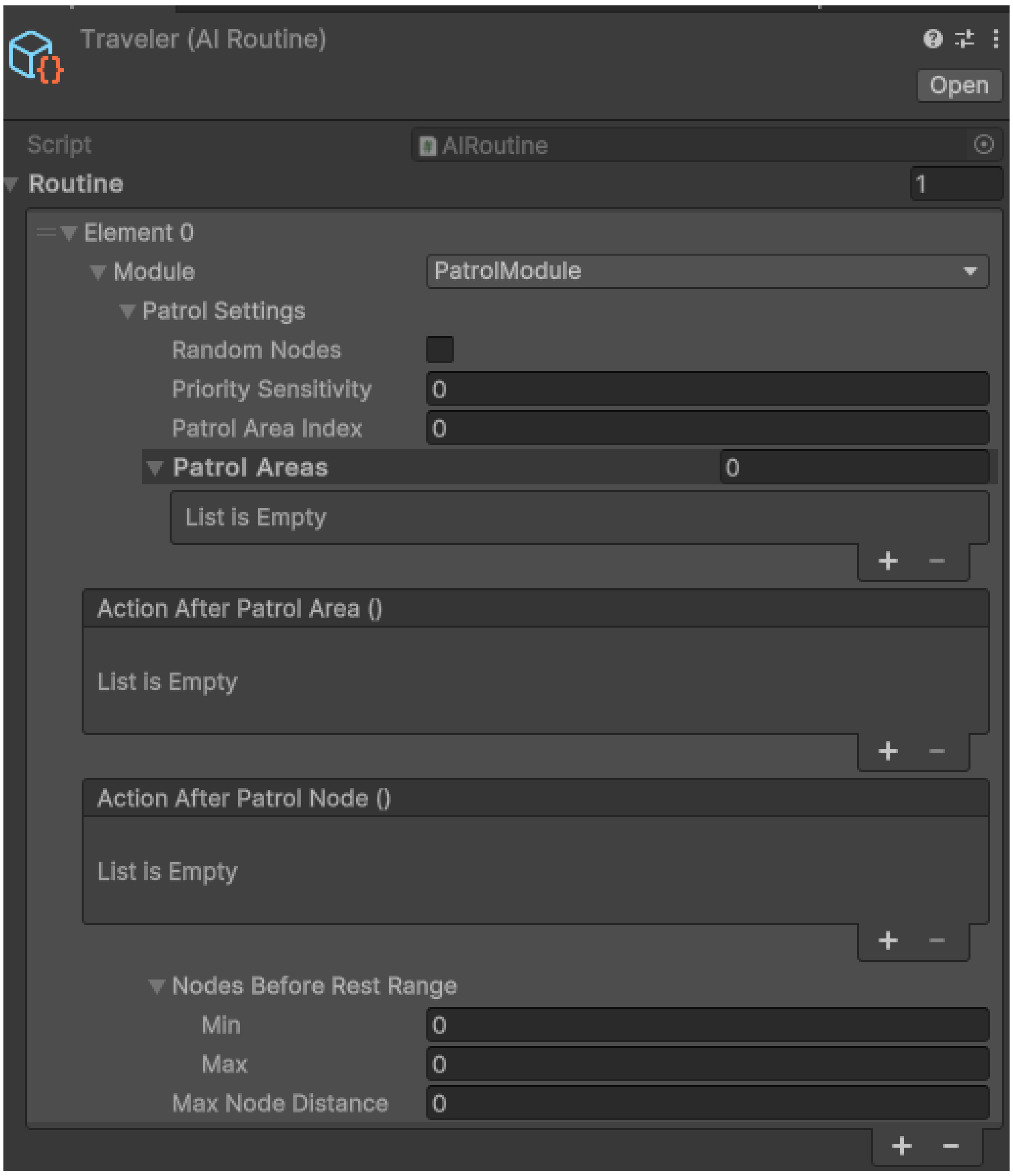


Idle

Residence	None (Transform)
Resting Time Range	
Rest Area Mask	Default
Max Scan Radius	27.16

Patrol

Some modules come with settings, these can be ignored. They are used by the NPC Generator to give the proper variables to play with for the NPCs. Patrol module is a fantastic example of this.



Traveler (AI Routine)

Script	AIRoutine
Routine	1
Element 0	
Module	PatrolModule
Patrol Settings	
Random Nodes	☐
Priority Sensitivity	0
Patrol Area Index	0
Patrol Areas	0

List is Empty

Action After Patrol Area ()

List is Empty

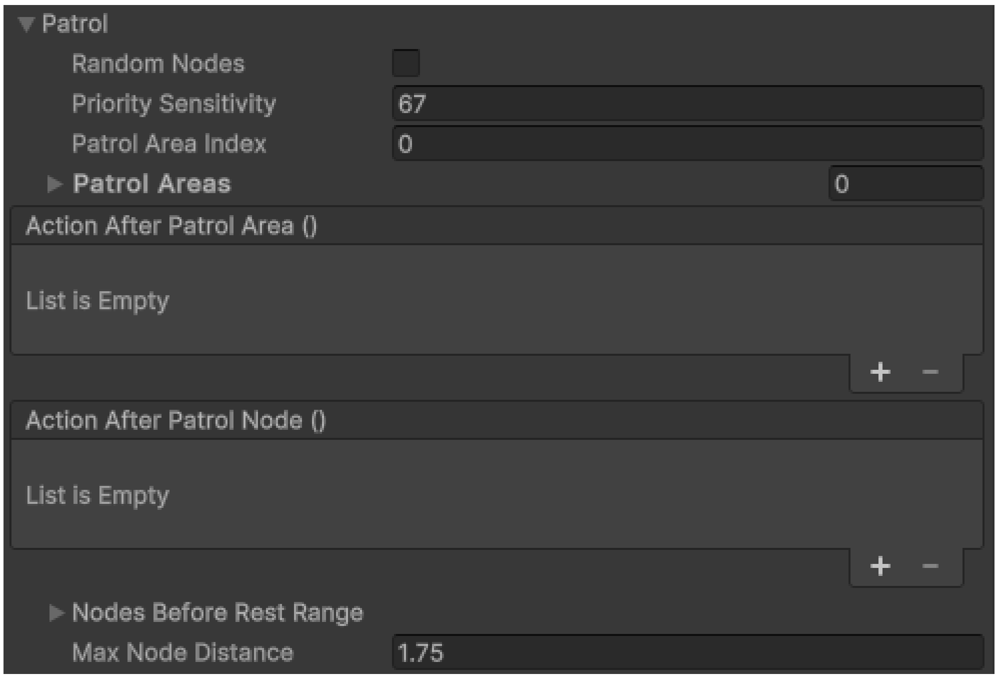
Action After Patrol Node ()

List is Empty

Nodes Before Rest Range	
Min	0
Max	0
Max Node Distance	0

The patrol module itself essentially allows the NPC to navigate in the world between nodes. This means it can use pathfinding in order to go somewhere and do something. By default this pack uses the Unity Navmesh Agent. You can of course remove this stack and use your own. Most likely this will be through patrol paths. A single NPC with a single patrol module can hold any amount of patrol areas. It sequentially goes from A to B, B to C, etc. The only randomness can be between the nodes from an area where it can go from C to H, H to A, etc.

An agent can have multiple areas to patrol at, but it also works with sensitivities. Some nodes contain priorities. The higher the sensitivity is, the more likely it is for an agent to visit that node more often. Once it has completed a patrol area, the agent might execute a specific action, this can be a callback method that you can simply drag in there. You can also set how many nodes the agent must explore before it gets tired, and lastly the range for when a NPC is at the node. In order to keep each node reachable place them roughly 0.5 to 1.0 units above the surface with a max node distance of at least 1.0 to ensure that the NPCs wont get trapped.

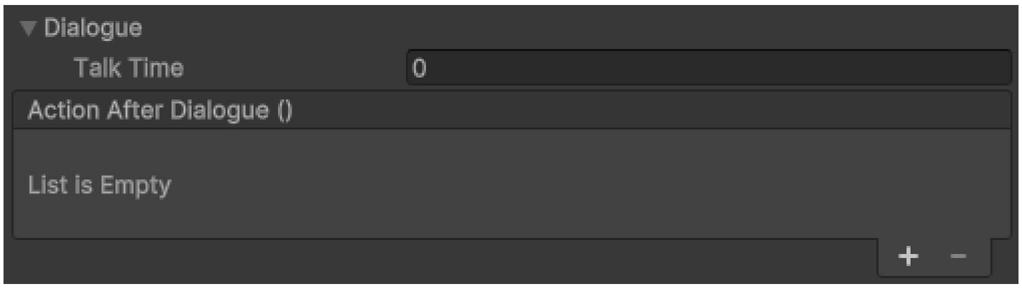

Patrol
Random Nodes ☐
Priority Sensitivity 67
Patrol Area Index 0
Patrol Areas 0
Action After Patrol Area ()
List is Empty
Action After Patrol Node ()
List is Empty
Nodes Before Rest Range
Max Node Distance 1.75

Dialogue

This module is scheduled to be added and released as an update.

With the dialogue module NPCs will have the ability to actually converse with one another. These can be seen as simple text fields in game. This will only be visualized when the player can see it, otherwise it isn’t necessary.

With the settings you can set how long a NPC should be talking for, and what it should do after the conversation.


Dialogue

Talk Time: 0

Action After Dialogue ()

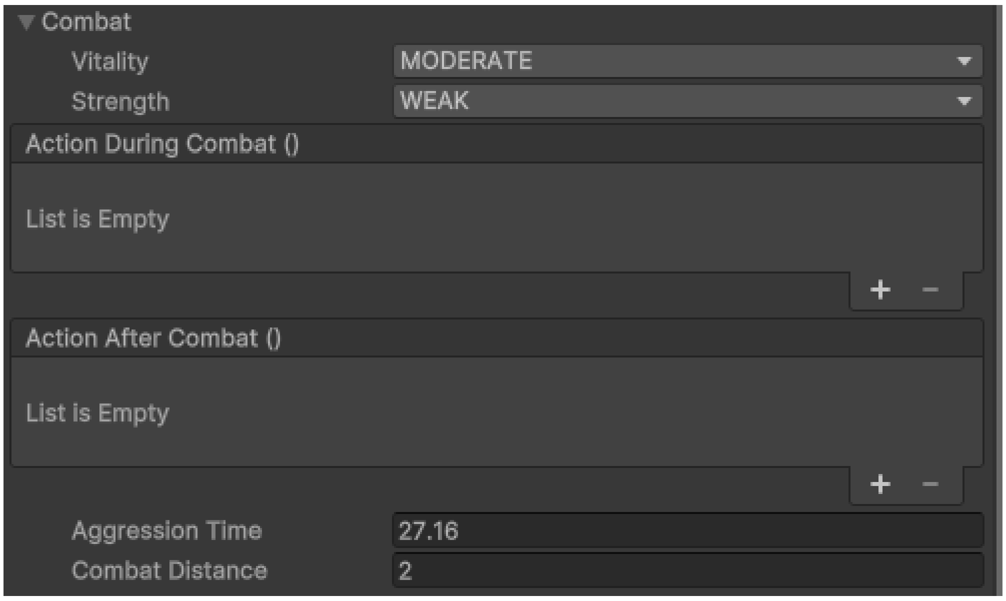
List is Empty

Combat

This module is scheduled to be added and released as an update.

With the combat module NPCs can fight or defend themselves. A player attacking a NPC will trigger combat. NPCs can choose to perform an action due to which they fight with another NPC as well.

The combat settings decides how strong or weak a NPC is for combat. It also shows how much distance it will keep and whether it should execute specific actions.


Combat

Vitality: MODERATE

Strength: WEAK

Action During Combat ()

List is Empty

Action After Combat ()

List is Empty

Aggression Time: 27.16

Combat Distance: 2

Sensory

This module is scheduled to be added and released as an update.

This module gives senses to the NPC as actual senses. The NPCs will be capable of listening and looking. A future expansion might include the sense of smell as well. A sense will allow impulse behaviour. For example a patrolling guard might see a fight breaking out due to which it decides to separate the two parties.

A citizen might hear a loud explosion and decide to run away completely frightened.

Retaliation

This module is scheduled to be added and released as an update.

This module allows fine tuning for how the NPC should respond when it receives an over rule. Over rules are immediate actions that require immediate responses.

If the NPC gets attacked should it fight back immediately or just run away? Is it a shy NPC that never wants to converse with others and just tries to walk away or maybe something completely different. With this module you can fine tune it.

Main Systems

The main driving force behind this asset pack are the brains that are powering the NPCs. Each brain has various focus points, a different purpose, and of course a different impact on performance. A clever combination can make this asset pack feasible for both low-end and high-end devices.

FSM

FSM stands for Finite State Machine. It is as the name implies something that operates on states. A state can be patrolling, idling, or any of the other modules. The FSM is a simple brain that has an x amount of tasks and it rotates chronologically through the order, once it is done it will start over again. In order to introduce some dynamic elements to it, it can be forced out of a state through an over rule. It can be forced to transition to a different state or simply just to resume what it was doing after dealing with the interruption.

This means that the AI has both an internal and external factor which can decide what it should do next. The internal factor is triggered when a state has been completed, this means, go to the next state. An external factor can be that the NPC is being talked to by the player, due to which it switches from patrolling to dialogue and after that back to patrolling.

The script BrainFSM.cs contains all the logic for this system. Each state is placed in a queue and dequeued when used in order to be queued again so it is placed at the end of the line.

GOAP

GOAP stands for Goal Orientated Action Planner. This means that this brain formulates a goal and a way to reach it. Currently out-of-the-box it comes with a simplistic way, each goal is an action and that's it. ActionPlanner.cs contains the logic for selecting an action to execute. It also contains a sample at the bottom for how you can created nested actions, for example in order to reach goal X, the NPC has to perform actions A, B, and C in this exact order. The sample code shows you how to reach this.

When the queue of actions you can simply tell the brain to dequeue the action in the queue and to execute it. When the queue is empty it will have reached its goal, which means that it can perform something new. This is essentially the main logic behind a GOAP.

Selecting a goal is truly what sets this system apart from others. In order to make a decision it has to weight various factors, e.g. how much energy does an action cost? How far is the action? How many requirements are there? Etc. This means that certain actions will have a positive bias where as others will have a negative one.

This bias factor is manipulated by the utility system and the emotion system. This makes the NPC more likely to choose 1 action over the other. For example if it wasn't able to complete a patrol action due to being attacked then it will remember that and if it can choose between patrolling or idling, it will choose idling. You can learn more about the emotions and utilities at the sub systems section of this manual.

When you use the GOAP system, your scene will have to be upgraded. You can find this tool in Tools > RaafOritme > Smart NPCs > GOAPify Scene. Without this your scene won't be GOAP ready. This ensures that all explorables in your world will be noticed by the GOAP. A rest area, patrol area, etc. are examples of explorables. They will each get a new component, which you will have to fine-tune settings wise.

If you use your own components, and other world explorables, please make sure that they contain a similar upgrade as well in order to become noticeable.

The BrainGOAP.cs is the main driving force of a GOAP NPC. It contains effects, conditions, and a personality. This personality decides how the NPC should pick actions, as it impacts the emotional behaviour.

Keep in mind that when you use GOAP that it is significantly more heavy than the FSM. Only hero-like NPCs should use GOAP. More are of course also okay as long as it is within the performance standards within your project.

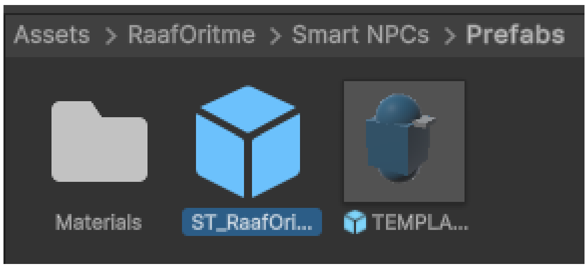
Performance

The performance brain is the second part of the brain of every single NPC out there. This is an extremely lightweight brain that simply updates positions of the NPC through patrol areas. This ensures that it feels as if the NPCs were still going on with their day-to-day job, without making things too crazy. On the other hand if you return shortly, the NPC won't be much further than it was before you left.

The MotherAI handles the update cycle of this, as well as when it should be toggled and when not. Toggling this comes in 2 ways. Disable the AI completely, go into performance mode. When you are pretty far from the NPC it doesn't need to be simulated, only when you are not too far.

This part of the brain makes the other 2 even more powerful, because with some good fine tuning, you can really gain a lot of performance while having a ton of diverse NPCs living in your world.

It is mandatory to have the ST_RaafOritme prefab in your scene, this contains the MotherAI script. Currently it finds the player by the tag "Player" and makes sure that everything around it has the correct settings NPC wise. You can expand easily upon this by giving it a list of public Transforms which the AI should check before determining whether it should be performant, off, or on. Each addition other than the player will make this significantly more heavy, especially when you have a lot of NPCs.



Sub Systems

Emotions

Emotions are 1 of the systems that impact the bias for the GOAP powered AI. The emotion system is capable of having emotional states, increasing / decreasing them over time and to gain a boost in a certain state. It however isn't a state machine, it is a combination of various factors.

Based on experiences it also registers whenever an action was completed how it experienced it (for now either good or bad). This means that bad memories will motivate a NPC to choose something else.

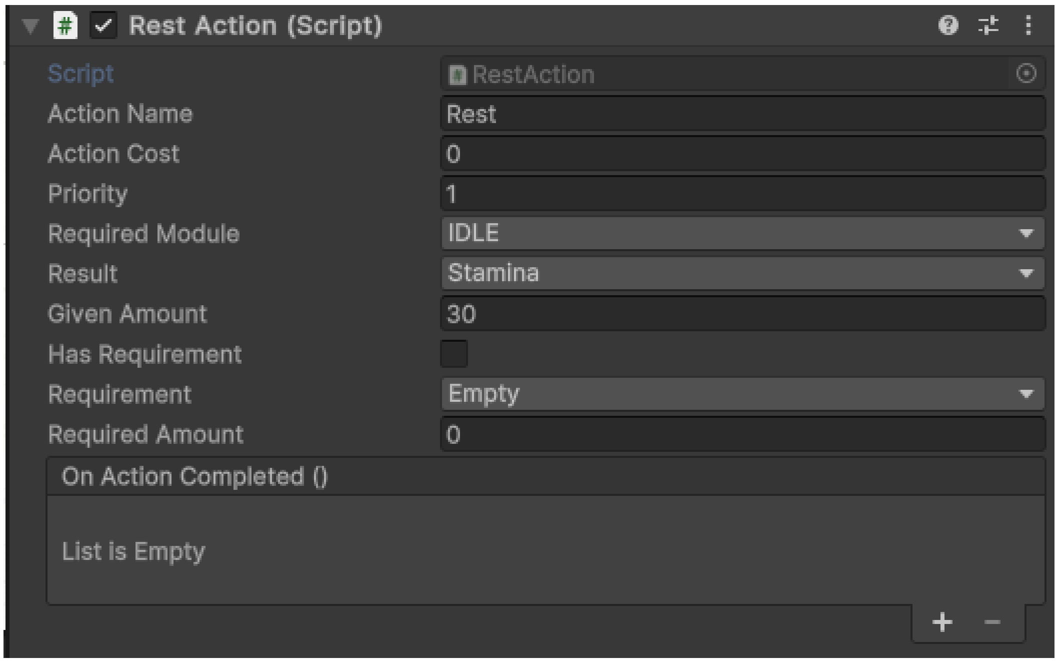
The personality of the NPC will also affect how certain emotions are updated and how emotional changes are received. An aggressive NPC might be more willing to engage in combat whereas an anxious NPC might just choose to run away.

This system gives a lot of depth and meaning behind the decision making of your NPCs. The folder, Assets > RaafOritme > Smart NPCs > Scripts > AI > Brain GOAP > Emotions, contains all the scripts related to this.

Utility

The utility system ensures that the GOAP has utilities that is has to keep track of. These can be as many and as crazy as you want. Each utility might give a certain condition to the NPC, which in return will have an impact on decision making. For example a tired and hungry NPC is not going to patrol, but going to rest at a restaurant. It can check if the restaurant gives for example food or stamina.

For now it can only give one result. A future update will allow you to use multiple results from 1 activity.


Rest Action (Script)

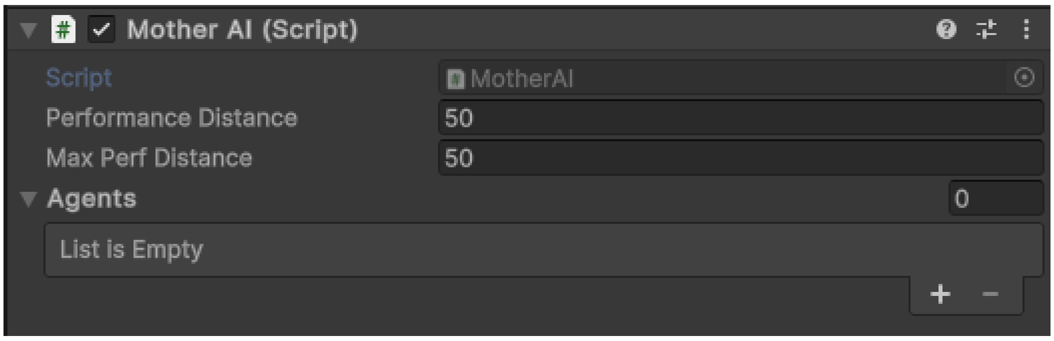
Script	RestAction
Action Name	Rest
Action Cost	0
Priority	1
Required Module	IDLE
Result	Stamina
Given Amount	30
Has Requirement	☐
Requirement	Empty
Required Amount	0

On Action Completed ()

List is Empty

MotherAI

The Mother AI is a component in the ST_RaafOritme prefab. It allows the NPCs to use the performance brain and is mandatory is it is the updater behind the brains. This ensures that the whole system is very performant. The performance distances correspond to how far the NPC has to be from the player in order to use the performance brain. Further than the distance means disabled, closer means that it will use either the FSM or GOAP based on your settings.

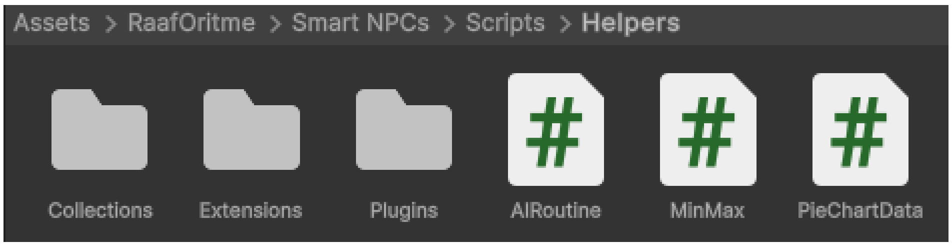

Mother AI (Script)

Script	MotherAI
Performance Distance	50
Max Perf Distance	50
Agents	0

List is Empty

The agents list is populated automatically.

Helpers


Assets > RaafOritme > Smart NPCs > Scripts > Helpers

There are quite a few helpers in this project to make things easier, both code-wise and designer wise. In the collections you can find most (if not all) of the enums used in the project. This makes it easier to expand on.

Extensions are scriptable additions to methods where you can use for example a custom/alternative method. For example in this asset pack there have to be made a lot of distance checks. Transform.WithinRange(target obj, x distance) allows you to get a true or false as a result, instead of doing a manual distance check. This method is also heavily optimized and more performant than the basic Distance check. Extensions are used in such a manner that you can use the same algorithm wherever you wish, especially for mathematics it can be really useful.

There is also a plugin, which was needed in order to create the picker in the ScriptableObject (routines). This allows you to see all the modules in the project as a selectable… creating this simple picker was extremely tedious, but makes your life as the customer a lot easier. Each module that you create will be automatically detected by this picker, due to which you can easily select it automatically on the scriptable object, fascinating isn't it?

Customizing

This asset pack is mainly intended for you and your studio / project to have a great baseline. This means it will help you getting started by offering basics for pretty much all mandatory elements and common routines. You don't need to be an expert to create your own NPC with this pack, however you still do need some degree of programming and design knowledge if you want to achieve the best results.

Per project it might vary what kind of components you might need. This implies that you might need a custom module that is only relevant to you and your project. The most prominent way of customizing this asset pack is by creating your own module or by editing the existing ones. Each module has its own chapter in this manual describing the basic premise. The script files contain commentary where needed and some times TIPs to help you out.

At the Expanding chapter I will explain in much more detail how you can add your own module.

In order to customize everything you need to be wary that this asset pack works with interfaces and abstract classes. As long as you inherit everything from those 2, then it means that dependant classes can still gather the right data. You can easily plug your own methods, callbacks, variables, and much more to everything. This entire asset pack was designed to be as modular as possible.

The tools themselves are also modular and editable. The lines of code present in each script should give you a general understanding of what is happening where.

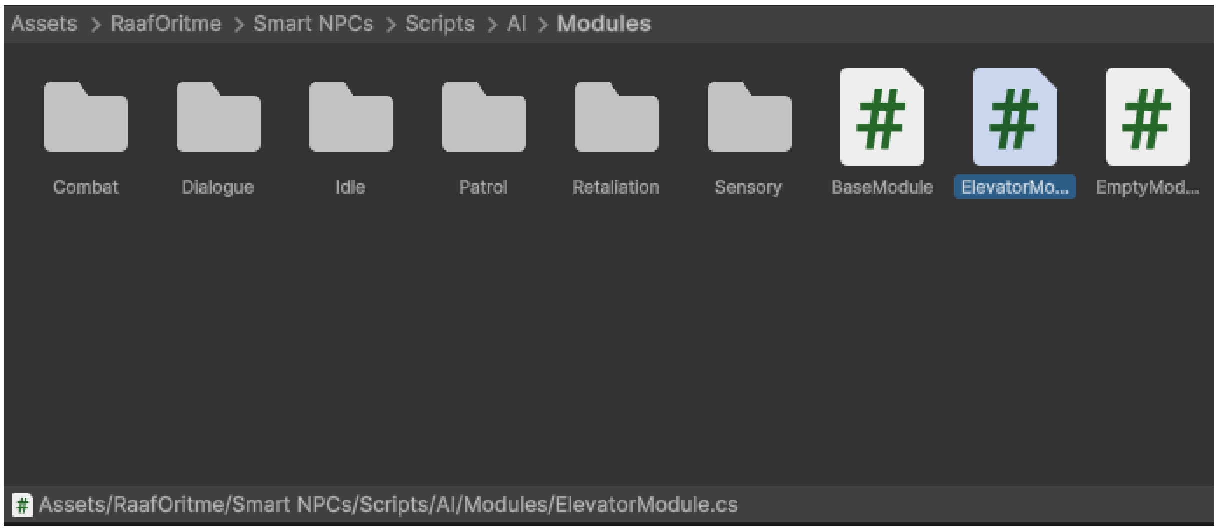
Besides as long as you don't get any errors when hitting the play button, your customization should be all good to go… 99% of the time.

Expanding

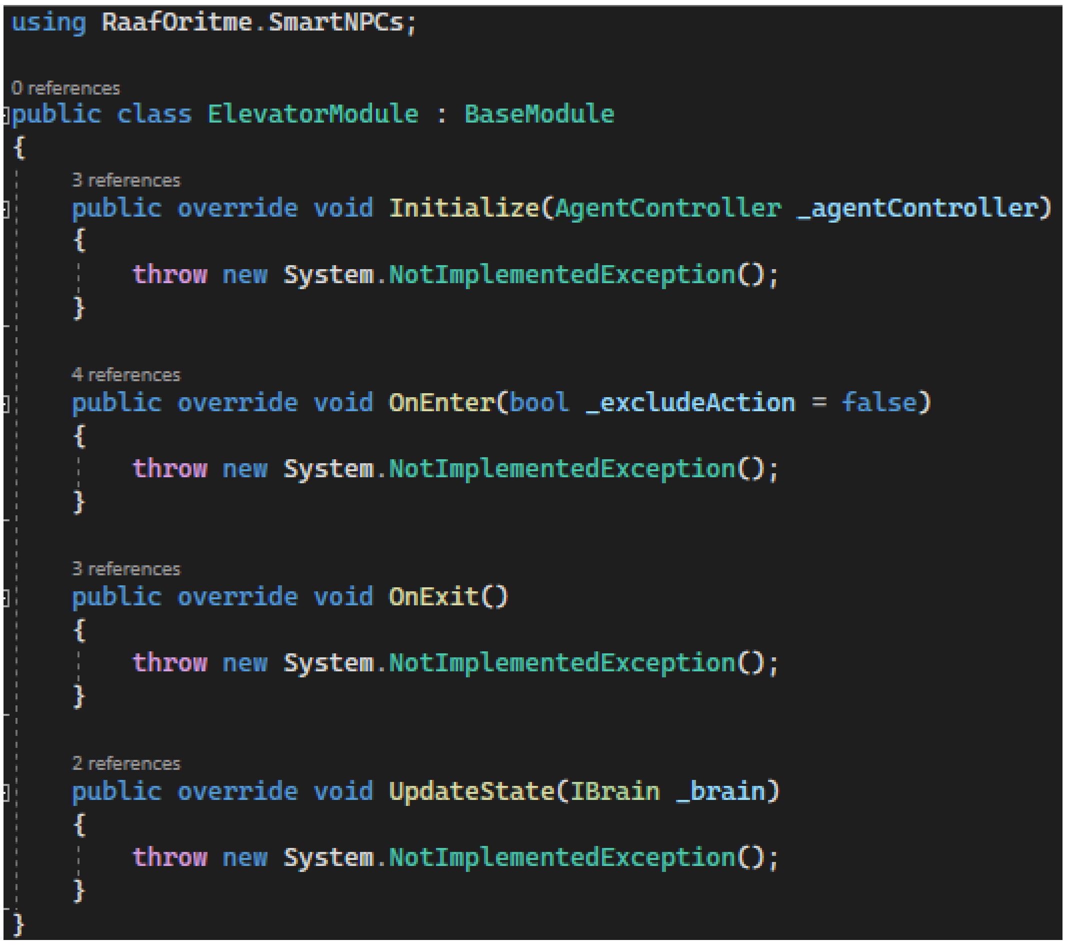
You can expand in this project on the tool itself, the brains, the modules, and possibly even more!

The most straightforward would be to create your own Module to use for NPCs. For example your NPCs require a behaviour that lets them use an elevator to go from place A to B. Especially for GOAP this can be a fantastic addition to make things seem more alive.

Create your script where ever you want, however it is recommended for ease of access to remain within the asset pack standards.

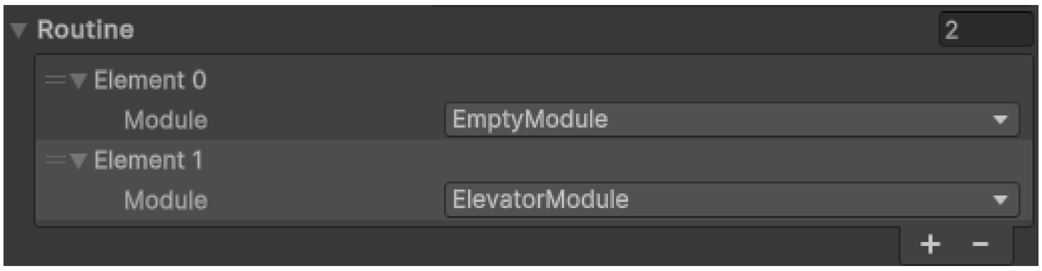

Assets > RaafOritme > Smart NPCs > Scripts > AI > Modules
Combat
Dialogue
Idle
Patrol
Retaliation
Sensory
BaseModule
ElevatorMo...
EmptyMod...
Assets/RaafOritme/Smart NPCs/Scripts/AI/Modules/ElevatorModule.cs

Make sure to use the right namespace and inherit the BaseModule. Implement the abstract methods, and done!



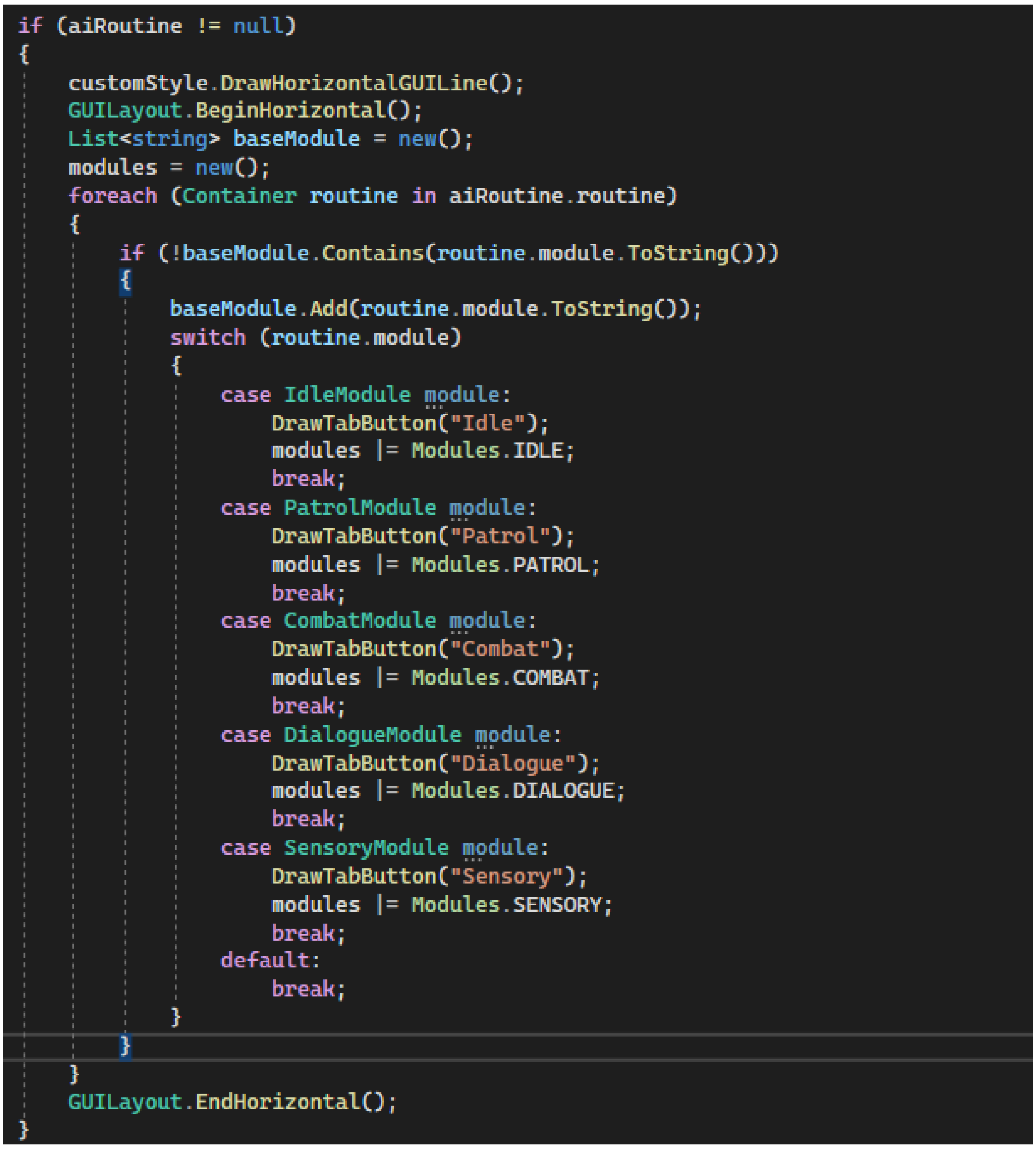
```
using RaafOritme.SmartNPCs;

0 references
public class ElevatorModule : BaseModule
{
    3 references
    public override void Initialize(AgentController _agentController)
    {
        throw new System.NotImplementedException();
    }

    4 references
    public override void OnEnter(bool _excludeAction = false)
    {
        throw new System.NotImplementedException();
    }

    3 references
    public override void OnExit()
    {
        throw new System.NotImplementedException();
    }

    2 references
    public override void UpdateState(IBrain _brain)
    {
        throw new System.NotImplementedException();
    }
}
```

Obviously you need to fill in the empty methods, but other than that it can now be used by the brains.



Now you might want to add settings and a window as well so that it shows up in the NPC Generator tool and in the Agent Controller component.

For the NPC Generator you should head to the StandardNPCGenerator.cs in the DrawModules method there is a foreach that checks what component is what. Here you want to add your new Module as a new case based on the others. While at it, also add it to the Modules enum (go to the file by doing CTRL+LMB on the Modules text.

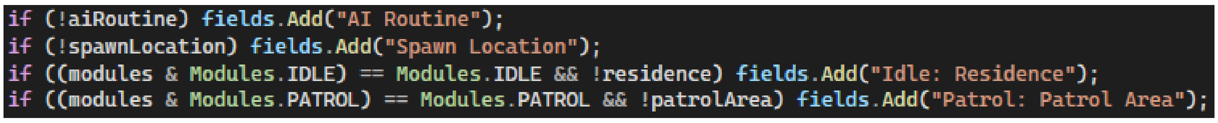


```
if (aiRoutine != null)
{
    customStyle.DrawHorizontalGUILine();
    GUILayout.BeginHorizontal();
    List<string> baseModule = new();
    modules = new();
    foreach (Container routine in aiRoutine.routine)
    {
        if (!baseModule.Contains(routine.module.ToString()))
        {
            baseModule.Add(routine.module.ToString());
            switch (routine.module)
            {
                case IdleModule module:
                    DrawTabButton("Idle");
                    modules |= Modules.IDLE;
                    break;
                case PatrolModule module:
                    DrawTabButton("Patrol");
                    modules |= Modules.PATROL;
                    break;
                case CombatModule module:
                    DrawTabButton("Combat");
                    modules |= Modules.COMBAT;
                    break;
                case DialogueModule module:
                    DrawTabButton("Dialogue");
                    modules |= Modules.DIALOGUE;
                    break;
                case SensoryModule module:
                    DrawTabButton("Sensory");
                    modules |= Modules.SENSORY;
                    break;
                default:
                    break;
            }
        }
    }
    GUILayout.EndHorizontal();
}
```

Below there you will also find a switch case in which you also have to add the new option including a method for drawing it.

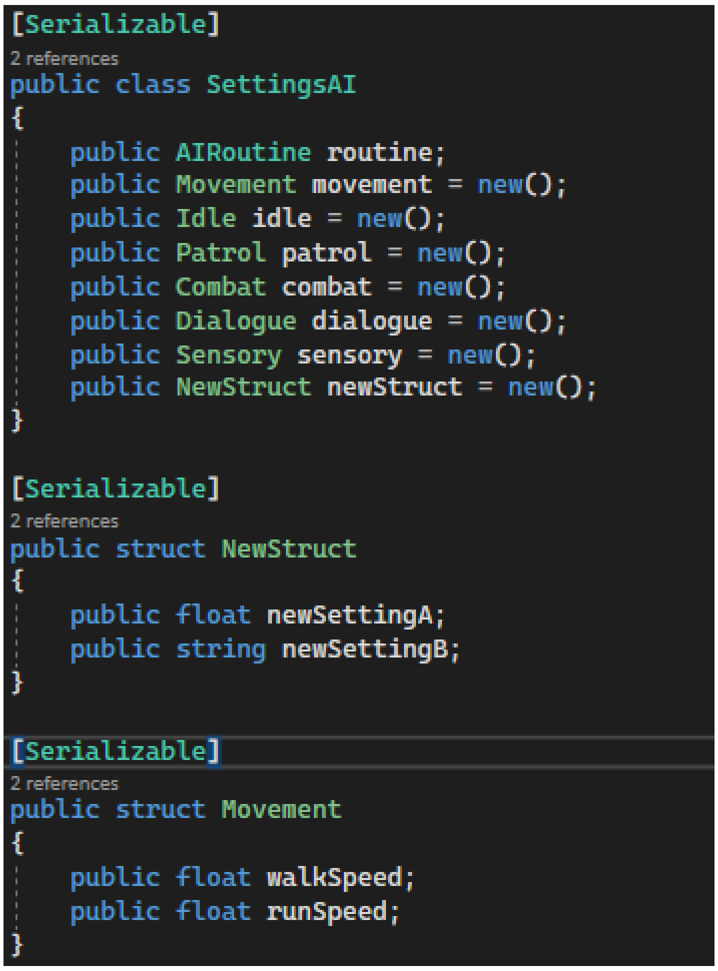
The drawing is done in BaseNPCGenerator.cs look at the example methods and create your own with the settings that you need.

In the StandardNPCGenerator.cs there is also the method DrawErrors. Here you can add the fields that are mandatory for your new module. If a field is missing it will be marked as red, make sure to check if your module was selected or not.



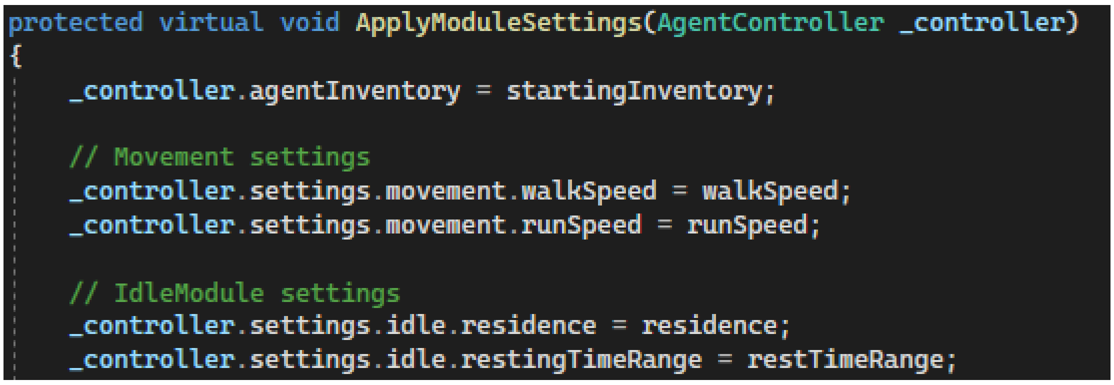
```
if (!aiRoutine) fields.Add("AI Routine");
if (!spawnLocation) fields.Add("Spawn Location");
if ((modules & Modules.IDLE) == Modules.IDLE && !residence) fields.Add("Idle: Residence");
if ((modules & Modules.PATROL) == Modules.PATROL && !patrolArea) fields.Add("Patrol: Patrol Area");
```

Now once you hit generate all the settings should be applied to the new NPC. Go to SettingsAI.cs and add your fields in there as a new struct. Please make sure to tag them as serializable.



```
[Serializable]
2 references
public class SettingsAI
{
    public AIRoutine routine;
    public Movement movement = new();
    public Idle idle = new();
    public Patrol patrol = new();
    public Combat combat = new();
    public Dialogue dialogue = new();
    public Sensory sensory = new();
    public NewStruct newStruct = new();
}

[Serializable]
2 references
public struct NewStruct
{
    public float newSettingA;
    public string newSettingB;
}

[Serializable]
2 references
public struct Movement
{
    public float walkSpeed;
    public float runSpeed;
}
```

Lastly in BaseNPCGenerator.cs you should find the method ApplyModuleSettings and add your own module settings to it. Look at the examples in there to understand how you can add your own. The order doesn't matter.



```
protected virtual void ApplyModuleSettings(AgentController _controller)
{
    _controller.agentInventory = startingInventory;

    // Movement settings
    _controller.settings.movement.walkSpeed = walkSpeed;
    _controller.settings.movement.runSpeed = runSpeed;

    // IdleModule settings
    _controller.settings.idle.residence = residence;
    _controller.settings.idle.restingTimeRange = restTimeRange;
```

That's it! The NPC Generator now contains all the fields, modules, and whatever else you have added, after generating your NPC contains these as well.

The NPCGenerator.cs contains samples as well about how you can add your own window to it, maybe you might need a window with default aggressive settings, and one with maybe no settings at all.

Coding Standards

In the codebase you can search for all “TIP:” comments. These tips give you an example or a heads-up of places to expand upon.

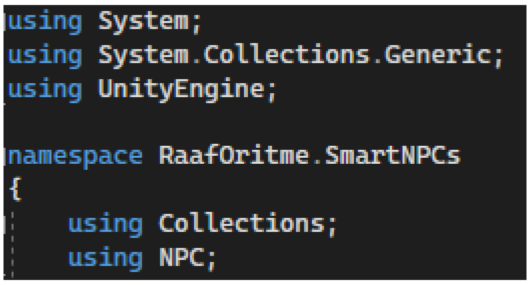
All “RO TODO” comments contain information about what will be done and added by RaafOritme in the near future.

Always include the access level of a method, property, etc. e.g. private, public, internal, etc.

Put classes in a logical namespace.

Always put used namespaces in alphabetical order (CTRL + R + G on Windows in Visual Studio).

Using namespaces within the same higher level namespace can be simplified.



```
using System;
using System.Collections.Generic;
using UnityEngine;

namespace RaafOritme.SmartNPCs
{
    using Collections;
    using NPC;
```

Unity namespaces are used on top of the file.

Public methods should always have a summary emphasizing what the method does, requires, and potentially returns. The only exceptions here should be overrides from abstract classes if it fits within the abstract description.

Methods, properties, and classes use Pascal Casing. E.g. ExampleClass();

All others use Camel Casing. E.g. exampleVariable;

Parameters start with an underscore followed by Camel Casing. E.g. _exampleParameter;

At the end of every code file should be an empty line.

FAQ / Common issues

Q: I keep getting a null reference error and my Routine keeps getting reset, how can I resolve this?
A: This is most likely due to a change that has taken place in the ScriptableObject script itself. When you change a SO you need to make these SOs over again. You can screenshot the previous settings and then manually add them on the new ones. This is unfortunately a short coming, but all the benefits it delivers makes it passable.

To be released

This asset pack comes with the promise of certain additions that will be released over time.

These features / additions will be released over time:

- A more comprehensive and in-depth debugger.
- Basic combat module.
- Basic dialogue module.
- Sensory module.
- Retaliation module.
- Quality of life updates and improvements.