FSM AI Template (v1.1.0 - 03/12/2019)

Thank you for supporting this asset, we develop this template because a lot of developers have good ideas out there but creating an AI system is really complex takes too much time.

The goal of this project was always to deliver a top quality AI that is easy to use for new developers and expandable and customizable for advanced users.

It comes with Civilian, Melee Combat and Shooter AI examples, and also with several examples of different FSM Behaviours.

With this template, you can set up a 3D Model in just a few seconds, without the need of knowing hardcore code or wasting time dragging and drop gameobjects to the inspector, instead you can just focus on making your game.

The template works great in your own project using VR or FPS and it works even better when using together with the Invector Third Person Template with all of our features.

- Invector Team

In case you didn't find what you're looking for here, please try your [Youtube] page we have several video tutorials showing how things work, or contact us at inv3ctor@gmail.com

Summary

USING THE FSM AI AND THE THIRD PERSON TEMPLATE

If you're using our **Third Person Controller Template** (*Basic Locomotion, Melee Combat or Shooter*) make sure that you're using the **latest version** available for both templates, otherwise it may present compatibility issues.

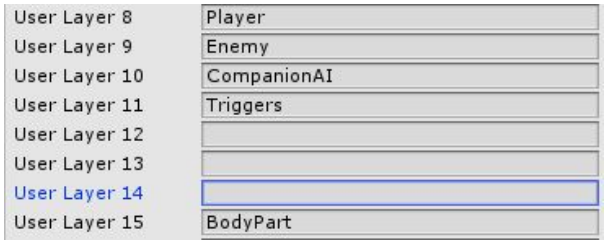
Third Person Templates 2.5.0 goes with FSM AI 1.1.0
Basic & Melee 2.4.2 - Shooter 1.3.2 goes with FSM AI v1.0.1
Basic & Melee 2.4.1 - Shooter 1.3.1 goes with FSM AI v1.0
Basic & Melee 2.4.0 - Shooter 1.3.0 goes with FSM AI v0.3
Basic & Melee 2.3.3 - Shooter 1.2.3 goes with FSM AI v0.2

> Import **first** the Third Person Template **than** the FSM AI Template.

LAYERS AND TAGS

After you import the AI Template on a Clean New Project, you will notice that the AI prefab is actually missing the layers.

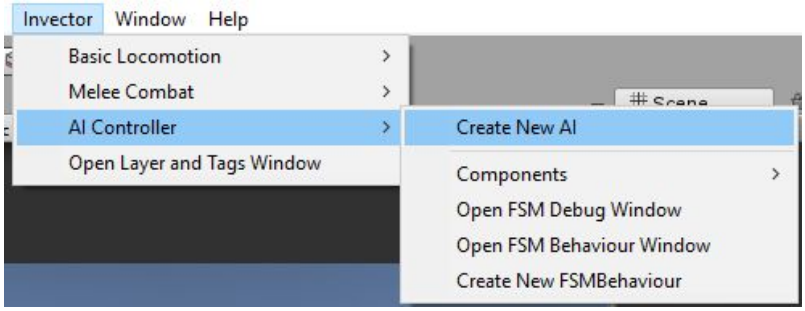
This will not prevent you from testing the demo scenes or cause any errors, but you must add the Layers in a specific order if you want to display correctly.



If you do not add the layers in this specific order, the layers will be displayed but with the wrong layer in the prefab, you can also use the layers of your going on Project or add it separately.

CREATING A NEW FSM AI CONTROLLER

To create a new AI Controller, go to:

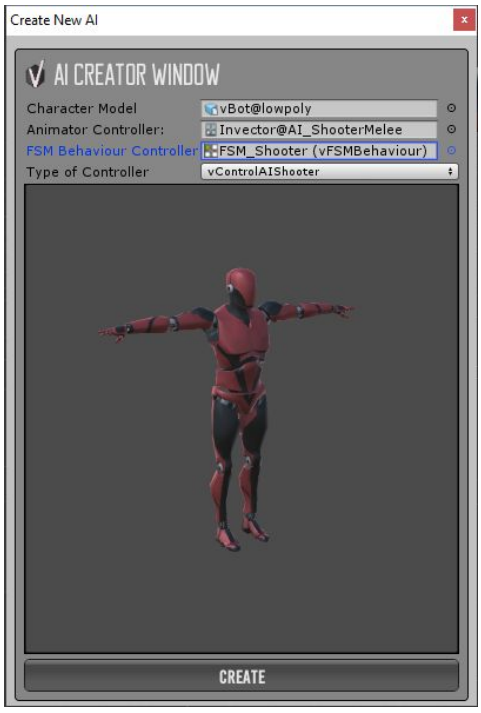


This window will pop up and you need to select:

1- Your 3D Rigged FBX Model

2- The Animator Controller (we provide 3 examples, Basic Locomotion (for passive civilians), Melee Combat and Shooter)

3- The FSM Behaviour you want to use, we provide several examples in the package but you can (and must) create your own once you learn how to use the tool.



Once you hit the "Create" button, the Character Creator Window will add every necessary component to your Character like:

- Animator Controller
- Rigidbody
- Capsule Collider
- NavMesh Agent
- FSM Behaviour Controller
- AI Controller
- Melee Manager (If you choose to create a Melee AI)
- Shooter Manager (If you choose to create a Shooter AI)
- Headtrack is automatically added to the Shooter AI, but you can also add it manually on both Civil or Melee to look at the target or at a specific object.

HOW TO BAKE A NAVMESH

The AI Template uses the NavMesh & NavMesh Agent to navigate, so you need to bake the navmesh in your scene for the AI to actually move and find path to navigate.

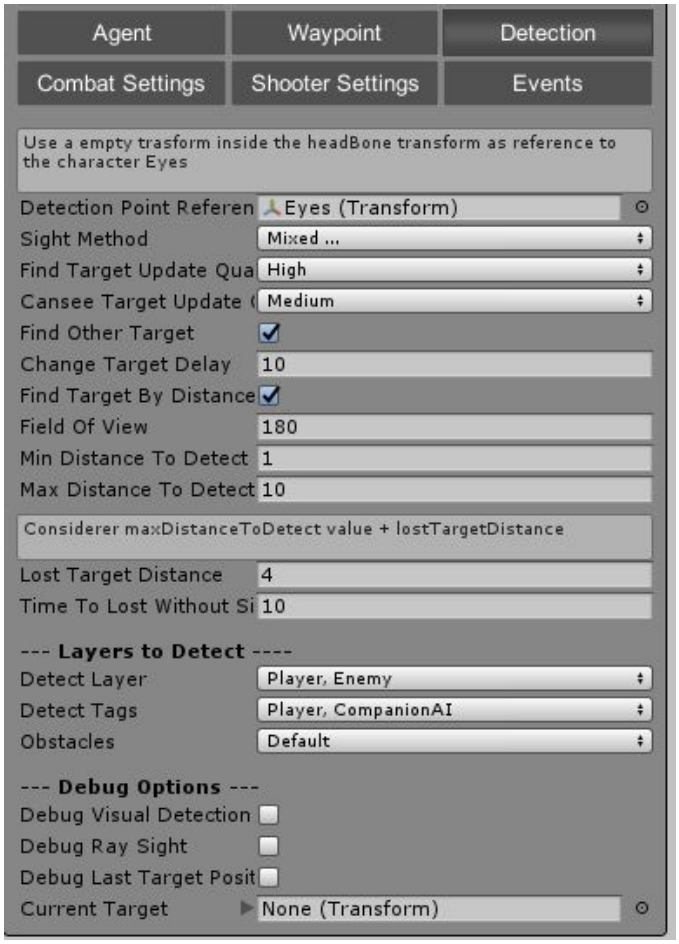
There is plenty of tutorials teaching how to Bake the Navmesh, here is one by Unity:

(Unity 5.x) https://unity3d.com/learn/tutorials/topics/navigation/navmesh-baking

(Unity 2017+) https://docs.unity3d.com/Manual/nav-BuildingNavMesh.html

HOW TO DETECT A TARGET

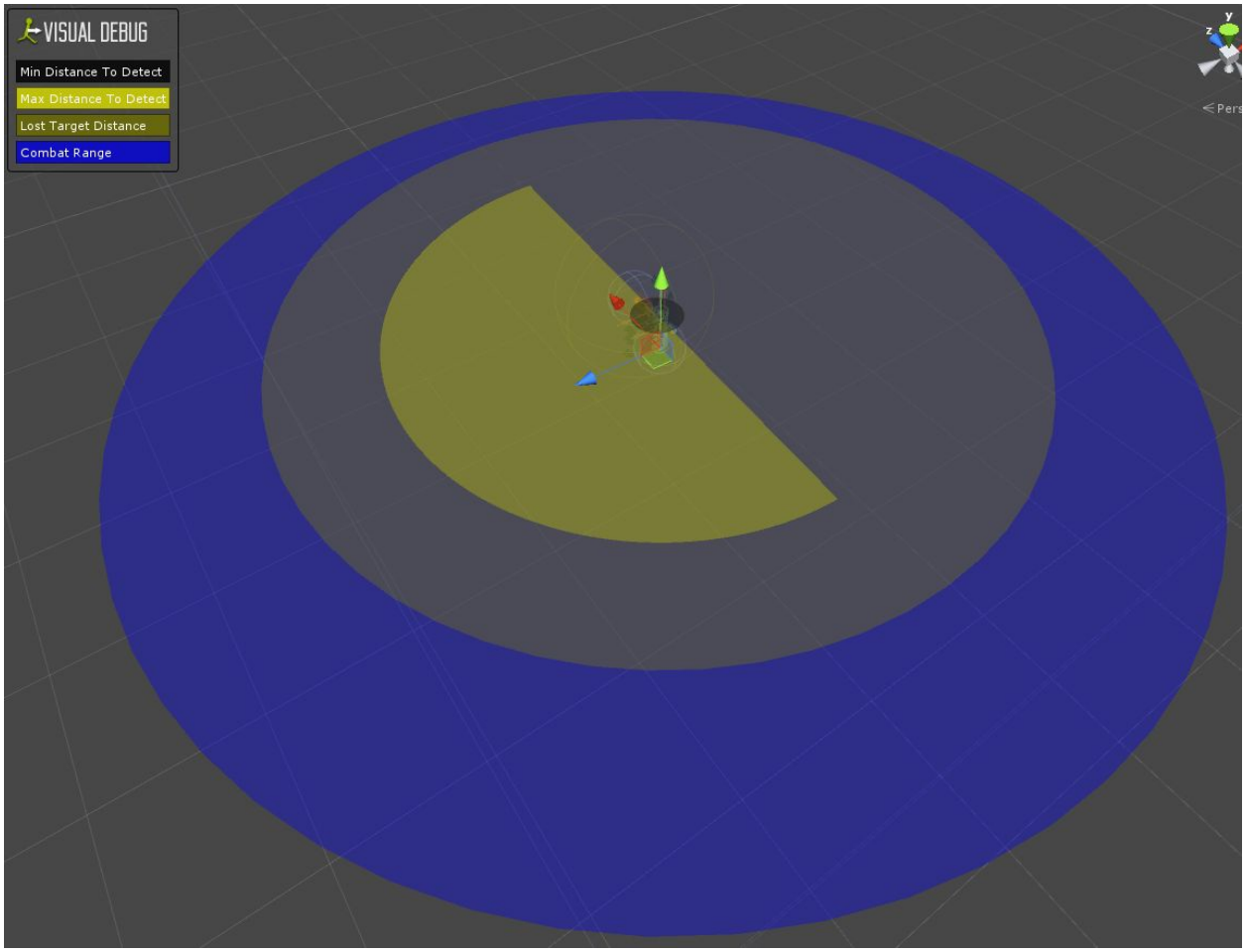
The most important settings here are the **Layers to Detect**, make sure to add the Layer and Tag of the target you want the AI to Chase, your target must have a **Collider**.



*Notice that the Detection Settings is to detect a target only, tin order to apply damage to a target you need to set this information at the MeleeManager or ShooterManager.

- Detection Point Reference: You can create an empty gameObject, place it in the character's head bone and use the forward direction as a reference for the 'eyes', this way if you have an Idle Animation were the character is looking around searching for a target, it will be more believable.
- Sight Method: You can choose between Top, Center, Bottom or All. This will trigger raycast to the target when close enough, you can debug this option by checking the Debug RaySight
- Find Target Update Quality: Frequency to update the find target method, the higher you choose more precise will be the detection, also it will be more expensive on performance.
- CanSee Target Update Quality: Frequency to update the CanSee method, to know if you still can see the target, the slower it is, the slower he will see or loose the target.
- Find Other Target: Can still find other targets even if already has one.
- **Change Target Delay**: Delay to change target.

- Find Target By Distance: If uncheck, the AI will get the first target he sees, and not the closest one.
- Field of View: Check the "Debug Visual Detection" option to see ingame.
- Min Distance To Detect: Min distance for the AI to detect the target, even without seeing it. Leave it to 0 if you want to do stealth attacks.
- Max Distance To Detect: Max distance to detect the target
- Lost Target Distance: This value will be the MaxDistanceToDetect+LostTargetDistance to lose the target.
- Time to Lost Without Sight: How long it will take to actually lose the target if you can't see it.

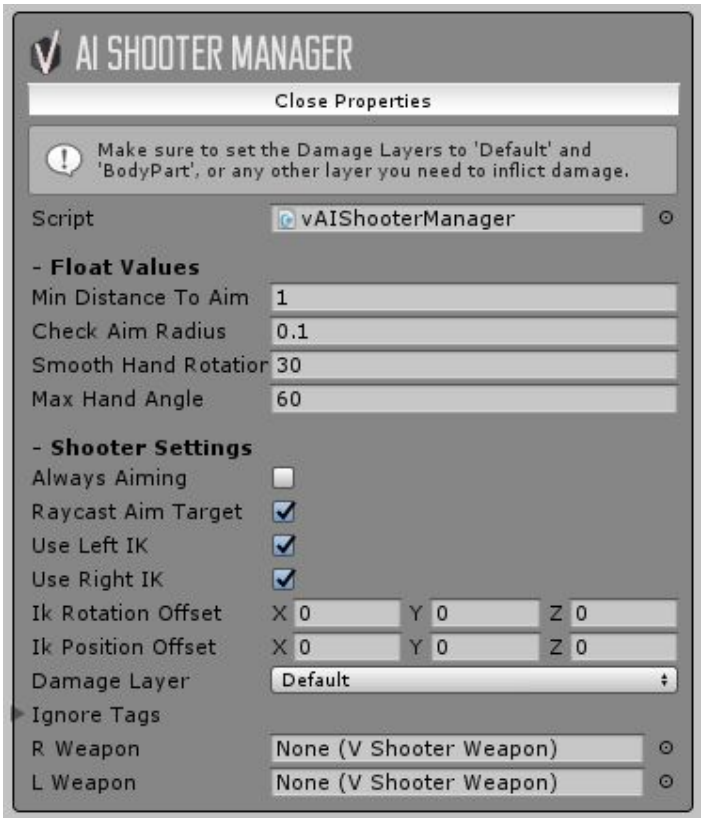


- Detect Layer: Layers to detect a target (Target must have a Capsule Collider)
- Detect Tags: Tags to detect a target
- Obstacles: Usually set to "Default"

HOW TO APPLY DAMAGE TO A CUSTOM TARGET

The target must have a **vHealthController** and a **Capsule Collider** so the AI can actually apply damage to it and know if it's alive, if you're target is using another system with another health system you will need to create a bridge to apply damage to this new system.

Shooter: You need to set the **DamageLayer** of your target in the **ShooterManager**.

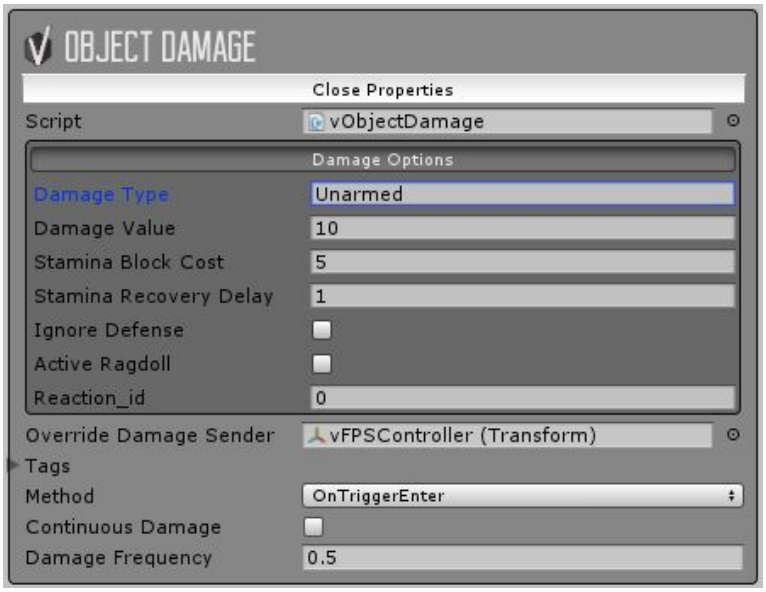


MeleeCombat: You need to set the Tag of your Target in the MeleeManager / HitDamageTags field, you can assign several tags to hit different targets.


MELEE MANAGER
Close
Script
vMeleeManager
Open Default Info
Open Events
Add Extra Body Member
Body Members
LeftLowerArm
RightLowerArm
LeftLowerLeg
RightLowerLeg
Who you can Hit?
Hit Properties
Hit Damage Tags
Size
1
Element 0
Enemy
Use Recoil
Draw Recoil Gizmos
Recoil Range
90
Hit Recoil Layer
Default
Weapons
LeftWeapon
None (V Melee Weapor
RightWeapon
None (V Melee Weapor

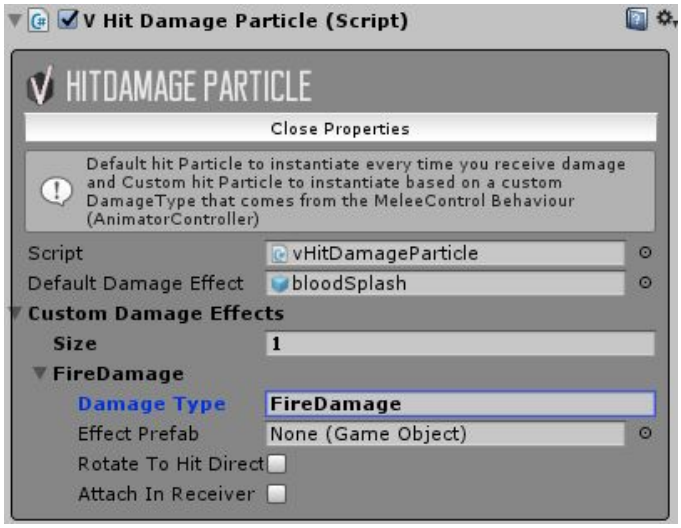
HOW TO APPLY DAMAGE TO THE AI

We have a few examples on how to apply damage to an AI Controller, basically you need the **vObjectDamage** script which is attached to the Pendulum or Hitboxes in the "AI_Examples" demo scene.



- **DamageType**: Used together with the HitParticleDamage component, you can trigger different particles for different type of damage.

For example, if you have an area with fire, you can add a **vObjectDamage** there and add a DamageType of "FireDamage", then add a Custom Damage Effect with the same DamageType to your HitDamageParticle on your AI Character and add the particle effect to burn your character.

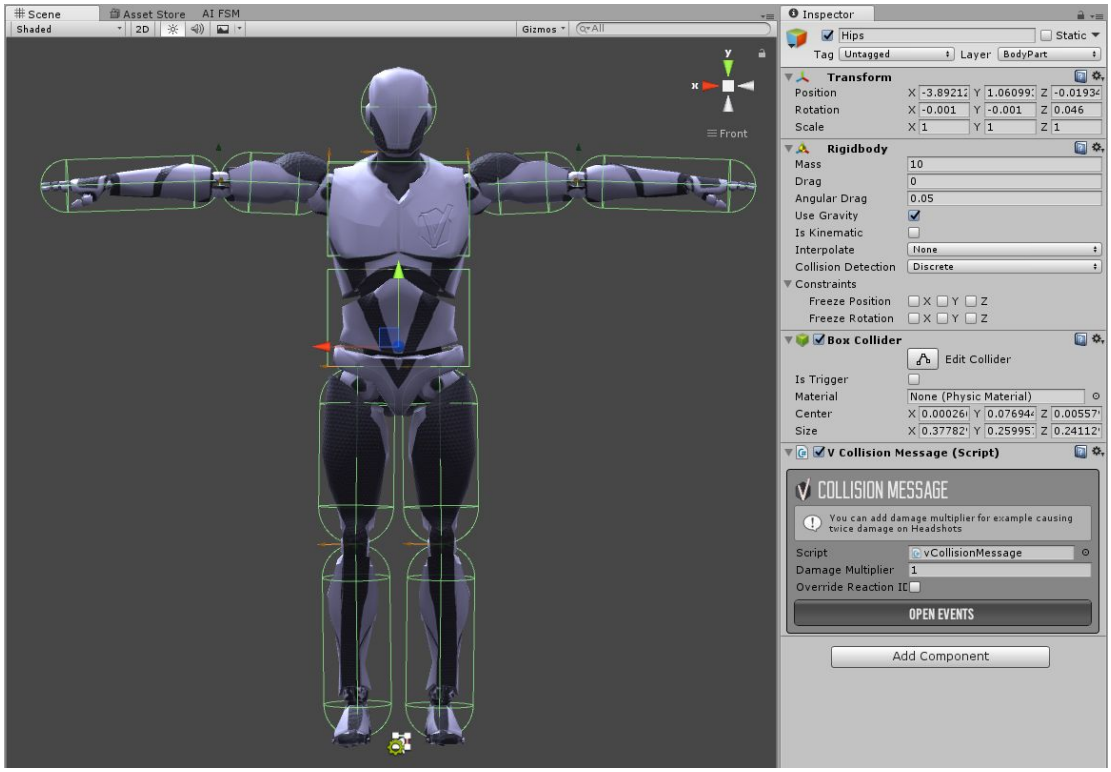


This component is attached to the AIController or any object that contains the **HealthController*

- **DamageValue**: How much damage it will be applied to the vHealthController of the target.
- **Stamina Block Cost**: You can ignore that option, it's only for the ThirdPersonController.
- **Stamina Recovery Delay**: You can ignore that option, it's only for the ThirdPersonController.
- **Ignore Defense**: If you're using a MeleeCombat Controller, it will ignore the defense and apply damage anyways.
- **Active Ragdoll**: It will active the ragdoll on your character, if it has one.
- **Reaction ID**: You can trigger specific hit reaction animation, you can use -1 if you don't want to trigger any animation.
- **Override Damage Sender**: Assign the root object otherwise the AI will target the object that has this component instead. For Example: If you apply the vObjectDamage to be a Hitbox of a LeftHand of a character, the vHealthController or AI will have the LeftHand as the target instead of the GameObject parent.
- **Tags**: What tags you will apply damage to
- **Method**: OnTriggerEnter or OnCollisionEnter
- **Continuous Damage**: Useful for fire damage for example
- **Damage Frequency**: Frequency to apply the damage, if Continuous Damage is enabled.

Using the **Ragdoll Colliders** as **BodyParts** to inflict precise damage.

SHOOTER > If you want to cause damage for each body member using the ragdoll colliders, UNCHECK the “Disable Colliders” and you can add damage multiplier on each member.



* You must use a different Layer in the Ragdoll Colliders like “BodyPart” and another for the main capsule collider like "Enemy", this way the Detection will detect the Enemy object as a target and chase it, but the ShooterManager will actually apply damage to the "BodyPart", allowing to apply damage on different parts of the body.

Body Parts use the Damage Receiver to receive damage.

A **DamageReceiver** is attached to each ragdoll collider, this will allow to Player or AI to apply damage to each bodypart instead of the CapsuleCollider.

- **Damage Multiplier**: multiply the damage value
- **Override a Reaction ID**: Check this option to override the hit reaction animation to trigger a specific animation.

For example, if you want to cause 2x damage and trigger a specific reaction animation when shooting in the Head, simple change the values in this component.

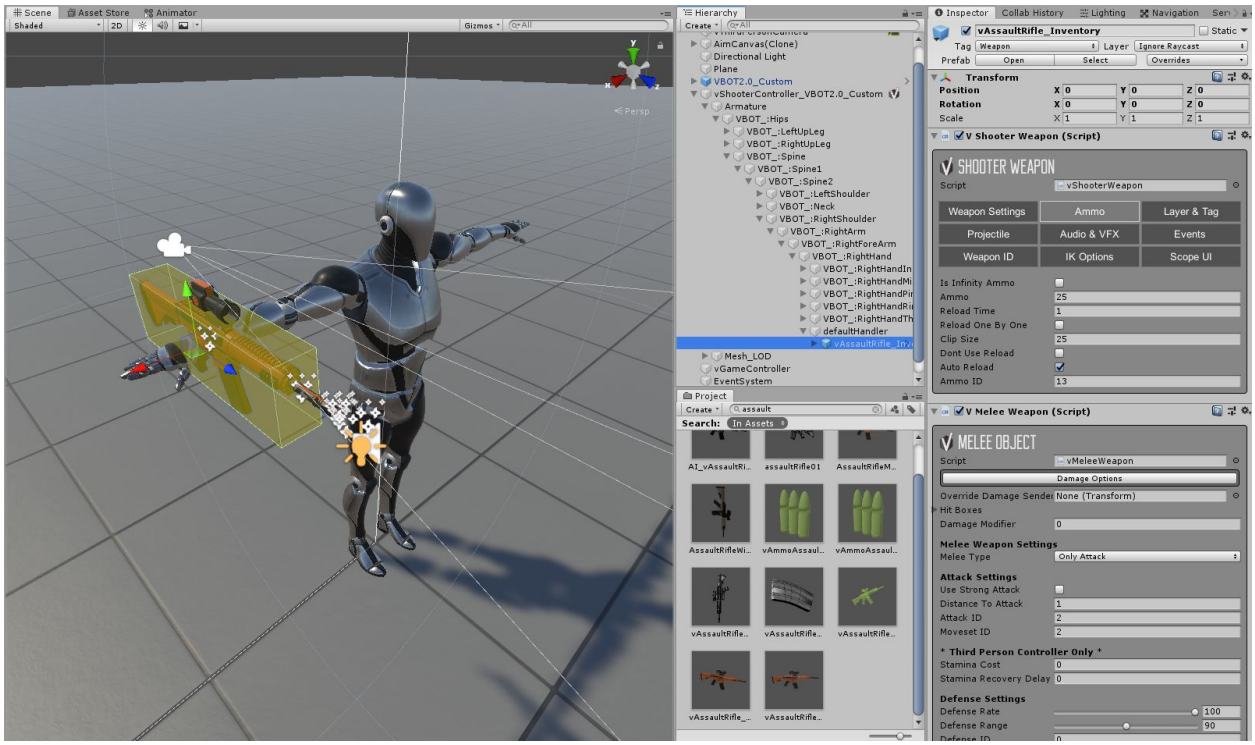
HOW TO ALIGN A SHOOTER WEAPON (RIGHT HAND)

After creating your character, you will need a *'handler'* to manage the position/rotation of the weapons you instantiate in game.

A **Handler** is basically an empty transform located inside your character's left and right hands, you will first align this handler into a position where a weapon looks correct in the characters hand and all weapons will be instantiated there when you pick up the collectible.

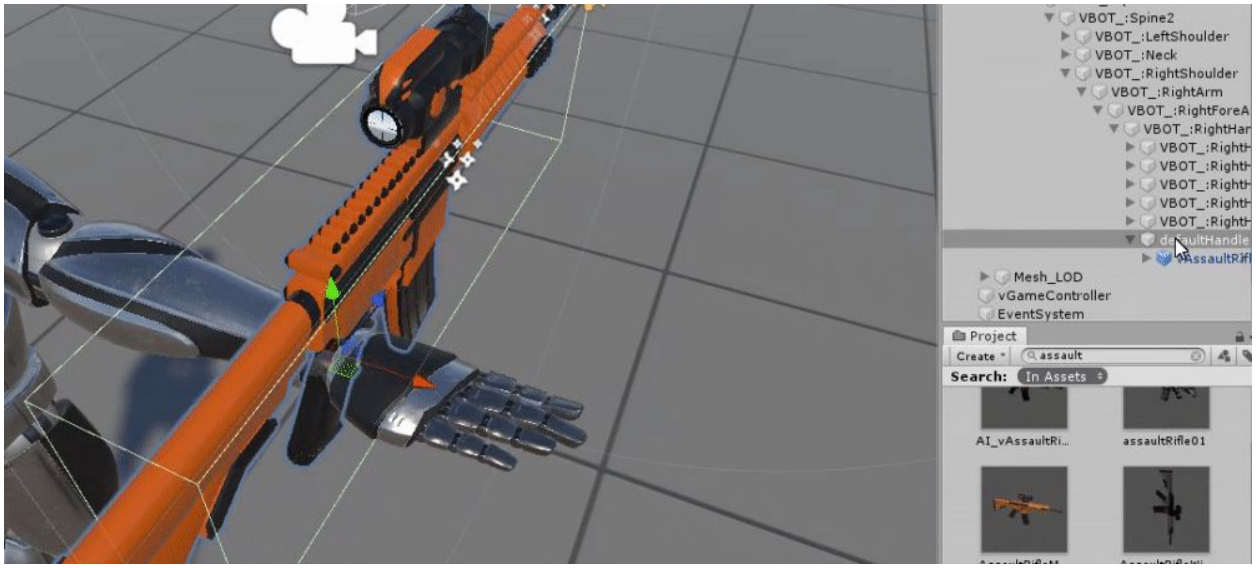
Shooter Weapons will be equipped in the **RightArm defaultHandler by default**, so select this transform in the hierarchy and drag and drop any Weapon Prefab inside the defaultHandler.

For example search in the Project window to a weapon prefab, drag and drop inside the **'defaultHandler"** and reset it's position and rotation back to 0, 0, 0.



Now you can manually align the 'defaultHandler' to a position close to where the weapon should be located.

* **ATTENTION**: You should **NOT** move the Weapon Prefab itself, this gameObject should be at 0,0,0, **you will only move/rotate the Handler**, this way all the weapon prefabs that will be instantiated inside the handler will be aligned as well.



You can now hit **Play** to see if the **weapon position in the RightHand looks good**, it doesn't need to look perfect because we're still going to the **IK Adjustments** where we will **align the LeftHandIK.**

HOW TO ALIGN THE LEFT HAND IK FOR ALL WEAPONS

Go to your ShooterManager inspector and select the tab “IK Adjustment” and hit the button to create a new “IK Adjust List”.

A new IK Adjustment List will be created and automatically assign for you, a new button called “**Edit IK Adjust List**” will appear, click on it and a new window will open, you can dock this window anywhere you like.

Hit **Play** and select the **ShooterController** to see this menu:

Hit **“Create New IK Adjust”** and before we started playing around with the many cool IK options, we first need to **align our LeftHand IK**, you can do this by sliding the **XYZ Left IK Rotation and Position Offsets**.

As you can see in the infobox this offsets **will affect all weapons** just like the Handler position/rotation you previously set up. Now that we have a base alignment we can proceed to explore the **features of IK Adjust**.

Once aligned, you don’t need to change the values of the LeftIK again since you now be able to create adjustments for each weapon and state.

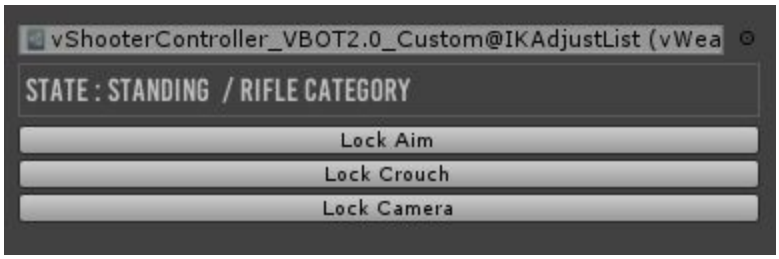
IK ADJUST FOR FINE TUNING OR CREATE NEW POSES

Now that we properly align the **defaultHandler** and the **LeftHandIK** for **all weapon prefabs**, we can start modifying the IK to create more **polished** or **even entirely new poses**, **for each state and each weapon category.**

The “State” will inform what state you are, there are basically 4 states that you can create unique IK modifications, those are:

- Standing
- Standing Crouch
- Aiming
- Aiming Crouch

You can switch the state by pressing the buttons to LockAim, LockCrouch and LockCamera.
Start by Locking the Camera so that the Headtrack doesn’t influence your pose.



You can create different IKAdjusts for each weapon or use the same adjust if the weapon share the same **Category**, for example, if you have a large number of Pistols in your game that use the same pose, you don’t need to create an IKAdjust for each weapon, just set the same Category for all the ShooterWeapon prefabs.

You can type any **Category** directly in the **ShooterWeapon** prefab:

You can create a **Copy** from another **IK Adjust**, for example if you have **2 rifles slightly different from each other**, just make a copy and do the adjustment for the other rifle.

You can also select the gizmos handlers directly from the menu or click on the little sphere/square of each arm.
Or reset the position/rotation by clicking in their respect buttons, the “Reset” button will reset both.

You can also create **Offsets** for the **Spine** and **Head** to help you set up the position.

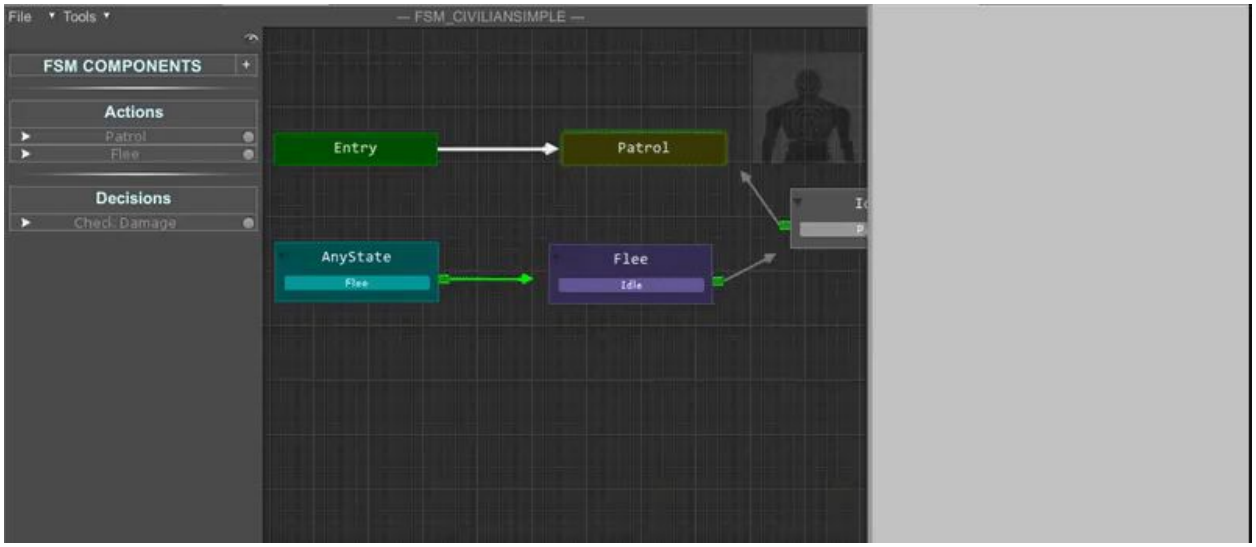
- ***This feature was designed to help you better align the weapon into your Character Pose, but if you want the best result as possible the ideal is to replace the animations to animations created using your Character Rig. We recommend the use of uMotion to quickly and easily adjust the animations to your character's rig.***

AI SEND MESSAGE / LISTENER RECEIVER

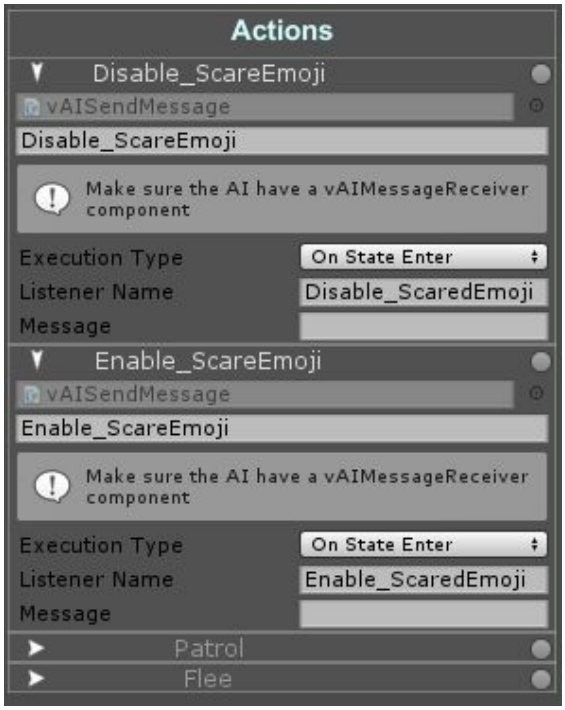
Description: With this feature you can send custom messages from FSM State to the Controller itself and call a public method, trigger a sound, particle, show/hide a gameObject, etc...

Here is a quick tutorial on how it works:

Let's create a SendMessage Action in the FSM, the message will be "Enable_ScaredEmoji" and assign to our Flee State using the OnStateEnter method.

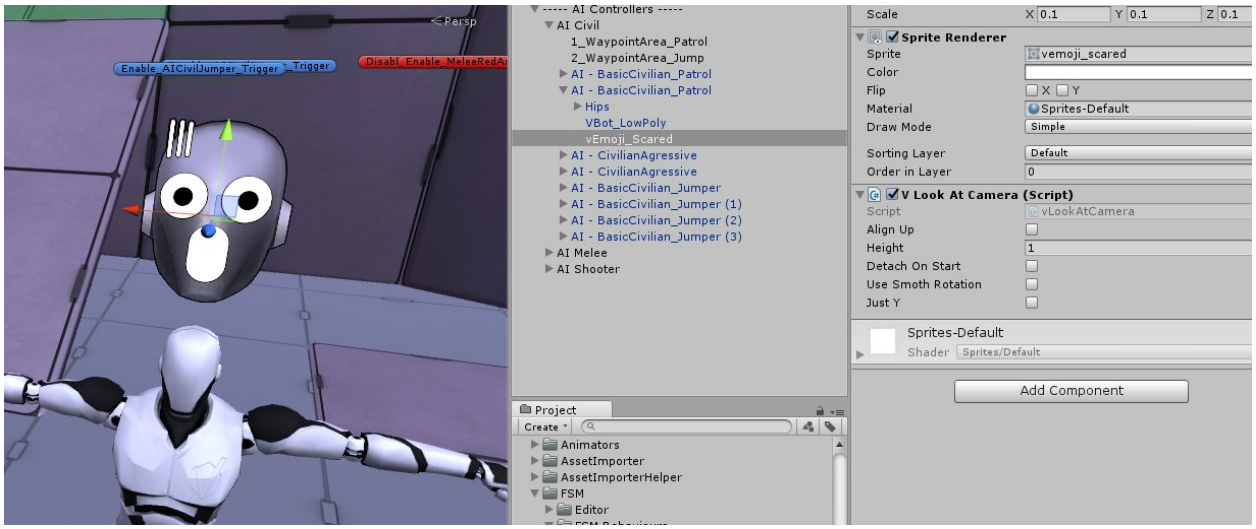


Do the same to "Disable_ScaredEmoji" and assign to the Patrol State.

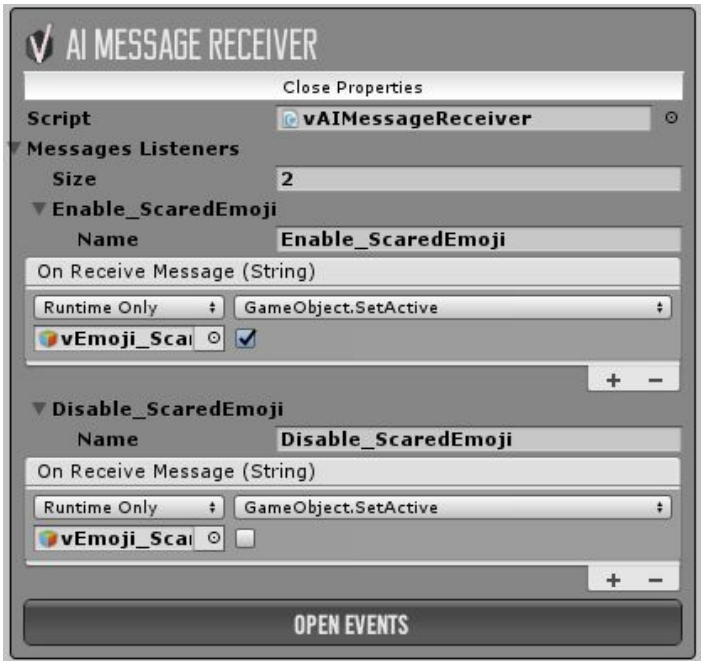


- **Listener Name**: Name of the MessageListener that you will be called in the MessageReceiver on your AI Controller.
- **Message**: You can pass a string parameter to the Event OnReceiveMessage(string)

Now go to your AI Controller and create a GameObject Sprite with the EmojiScared texture:



Add the Component vAIMessageReceiver into your AI Controller and set the messages "Enable_ScaredEmoji" and "Disable_ScaredEmoji". Now simple assign the vEmoji_Scared GameObject and use the Events to turnOn/Off the gameObject.



You can trigger particles, sounds, call public methods, enable/disable objects, etc...

FSM BEHAVIOURS - HOW IT WORKS?

Description: The FSM Behavior window is a Node Editor that you can create customizable new behaviours using Finite State Machine workflow.

The AI Template comes with several **Actions** and **Decisions** build in, they are basically small pieces of code that allows you to verify something or do some behaviour.

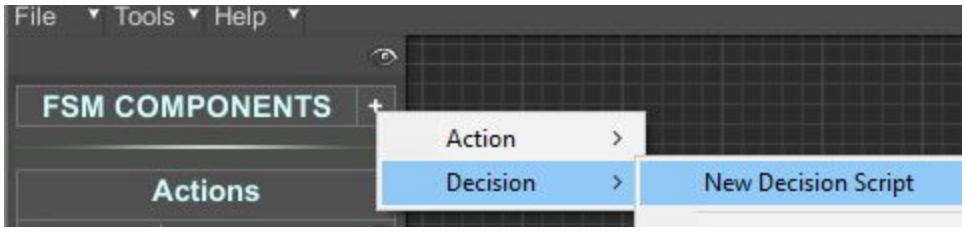
01. **FSM Behavior** : The behavior is the actual entry point into the AI's behavior. The FSM behavior contains the state flow. There are two important states that will be in every behavior. One is of course the Entry and the other one is the Any State.

The "**Entry State**" governs the entry point that creates the "current state", just like the Animator Controller does.

The "**Any State**", is the one that does the decision making to go from the current state, to "any state".

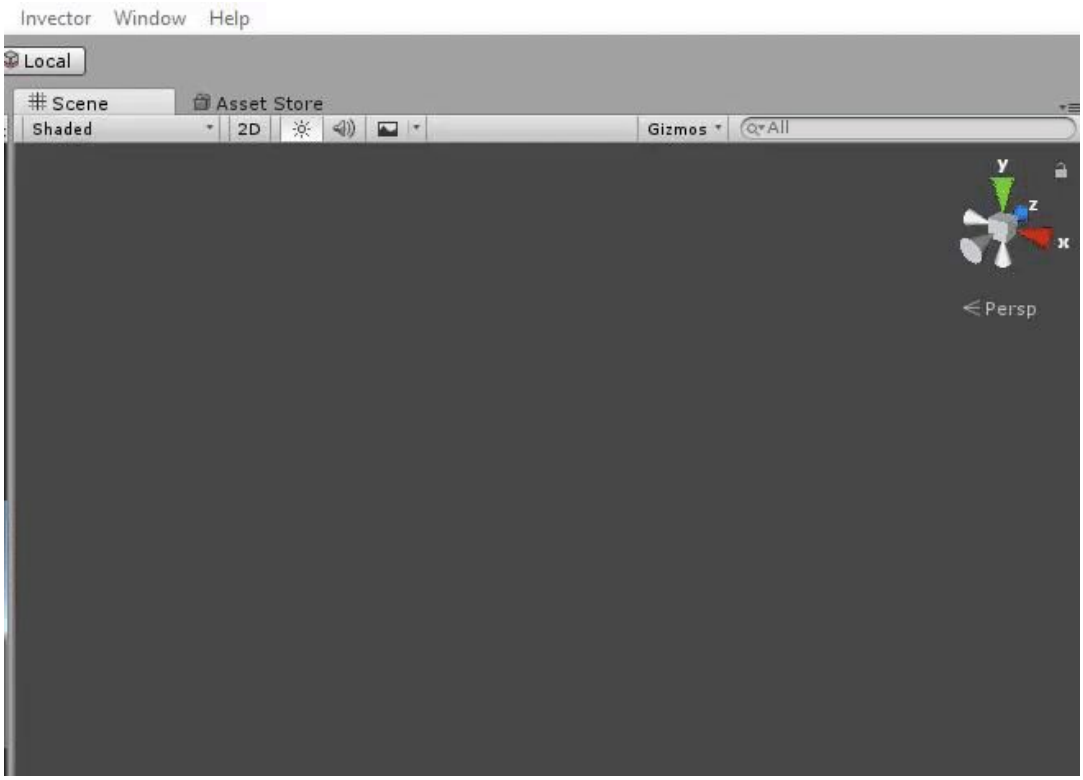
02. **States** (node) : To create a new State or Node right click in the background of the FSM Behaviour Window and hit "New Node". Every state can have Actions and Transitions, every Transition can have Decisions that will return true or false.

You can create your own Custom Actions and Custom Decisions to create more unique behaviors.

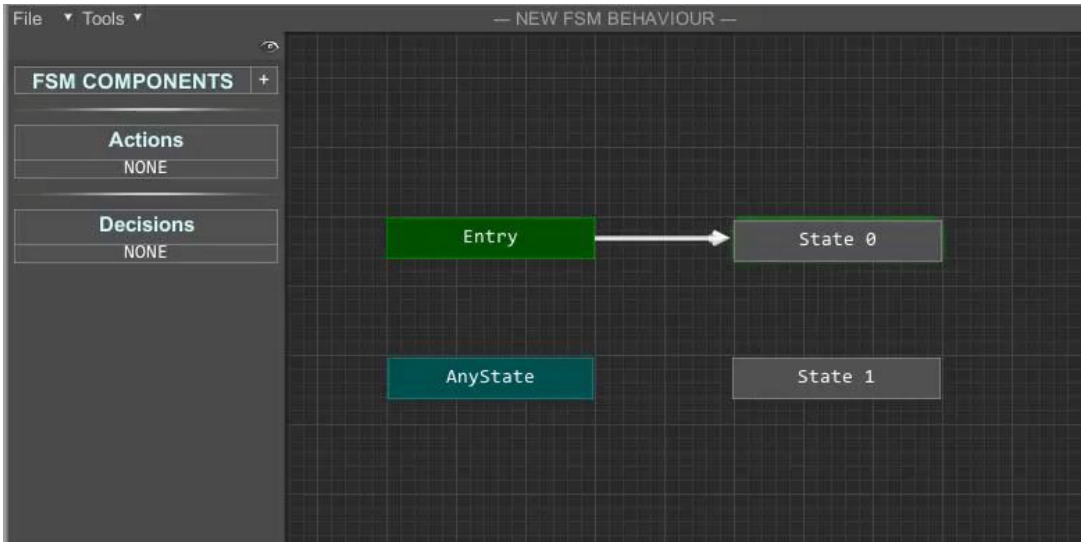


This will create a script template for a *CustomAction* or *CustomDecision*.

Or you can use one of our several actions and decisions built in, which are already pretty useful to create a lot of different behaviors.



03. **Transitions**: To add a new Transition right click in your State and hit the button "New Transition", them open the state folder and click and hold the from the transition point to the next state.
*(*v1.0.2 now automatically draw the arrow, just drag to the next state)*

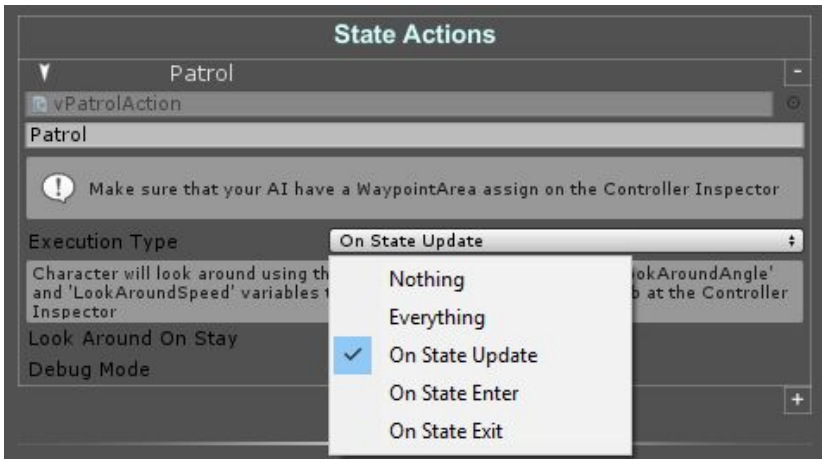


You can add a Transition Delay to your transition to create more interesting results, for example:
The character will go from Searching to Patrol after 10s

05. **Decisions**: You can add Decisions in your FSM Behaviour just like you can add several different types of variables in the Animator Controller, and use them to drive your Transitions. A Decision will return a value of True or False and you can also add a Transition Delay for each transition.

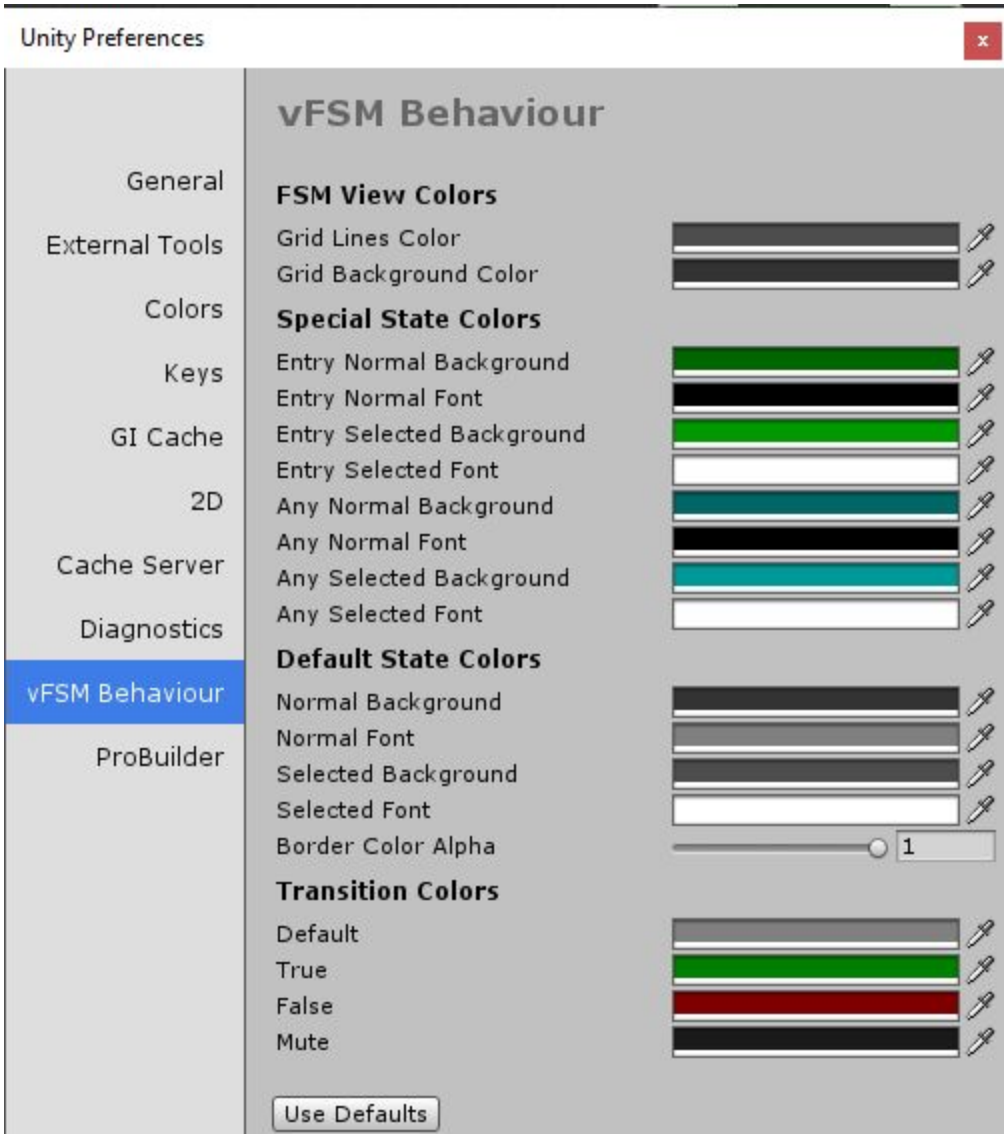
You must first add the Action or Decisions in the FSM Components (left panel) and then add it to your State (right panel), just like the example above.

06. **Actions**: They are scripts with an encapsulated behaviour, for example if you open the vPatrolAction script you will see that it contains only the patrol logic but the variables are actually located on the AI Controller itself, so you can assign different WaypointArea's for each controller, this way you can re-use the Patrol behaviour and the FSM on several different AI's but different values assigned to it.



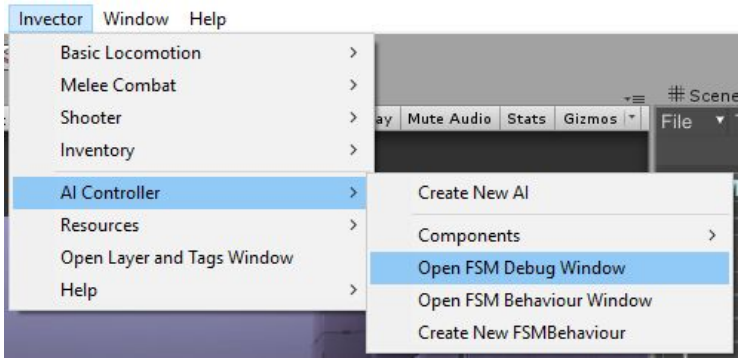
Actions can be executed OnStateEnter, OnStateUpdate, OnStateExit, All or Nothing.

You can also customize your FSM editor colors by going to *Edit > Preferences > FSM*

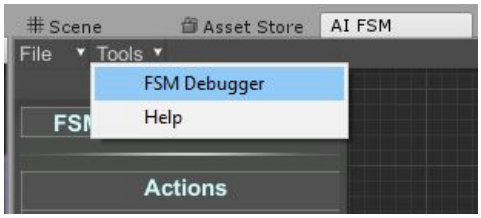


FSM DEBUG WINDOW - HOW TO USE IT

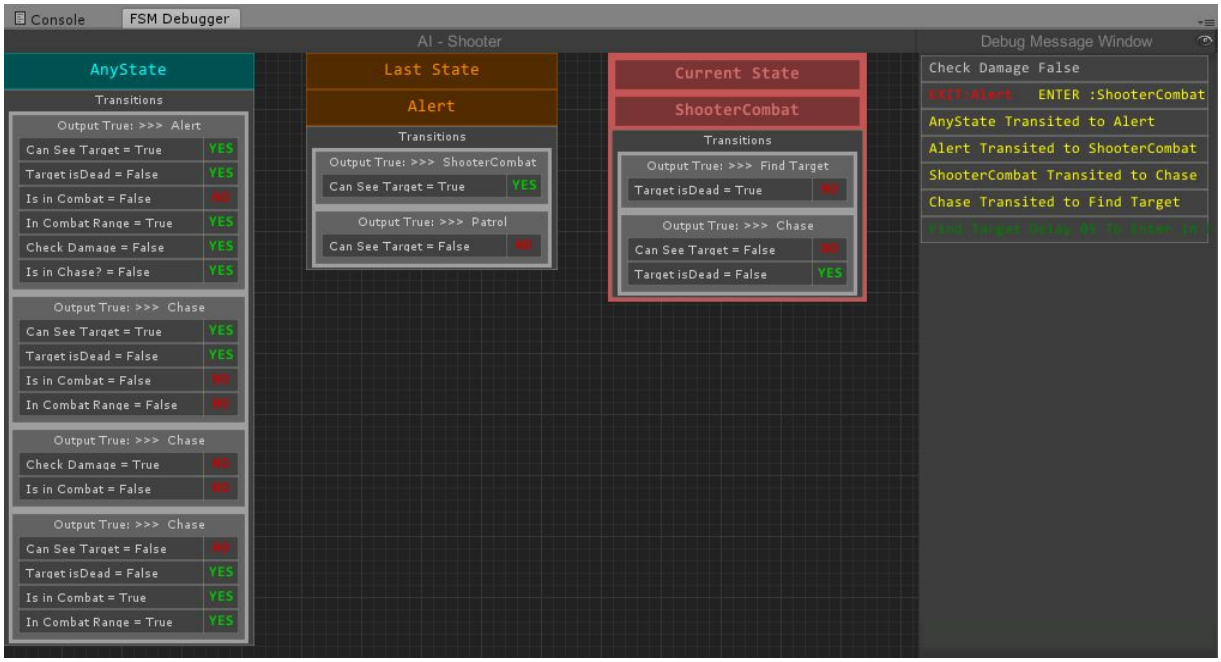
You can open the **FSM Debugger window** by going to:



Or if you have an FSM Behaviour Window opened, go to **Tools/FSM Debugger.**



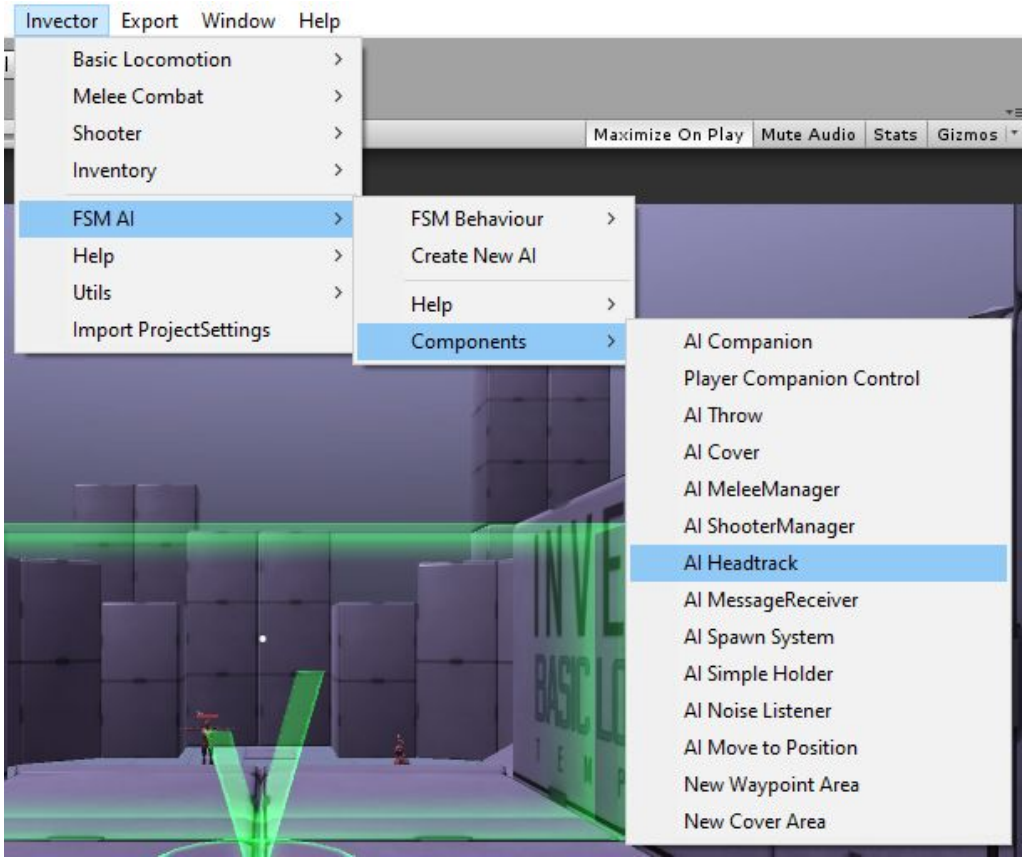
Select the AI Controller in the Hierarchy that you want to debug, and hit play to follow in real time the States, Actions, Decisions and Transitions. This will help you debug your FSM logic and quickly find the solution for your logic.



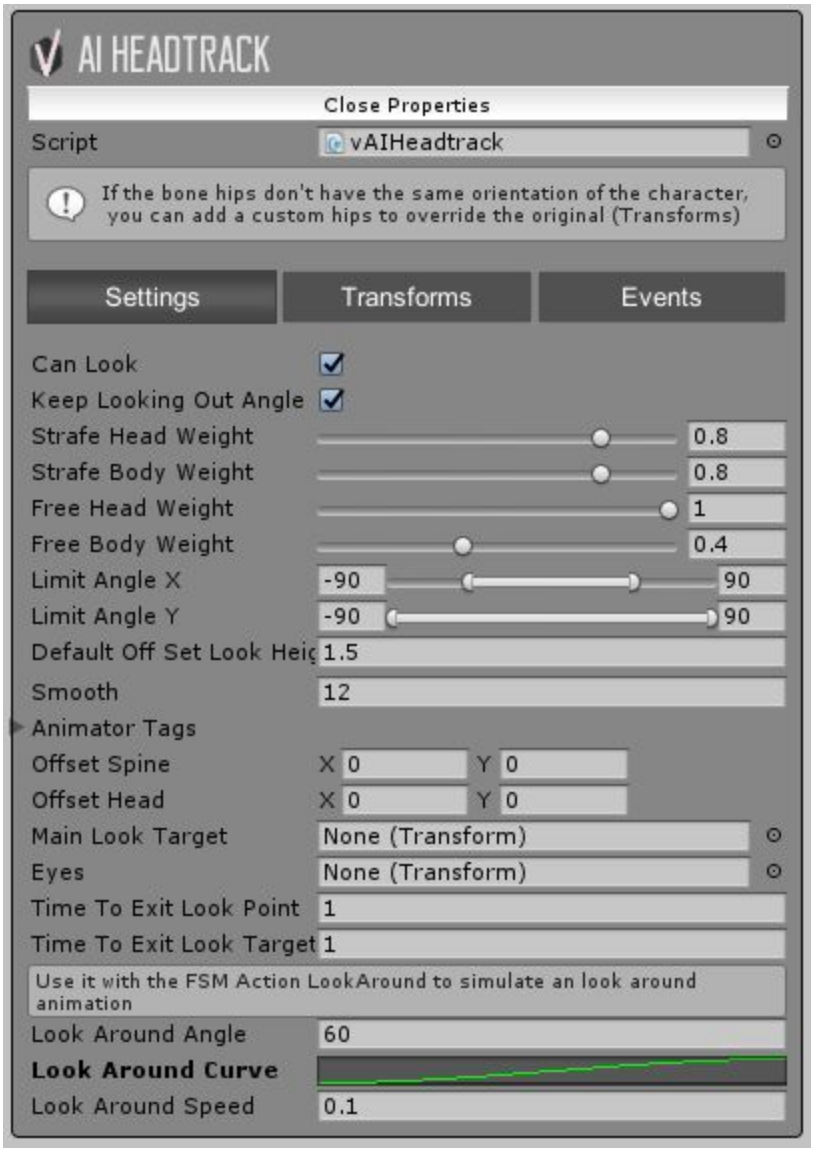
HEADTRACK

Description: Use to make the AI look at something.

You can manually add an AI Headtrack component using the menu:

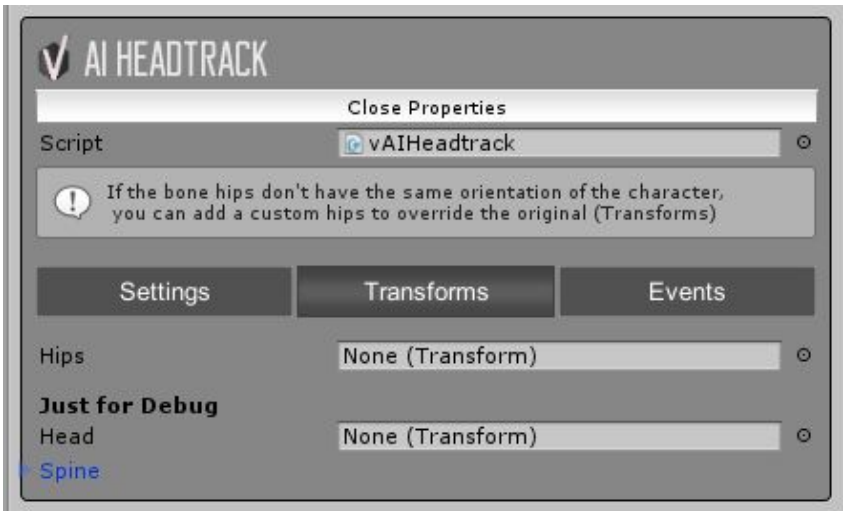


ps The AI Headtrack will be automatically added if you're creating a Shooter AI, it's a required component in order to Aim.*



CanLook: Pretty much for debug purposes, enable or disable the headtrack
Keep Looking Out Angle: Use the headtrack even if the target is out of range (useful when something is pursuing you for example)
Weights: You can set different weights depending on the **Locomotion Type** (Free Movement or Strafe)
LimitAngle X and Y: You can limit the max angle to look up/down and left/right
Default OffSet Look Height: if you want to predetermine an offset Y when using the Look Point
Smooth: the speed to look at something
AnimatorTags: Which animator tags will ignore the Headtrack, for example you have an animation that the character opens a door and look at the knoble, it's best to not use the headtrack while this animation is playing, making the animation play as it was designed without the headtrack interfering.
Offset Spine//Head: more offset options to in case you want to fine tuning your headtrack.
Main Look Target: use any transform to prioritize as a look target.
Eyes: usually an empty transform inside the head bone of your character
TimeToExitLookPoint: how much time it will wait to reset the headtrack when looking at a LookPoint

TimeToExitLookTarget: how much time it will wait to reset the headtrack when looking at a LookTarget
Use LookAround: This option works with the FSM LookAround Action, you can add this action to any state, it's basically
LookAroundAngle:



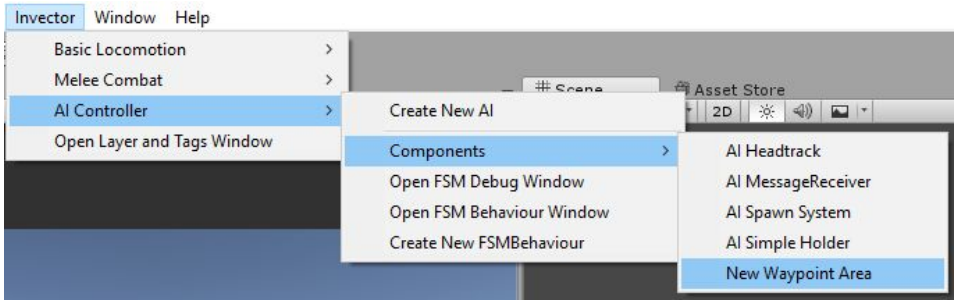
Sometimes your character Hip bone doesn't have the forward pointed forward, in this case you can create an empty transform inside the hip and rotate to have the blue arrow forward, and assign to the Hips field in the Transforms tab, this will assist the headtrack to aim correctly.

Heatrack FSM Actions:

- LookAround: Use with the LookAround settings of the Headtrack to simulate a look around animation.
- Look To Target: will look to the Current Target
- Look To Damage Sender: Will look to the damage sender position

WAYPOINT AREA

You can create a new Waypoint Area by opening the Invector > AI Controller > Components > New Waypoint Area.



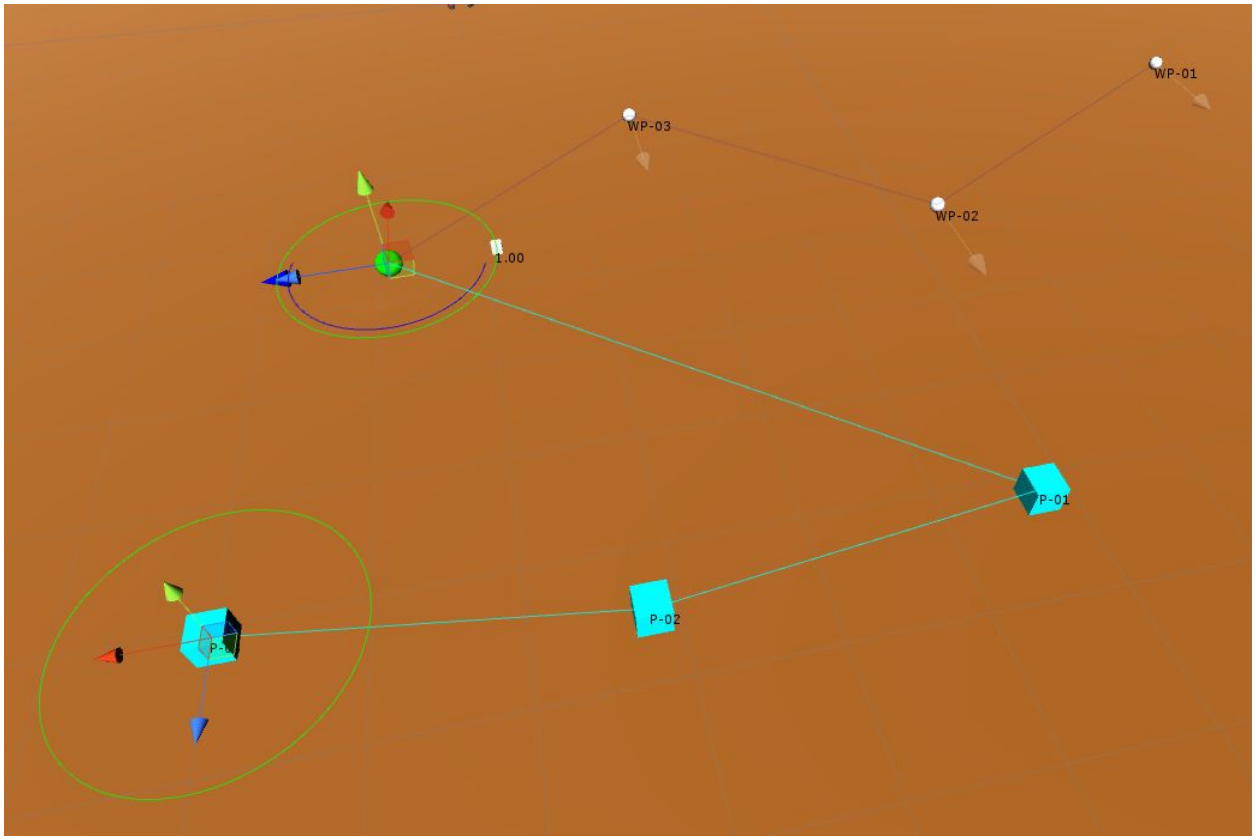
To create a new Waypoint, just hold Shift + Left Click on any surface with a collider, to reposition the same waypoint hold Shift + Right Click.

The same goes to create Patrol Points, but you will hold Ctrl instead of Shift.

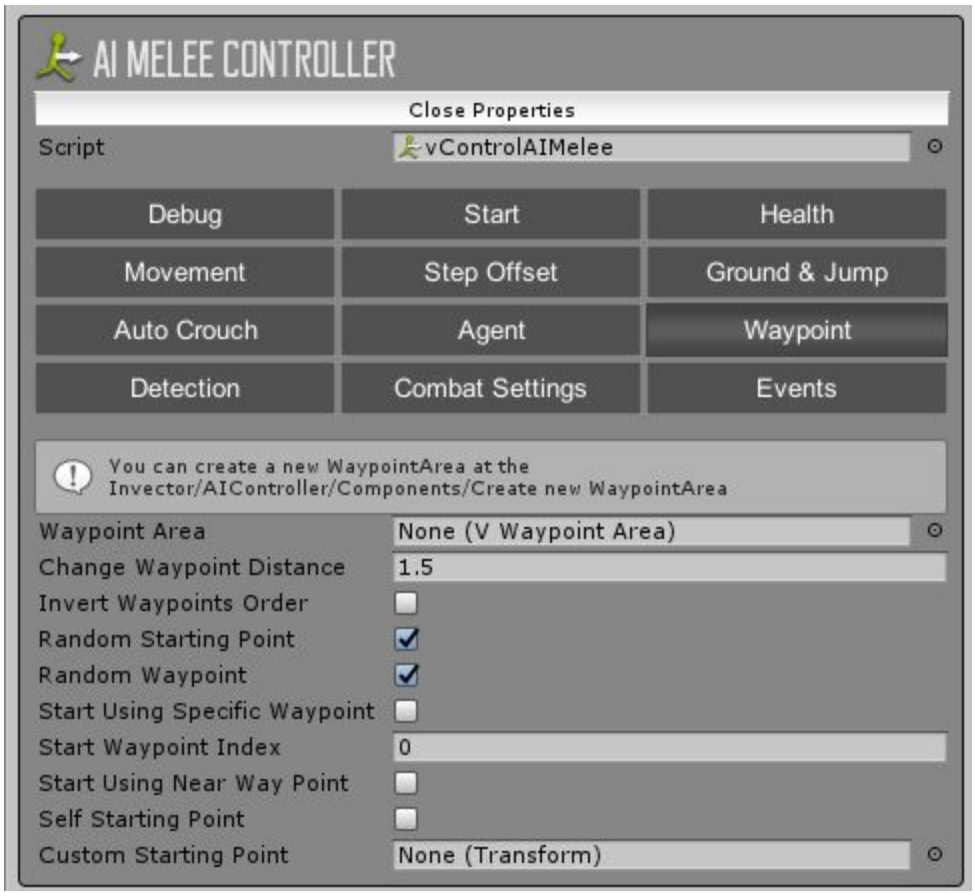
You can assign this Waypoint Area to many AI as you want, and limit the area / limit of AI that will access.

- **Out of NavMesh**: This warning will appear if you have placed the waypoint on a surface that is missing a navmesh, meaning that the AI will not be able to find a path to get there.
- **Patrol Points**: basically, points of interest that one waypoint has, for example if you have a corridor with 3 rooms, you can create 1 waypoint in the middle of the corridor and 3 patrol points with Max Visitors of 1, this means that if an AI is already on a room, the other AI will not come to the same room, he will go to the next one.
- Time to Stay: How much time the AI will stand there.
- **isValid**: Is a bool that you can turn on/off to disable a waypoints/patrol point in real time.
- **Area Radius**: Total area of the waypoint.
- **RotateTo**: The Agent will rotate to that forward direction once he arrives at the waypoint.

You can make the AI walks randomly at waypoints by selecting the option Random Waypoints on the AI Inspector. To make random patrol points, select the option Random Patrol Point on the Waypoint Inspector.
Waypoints are represented by Spheres and PatrolPoints are represented by Cubes.

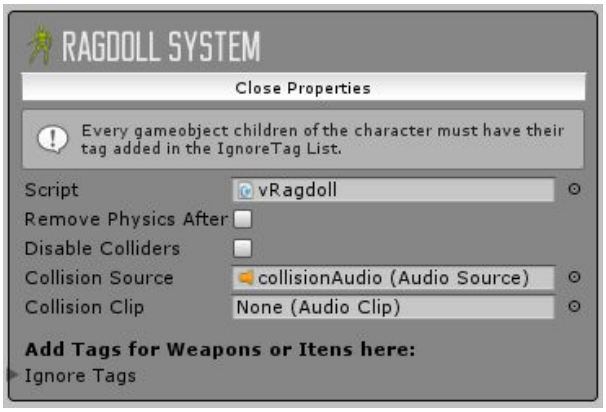


After creating your WaypointArea you can assign to your character's inspector:

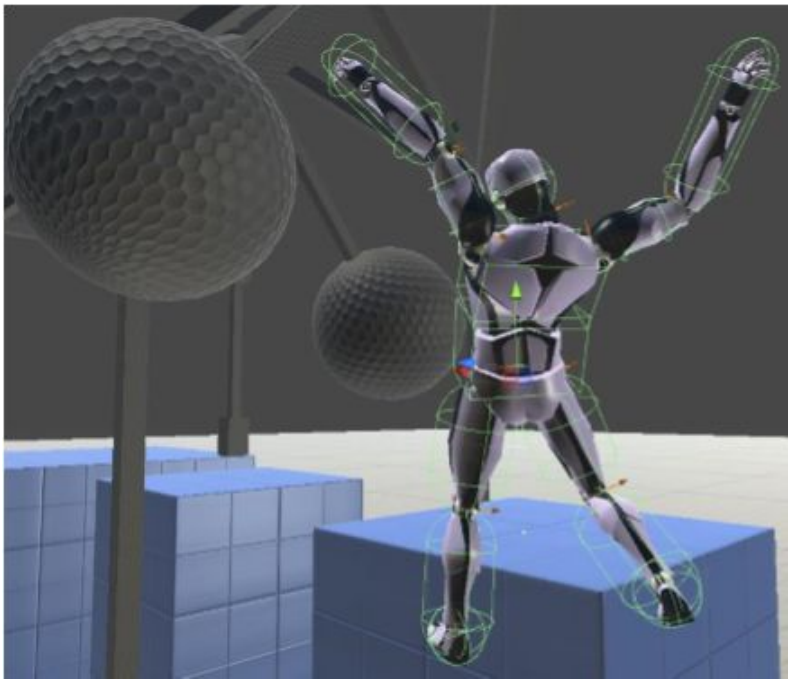


Change WaypointDistance: increase the distance to change the waypoint before reaching the destination
I**nvert WaypointOrder**: Inverts the order of the waypoint index
Random StartingPoint: Randomly select a waypoint index to start
Random Waypoint: Makes a random patrol between all waypoints
Start Using Specific Waypoint: Check to use a specific waypoint to start
Start Waypoint Index: Assign the index of the first waypoint you want
Self Starting Point: In case you don't have a WaypointArea, the AI will make the spawnPoint his default location
Custom Starting Point: Use a transform to indicate his default location

HOW TO CREATE A RAGDOLL

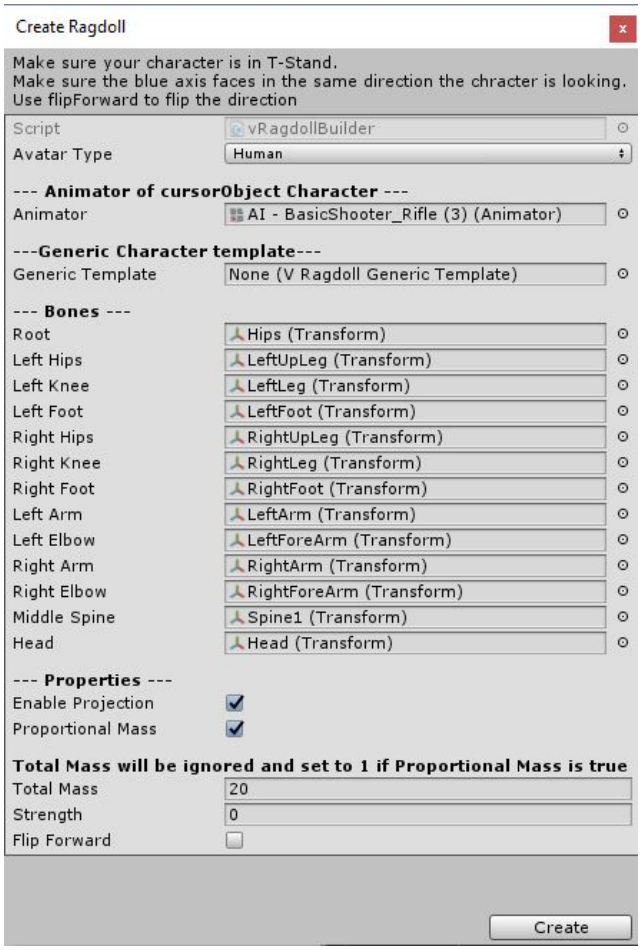


Creating a Ragdoll is just easy as creating your Character, just go to the tab Invector > Basic Locomotion > Components > Ragdoll.



If you have your character using the Rig Type Humanoid, simple select on the Hierarchy and all the fields will autofill, if not, you will need to manually drag the bones to the wizard.

We strongly recommend keeping the Enable Projection and the Proportional Mass enabled, and do not forget to adjust the Scale Factor of your fbx Model. This you provide better behavior of your ragdoll.



If you're using a Generic Rig Type, you can create a Generic Template, allowing you to set the bones only 1 time, this will save you time if you need to add a ragdoll into the same character later on.

Using the Ragdoll Colliders as BodyParts to inflict precise damage.

SHOOTER > If you want to cause damage for each body member using the ragdoll colliders, UNCHECK the “Disable Colliders” and you can add damage multiplier on each member.
* You can use a different Layer on the Ragdoll Colliders like “BodyPart” and a different one for the main capsule collider like "Enemy", this way the Detection will detect the Enemy, but the ShooterManager will actually apply damage to the "BodyPart".

A DamageReceiver is attached to each ragdoll collider, this will allow to Player or AI to apply damage to each bodypart instead of the CapsuleCollider.

- **Damage Multiplier**: multiply the damage value
- **Override a Reaction ID**: Check this option to override the hit reaction animation to trigger a specific animation.

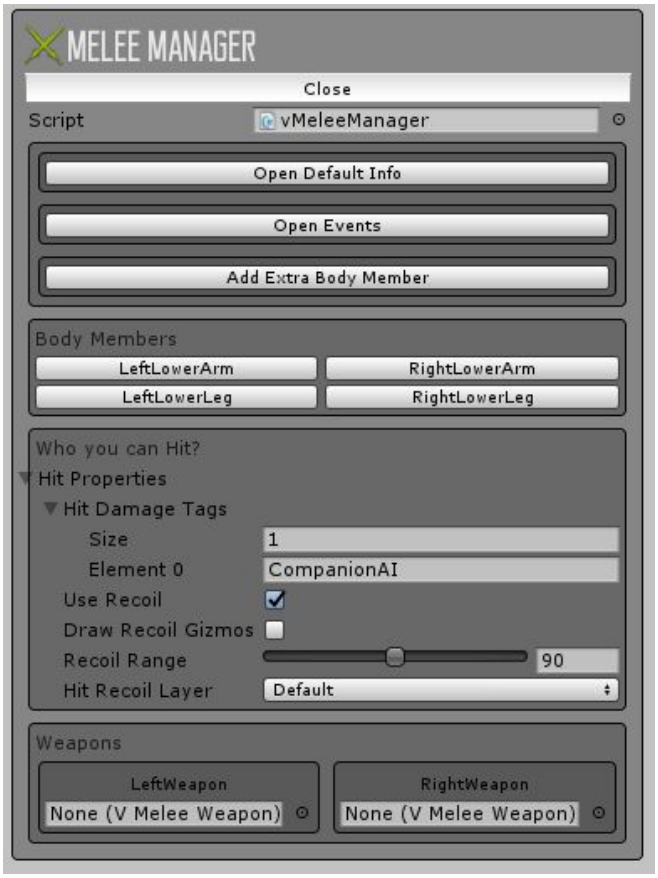
For example, if you want to cause 2x damage and trigger a specific reaction animation when shooting in the Head, simple change the values in this component.

MELEE MANAGER

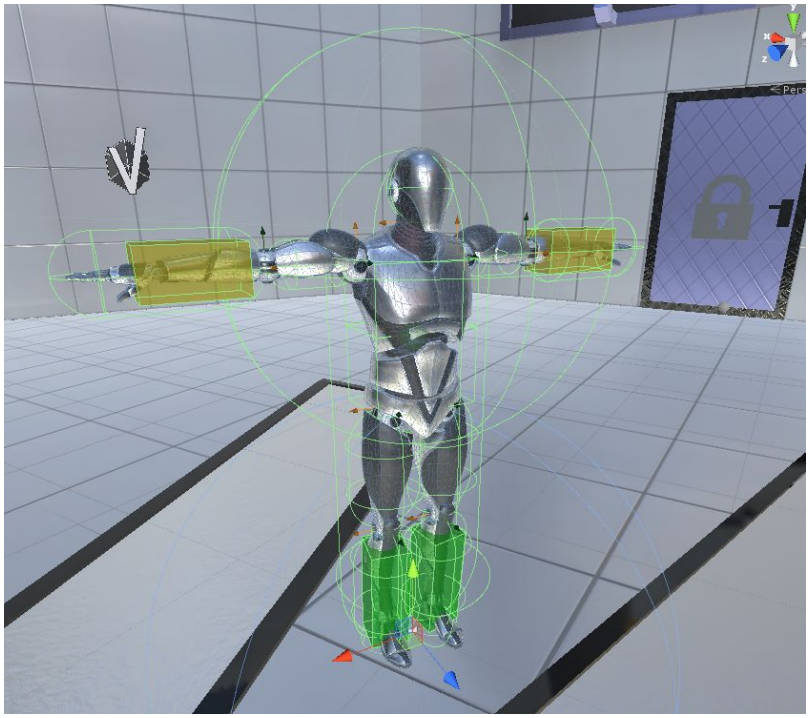
Description: Add this component to handle all the MeleeCombat features like adding a Hitbox, Identify what Objects you can apply Damage, and several other options.

You can add a **Melee Manager** Component by opening the Invector tab > Melee Combat > Components

Open Default Info: here you can setup the default values for Hand to Hand Combat
Open Events: here you can add generic events like trigger something when you make damage
Add Extra Body Member: If you need an extra hitbox for example a Head Hitbox for a zombie, you can add
Who you can Hit > **Important** this is the tag that will receive Damage, so if you are using this component on the Player, assign the Tags of the gameObjects that you want to apply damage (the receiver need to have the method TakeDamage).
Use Recoil > Check if you want the character to trigger a recoil animation when hit a wall
Recoil Range > max angle to allow trigger the recoil animation
Hit Recoil Layer > the layer that will affect the recoil (usually it's the Default layer)



When you assign the **MeleeManager** component into your character, it will automatically create default hitboxes for both hands and legs, you can add an extra hitbox if you need.

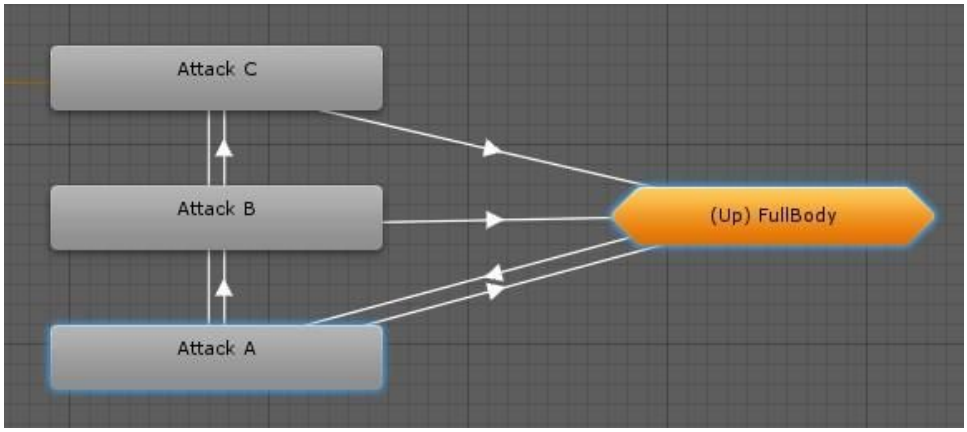


The animations for the hand to hand combat can be set up in the **Unarmed** state machine, trigger by the **ATK_ID 0** and the defense **DEF_ID 1** on the UpperBody Layer**, Default Defense.**

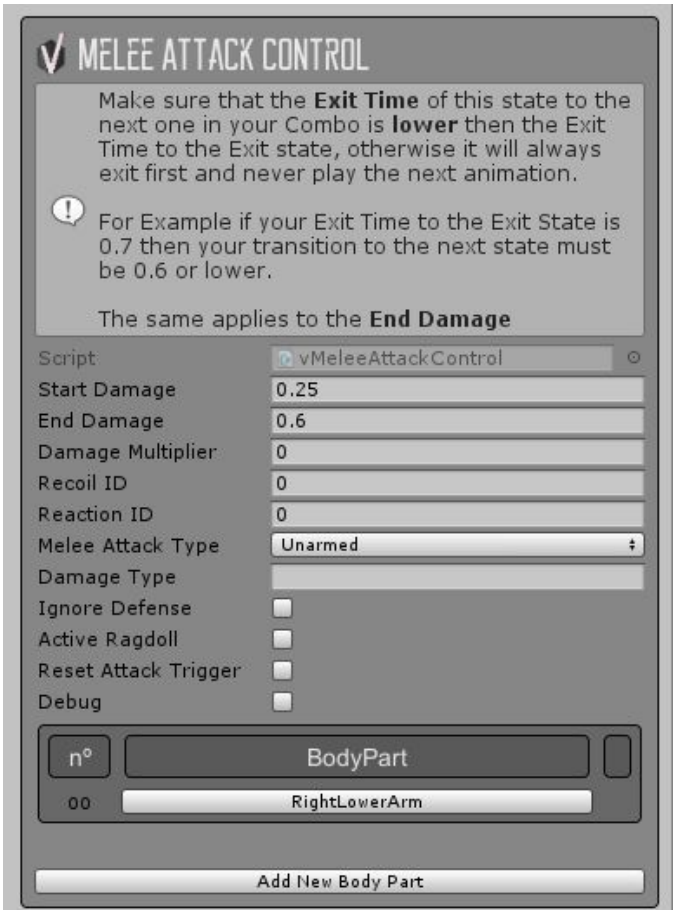
The **Basic Attack** State Machine is just an example, you can have as many State Machines you need, just remember to set up the **ID** to the corresponding weapon.

You can use **UpperBody** to attack as well, this way you can move the character and attack at the same time.

You can set up as many combos as you want, just put the attack animation and apply a transition.



Every Attack State need to have a **vMeleeAttackBehaviour** script attached.



StartDamage > Time of the animation that you will apply damage
End Damage > Time of the animation that will stop trying to apply damage
Recoil ID > Trigger a Recoil animation if you hit a wall or an object
Reaction ID > Trigger a Reaction animation when you take damage
Melee Attack Type > Select Unarmed or Melee Weapon
Reset Trigger > Check this bool for the last attack, to reset the combo
Attack Name > You can write an Attack Name to trigger different HitDamage Particles on the Target, Ex: If your weapon has electric damage, you can match the Attack Name with the HitDamage Particle and instantiate a different particle for this specific weapon.
Ignore Defense: it will apply damage even if the target is blocking
Active Ragdoll: activate the target ragdoll

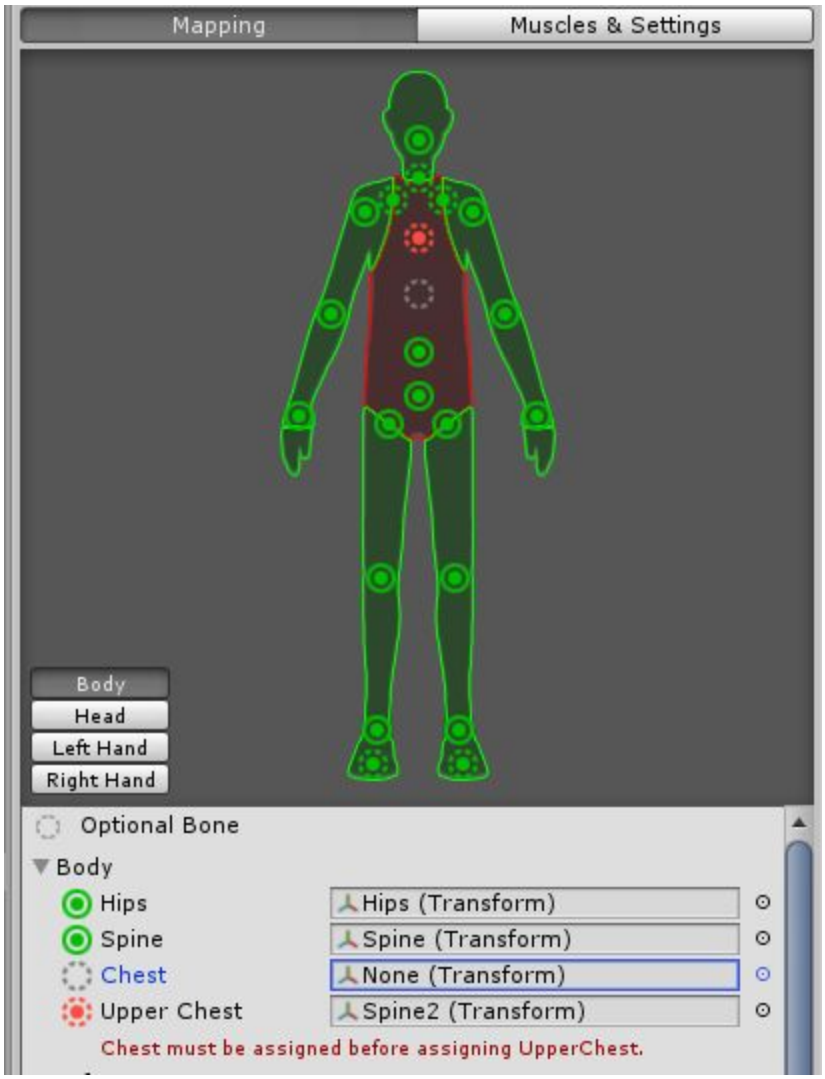
Ps* Don't forget to assign the limb member of your BodyPart to match the animation, this will trigger the correct HitBox, you can add new BodyParts if your attack use more than one member.

BODY SNAPPING ATTACHMENTS

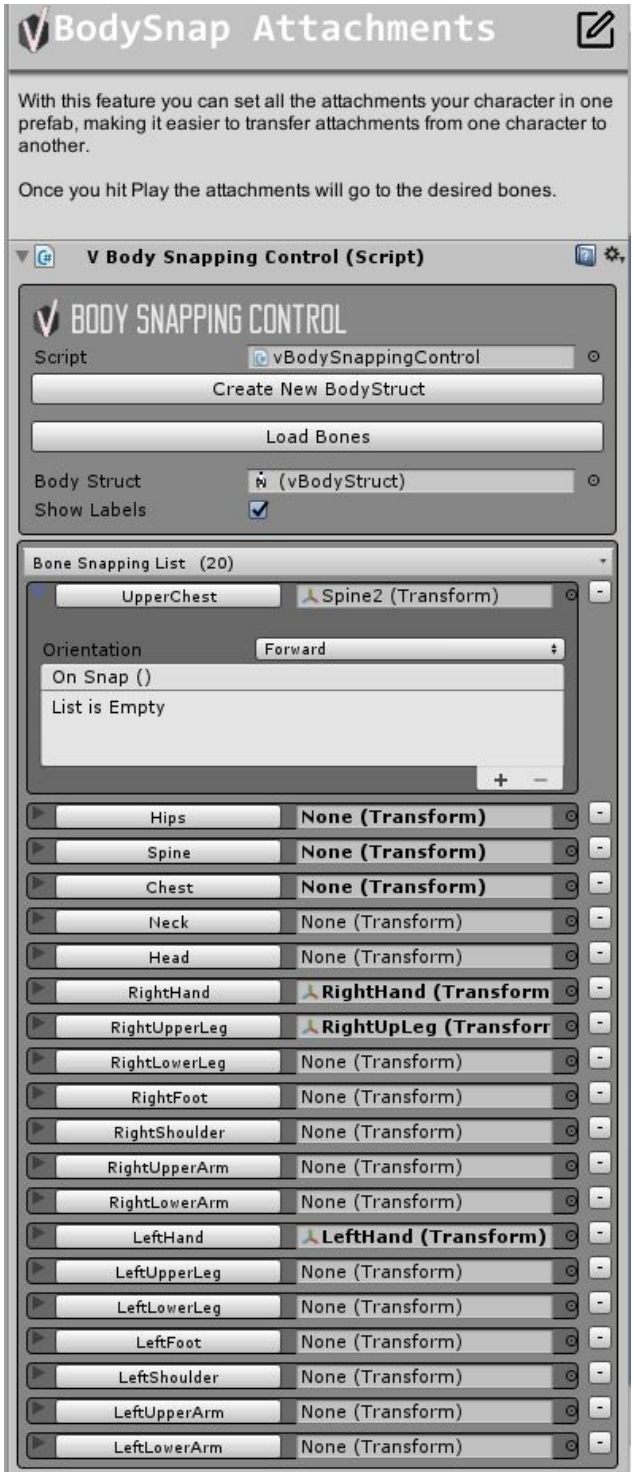
Description: We created this feature to make it easier to transfer attachments from one controller to another. This means that you can create a Prefab of a Character Attachments and quickly add to another character, without the need of adding attachments one by one on each bone.

First, create an Empty GameObject inside your character, add the "**vBodySnappingControl**" and hit the "Create New BodyStruct.

If your Avatar is already set up as Humanoid and all the bones are correctly mapped, it will all be automatically assigned for you, in some cases Unity doesn't recognize a Spine or Chest, so you need to fix by going to your Avatar and assigning the correct Bone, example:



Now going back to our Character BodySnap Control, you can add all your character attachments such as particles that activated on a specific bone, itemManager Handles, anything that you may use and assigned to a specific bone, once you hit Play that GameObject will be attached to the bone you assigned.



ANIMATOR TAG

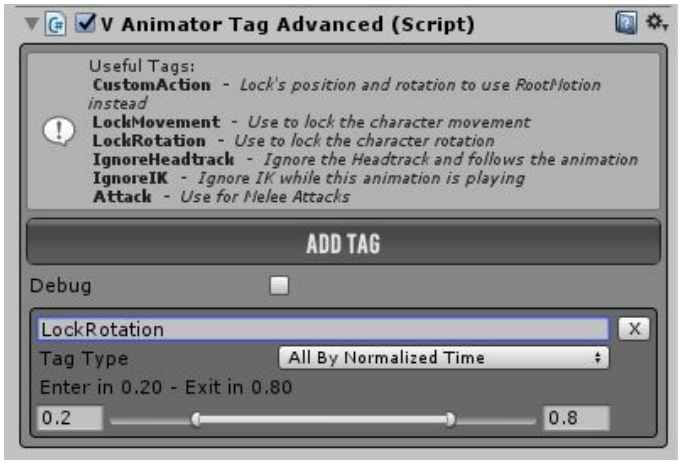
Description: It's an Animator Behavior that you can attach directly on Animation State inside the AnimatorController, it's useful to know what animation is being played and what you can do while this animation is playing.

We have a few useful built in tags such as the image below in the infobox, but you can also create your own tag and verify in code, for example:

First access the **vThirdPersonController** via script so you can call the **method**

```
if(tpController.IsAnimatorTag("MyCustomTag")
{
        //do stuff
}
```

We also have a **AnimatorTagAdvanced** in case you want to check a tag but only during a certain period of the animation, every animation goes from 0 to 1 and you can filter the tag to only run your method during this time.



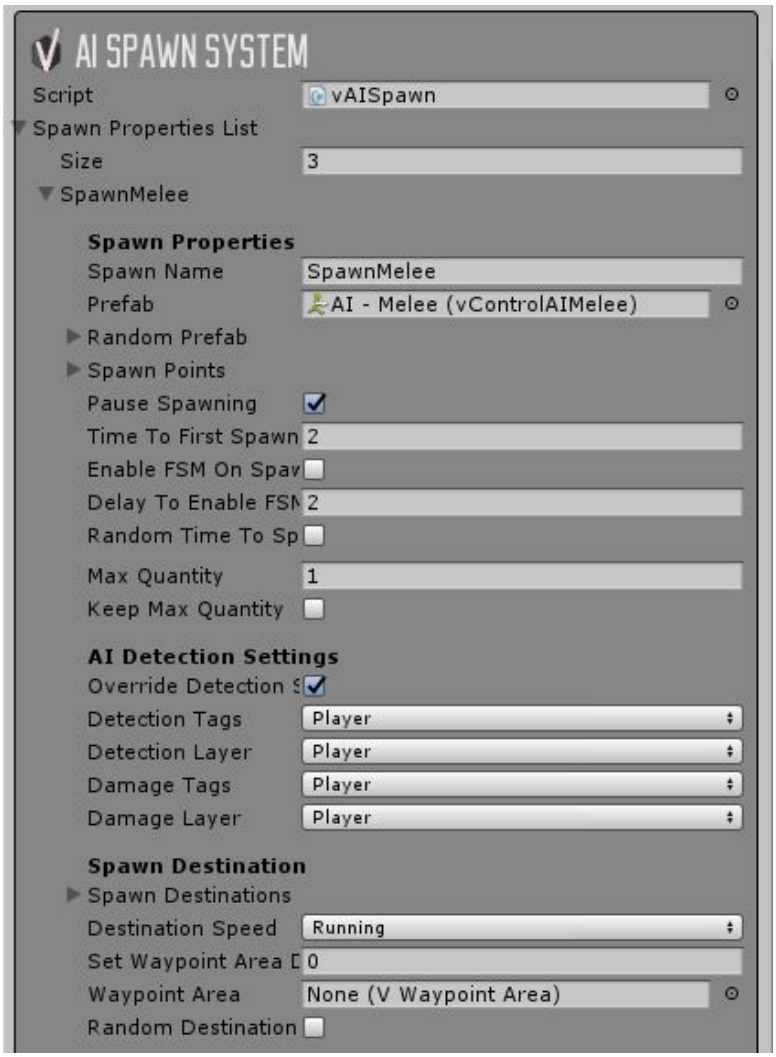
ANIMATOR EVENT MESSAGE/RECEIVER

Send messages from a AnimationState to any GameObject:
https://www.youtube.com/watch?v=uZn53kKsI0I

AI SPAWNER SYSTEM

Description: Instantiate one or different AI Prefabs with several options as setting different detection settings to pursue different targets, sending the AI to different locations or waypoints.

You can create a new Spawner System by going to the menu Invector > FSM AI > Components > AI Spawner System.



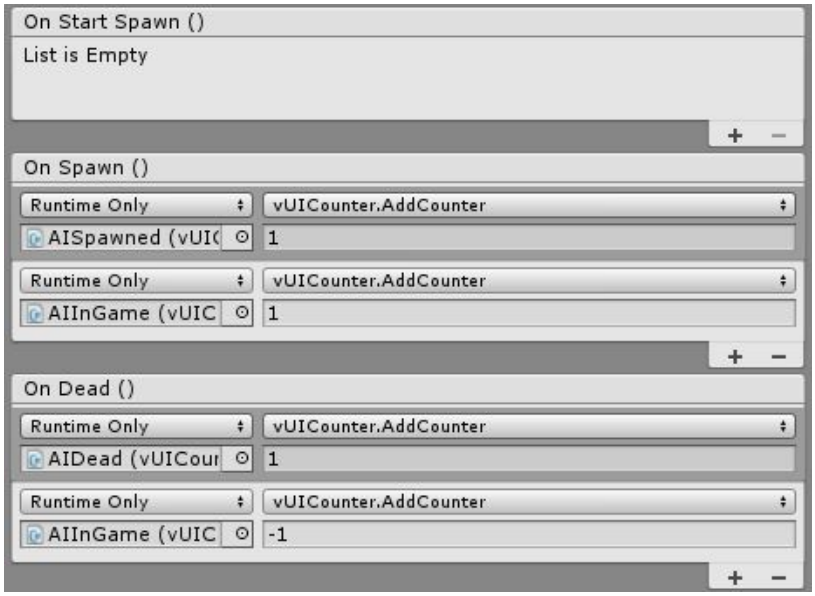
SpawnName: Here you can create a list of different AI's to spawn, first assign a name to identify who is being spawned.
Prefab: If you want to assign a unique prefab, assign here the AI Controller prefab.
Random Prefab: If you want multiple AI prefabs assign here all your prefabs, it will spawn randomly.
SpawnPoints: You can assign different SpawnPoints to assign your prefabs, you can create an empty transform and add the SpawnPoint component, you also need a Collider set to IsTrigger.
Pause Spawning: It will pause spawning the prefabs (useful to wave control timing)
Time To First Spawn: delay for the first time spawning
Enable FSM OnSpawn: Enable the FSM Behavior Controller
Delay To Enable FSM: Only enables the FSM after a period of time
Random Time To Spawn: Sets a min/max time to keep spawning prefabs
Max Quantity: How many prefabs are allow to be spawned
Keep Max Quantity: If an AI dies, it will keep spawning until reaches the Max Quantity.

- AI Detection Settings
Override Detection Settings: Check this to override the Detection Settings of your AI Controller
Detection/Damage Tags - Set the Detection/Damage layers
Detection/Damage Layers - Set the Detection/Damage tags

Spawn Destination: Set an empty transform to make your AI go there as soon as it is spawned (needs to disable the FSM in order to not run any states like Patrol, don't forget to set a Delay to enable the FSM)
Destination Speed: Set the speed the AI will go to the destination
Set WaypointArea Delay: Set a delay to reach for the Waypoint Area
WaypointArea: Assign here a WaypointArea to make the AI Patrol after spawned
Random Destination: If you assign multiple Spawn Destinations, it will choose randomly.

AI Spawned List: A list of the spawned AI's in the scene

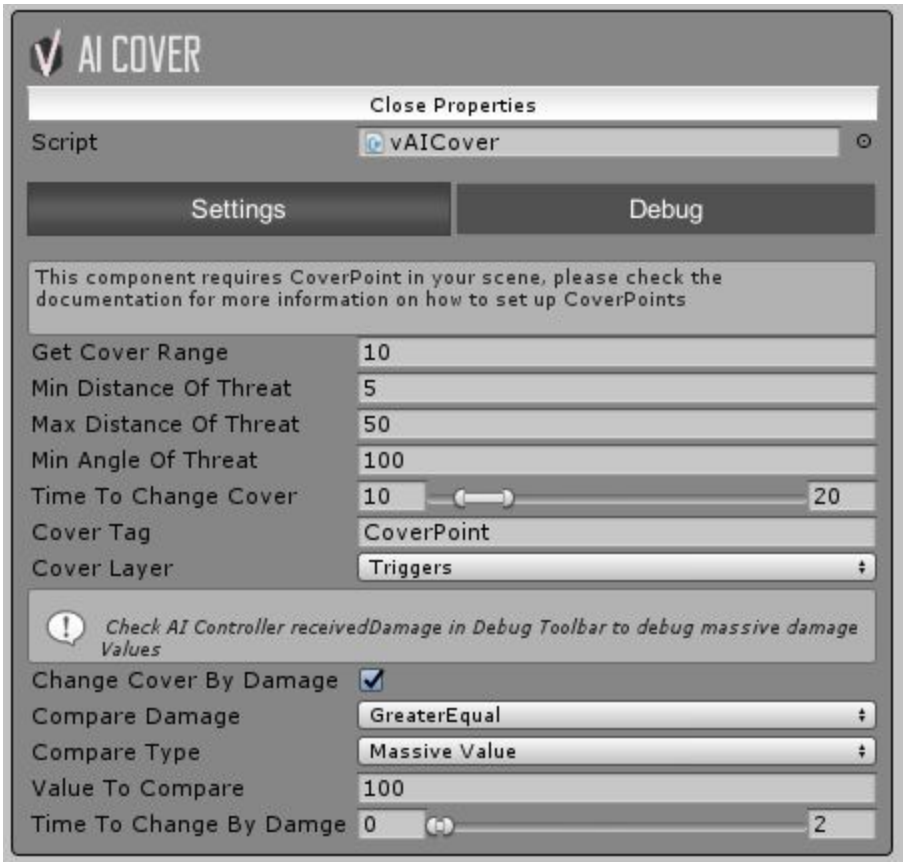
You can also trigger Events for **OnStartSpawn**, **OnSpawn** and **OnDead**



AI COVER SYSTEM

Description: The AI Cover Component works with the **AI Shooter Controller**, **AI Cover Area** for the scene and the FSM **Get Cover** Action to search and find a CoverArea, go to the position using GetCover Action and duck to avoid taking shots.

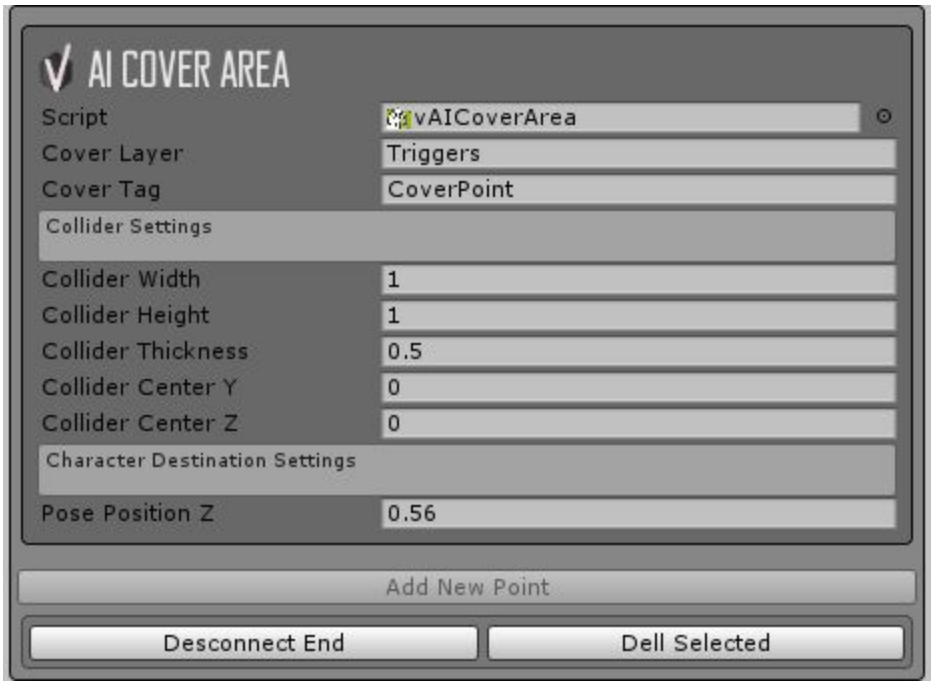
You can add a Cover component by going to Invector > FSM AI > Component > AI Cover.



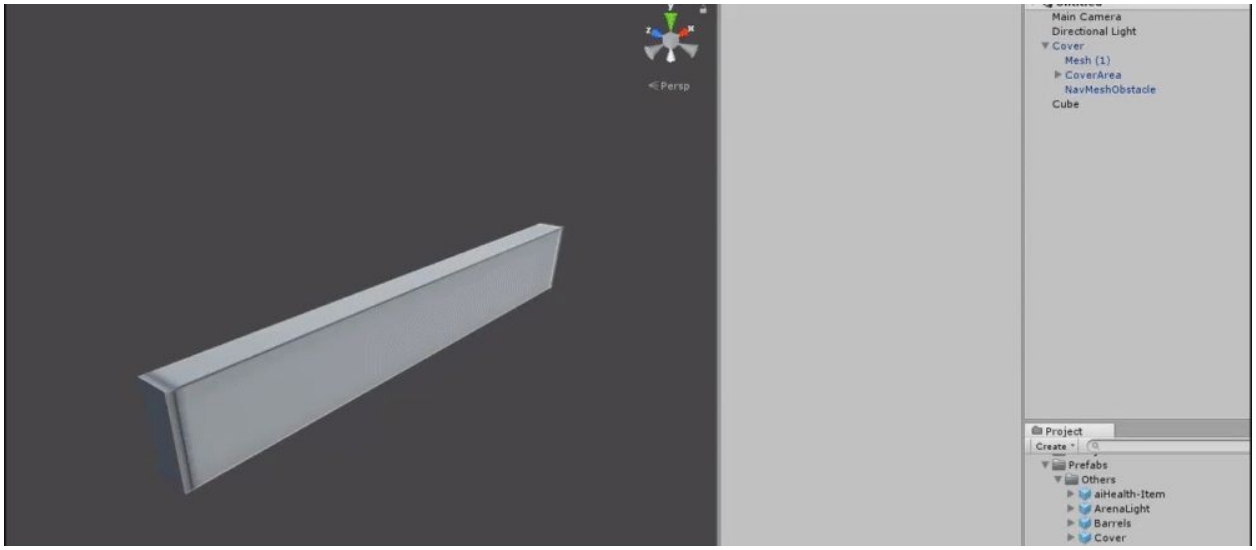
GetCover Range: The distance to find and get a cover
Min/Max Distance of the Threat: the minimum and max distance to be from the thread
Min Angle of Threat: Once in cover, you can set the min angle to avoid the threat, if reaches the angle the AI will search for a new cover
Time to Change Cover: min/max time to change between covers
Cover Tag/Layer: tag and layer from your CoverArea object in the scene

ChangeCoverByDamage: Check this to change the cover if you receive a 'x' amount of damage or hits.
Compare Damage: Compare if the damage was less or greater/equal to the **Value To Compare**
Time To Change By Damage: Use this to create a random timing to change the cover after 'x' so that not every AI change the cover the same time every time it takes 'x' damage/hits.

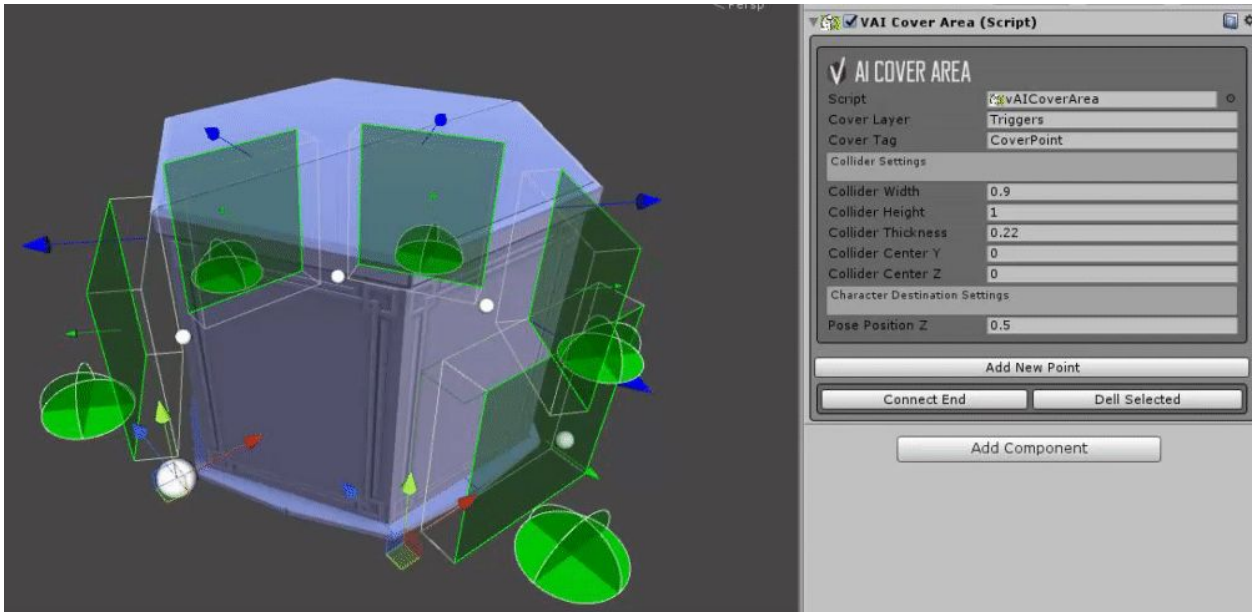
You can use the prefab “Cover” as a starting point to create your own cover.



Use this tool to create multiple cover points, you can expand using the gizmos in the scene to allow several points to be added and also set the direction of the cover.



Create a EmptyGameObject inside your 3D Model and add the **Cover Area** component, now reposition to where you need it and expand to fill with how many cover points you need, you can set the direction, width, height, thickness,etc...



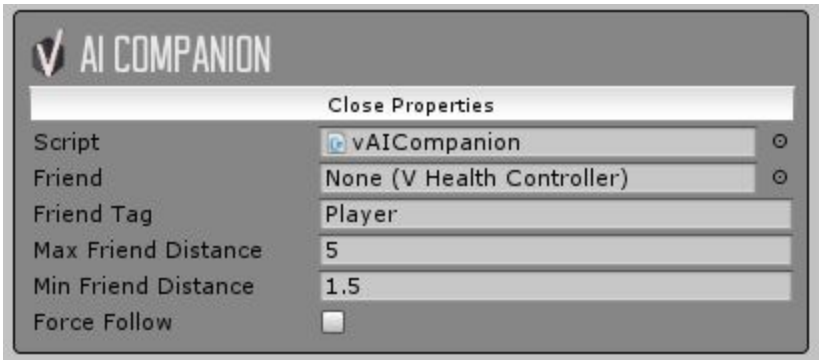
You can also create Create new Points, Delete or Connect with the End, to model your CoverArea on pretty much any object you need.

Now you just need to add the Action **Get Cover** to your FSM State running on an Update Execution Type in order to keep finding CoverAreas for your AI to go.

AI COMPANION

Description: Simple example of an AI Companion that follow a friend.

You can add the component by going to *Invector > FSM AI > Components > AI Companion.*



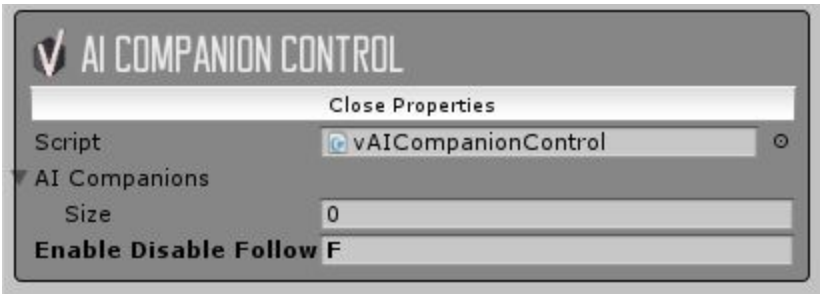
Friend: The Friend must be a vHealthController
Friend Tag: If you don't assign the field above, you can find one by the Tag
Max/Min Friend Distance: Max/min distance of your friend
Force Follow: It will force the companion to stop what's doing and to the friend position

FSM Companion Actions:

Go to Friend: Makes the AI go to the Friend position

AI Companion Control (Attach to a Friend/Player)

This is a simple script that automatically identify any Companions in the scene so you can ask him to Follow or Stay by pressing an Input.

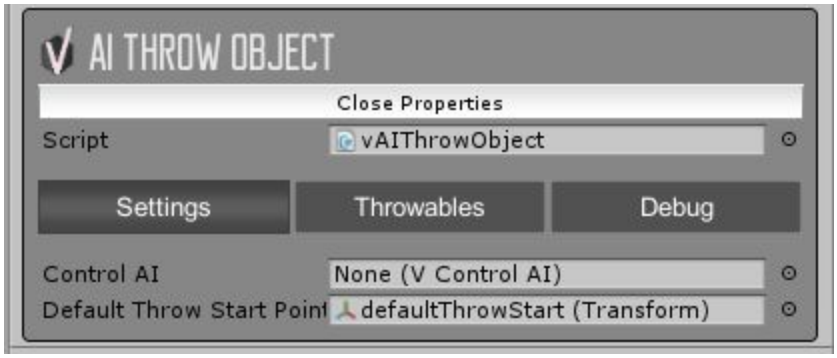


AI THROW OBJECT

Description: This will allow the AI to throw objects at a target.

If you have trouble trying to set up this feature, try looking into the “AI_Cover-Throw-Companion” demo scene or the ‘AI - ShooterCover’ prefab to see how it works.

You can add this component by going to *Invector > FSM AI > Component > AI ThrowObject*

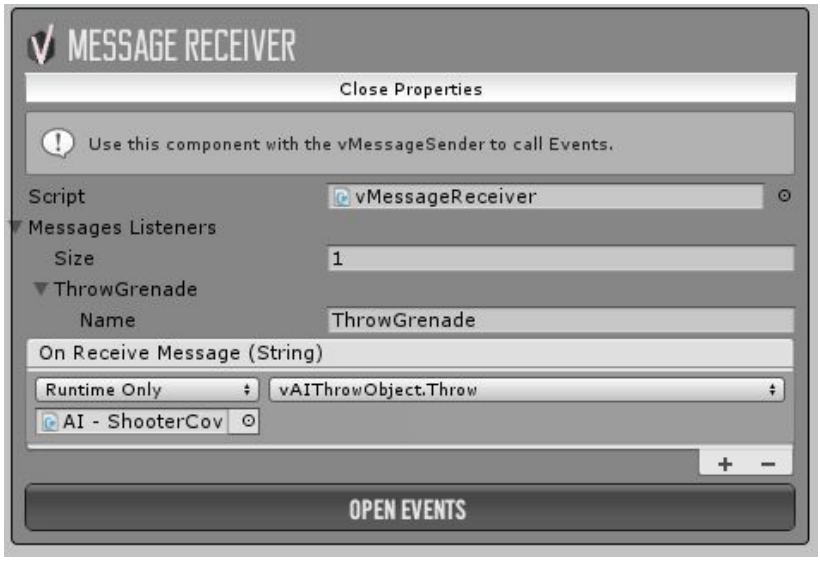


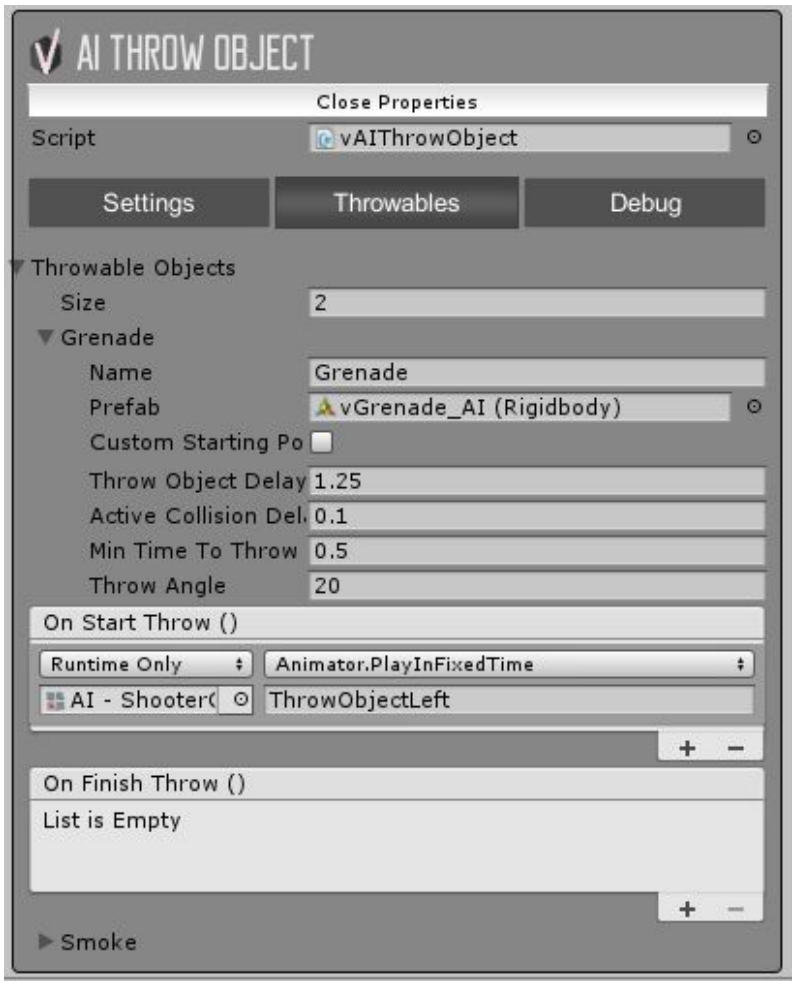
Control AI: It will automatically assign when start
Default Throw Start Point: Create an empty gameObject inside your character’s hand bone.

You need to manually trigger the **Throw** method in order to make the AI Controller to actually start the throw routine, in the AI - ShooterCover we use a **MessageReceiver** to communicate the FSM Behavior when the character will throw the object.

Notice that we send the message **ThrowGrenade** with the parameter string **‘Grenade’** to identify the type of the throwable we want.

The **MessageReceiver** will call the **Throw** Method from the **AI Throw Object** and filter the parameter **'Grenade'** to identify the Name of the Throwable Objects in the list





You can set different throwables in the **Throwable Objects List.**
Name: Name of the object you want to throw
Prefab: Prefab of the object
Custom Starting Point: If you don't want to use the default Start point, you can set a different transform
Throw Object Delay: Delay to instantiate the object
Active Collision Delay: Delay to active on collision
Min Time to Throw After: minimum time to wait until it can throw again
Throw Angle: Angle to throw the object

Events:
- OnStartThrow - can be used to trigger sound, particles, animations, etc...
- OnFinishThrow - can be used to trigger sound, particles, animations, etc...

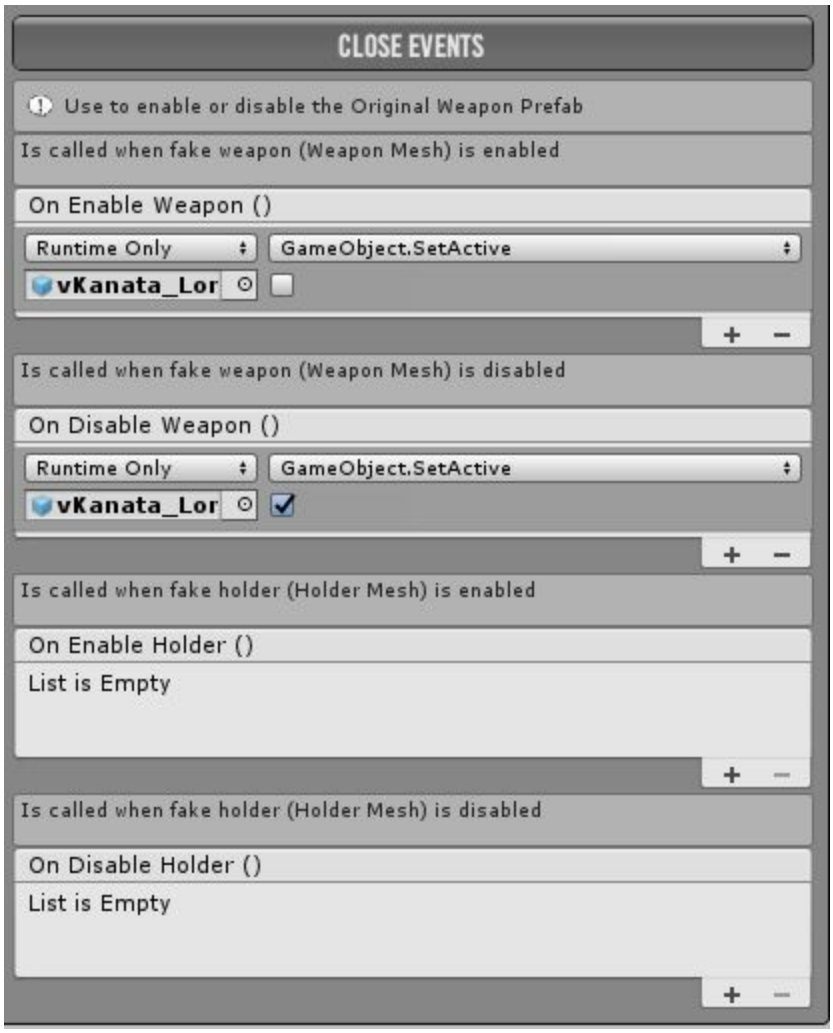
AI SIMPLE HOLDER

Description: Add a holder to your AI, for example while patrolling the AI will unequip his weapons and equip when in combat.

You can add this component by going to *Invector > FSM AI > Components > Simple Holder.*

Holder Object: If your 3D Model has a holder, place into your character and assign here
Weapon Object: This is a fake weapon, just the 3D mesh without any collider or components, also place inside your character bone
Animator: Assign the Animator of your character

Equip/Unequip Anim: Check your animator and assign the name of the animation you want to equip/unequip
Equip/Unequip Delay Time: delay to actually set active/disable the weapon, useful to match with the animation



Events:

OnEnableWeapon: Use to disable your Actual weapon prefab that should be already in the character
OnDisableWeapon: Use to enable your Actual weapon prefab
OnEnableHolder: use to trigger a sound or effect when the holder (fake weapon) is active
OnDisableHolder: use to trigger a sound or effect when the holder (fake weapon) is disabled

Now we need to manually call the **EquipWeapon/UnequipWeapon methods**, you can do that by using the FSM **SendMessage** Action, which sends a message from the FSM State to the Controller.

For example: When running the Patrol State you can send a message to UnequipWeapon and as soon as you find a target change state to Combat, send the message to EquipWeapon.