AI DESIGNER PRO & LITE
v0.95 Beta

USER MANUAL

The complete solution for crafting complex characters without any code.

COMMON QUESTIONS & PROBLEMS: Please see separate document in the "/Documentation" folder: *Answers to Common Questions & Problems.PDF*

HOW TO USE THE EDITOR: See Last Chapter, "Editor Tips and Tricks"

GETTING AROUND THIS MANUAL: All Chapter links in the Table of Contents can be clicked

Note: This is the "Pro" version, so any chapters that say "Requires Pro" is referring to this. We included that statement because there might be potentially different versions released for the future. This Pro version will always be the heaviest, most feature-rich.

We initially built this AI system for our own use and needs.
If you'd like to see something either in this user manual or a new feature request, suggest it to us at **aibotsystem@gmail.com**. If there's enough demand, we'll add it!

Table of Contents

1 | INSTALLATION

First-time users:

1) Create a new blank project.

2) Import our asset.

3) Play demo scenes folder (AI Designer Pro / Demo Scenes)

4) Make sure there are no conflict script / asset names with your existing projects to prevent overwriting.
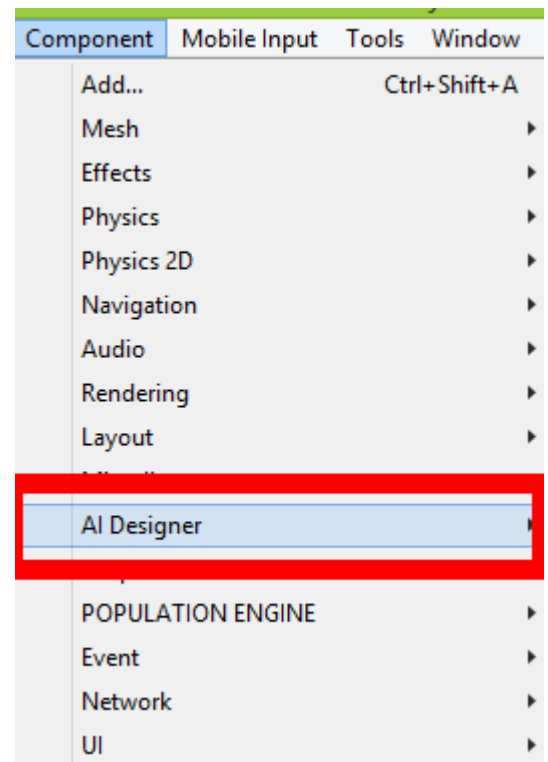

Future installations:

1) Import into any project.

How to know if installed correctly- You should see 2 new menus in your Unity Editor:
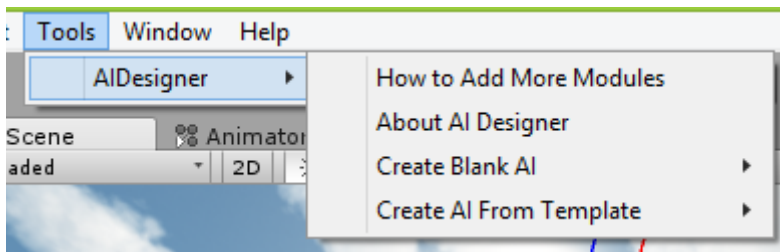
1) Menu "Tools / AI Designer"

2) Menu "Component / AI Designer"

After import, First-time users should read this entire section before continuing.

THIS LOOKS LIKE A BIG SYSTEM! WHAT'S THE LEARNING CURVE? It is a big system. Thousands of lines of code make up this AI engine so you don't need to code it yourself! When you first open up the demo bots, the massive number of configuration settings may seem daunting. BUT – don't worry! For simple AI, you can get set up in about 10 minutes – many settings can be ignored. For the more advanced stuff, it may take a few hours or days to explore everything. This is actually a good thing – the amount of configuration settings mean you can do SO MUCH without any coding.

HOW DOES IT WORK? Normally, when you add any raw model (animal, car, jet) to your scene, it doesn't do anything by itself. What AI Designer does is it adds many components to that model, instantly giving it the abilities to perceive its environment and act. These new components reside in the Unity Inspector. These components, such as the Movement Controller, are built from thousands of lines of code and work together with other components to make the AI think and move. You, the game developer, can use these components to visually build your AI, without writing any code. If you know how to use the Unity Inspector and understand the basics of Unity, then you can to learn to use AI Designer rather quickly.

WHAT ARE "CONTROLLERS"? In AI Designer, each component of a character is called a "Controller." Controllers give your characters specific AI functions. So if your AI needs the "Flee" ability, attach the Flee Controller. If another AI doesn't need to Flee, remove the Flee Controller. If you look at the demo bot prefabs, each bot has a "Movement Controller", a "Target Controller", a "Wander Controller" and so on.   Controllers mostly do not act by themselves. They need the BRAIN component to activate them using conditional logic – this you can edit visually, without any code.

WHAT IS THE BRAIN COMPONENT? This is a powerful central system for creating logic your character needs to come alive. The brain has "Actions", consisting of the conditions for triggering actions, and the actions themselves. For example, if you want the AI to chase enemies, you would add a new Action. Name it something like "Chase enemy". Under "Targeting Conditions" click "Target in Sight".   Then, in the ACTIONS section, find "Engagement Actions" and click "Do Chase Target". That's it – it's simpler than it sounds once you get used to it. Note: Take caution in the name you pick for your brain Action. This is a case-sensitive name that will be referred to by other Controllers or third party scripts.

WHAT ARE FLAGS? The Flag system is an event system that signals other Controllers when to start/stop certain tasks and can integrate with your own game systems & scripts.. Flags are set up on the Brain component under "Flags". Flags can be named anything you'd like, and are case-sensitive. Each Flag has 2 states: Active or Inactive. **In most use-cases, you do not need to touch the Flag system.** But once you need more advanced functions, Flags come in handy. Example #1: You are making a shooter-deathmatch game. You have a script that starts each round with a countdown: 3, 2 1… it only activates enemy AI when it counts down to 1. On your AI Designer character, you'd add a Flag name called "START_GAME". Then, create a new AI Action called "Chase Enemy" and add 2 conditions: "Target In Sight" and "Check These Flags". You'd add the flag name "START_GAME" under the "Check These Flags" section and check the box "Check Flags". Finally, in your own countdown script, use our simple API to activate the flag "START_GAME" on the character. Example #2: Your character gets hungry after some time (created using the Emotion Controller) and needs to find Food objects (using the Goal Controller). But what if there are enemies nearby? Should he grab the food or chase the enemy? Say you want it to grab the food first before engaging enemies. So you create a new Flag called "HUNGRY". Then, under your "Chase enemies" Action, insert "HUNGRY" into the Flag conditions for "Check These Flag INACTIVE". Now, the AI will ONLY chase enemies if the "HUNGRY" flag is inactive, and stop chasing enemies when active.

HOW DO I ADD EFFECTS? AI Designer comes with our powerful "Effects Builder" system that lets you build re-usable effects without any code. This lets you trigger specific animations such as Melee attacks, jumping, crouching, dying, and more.   The "Effects

Builder" component is attached to every AI, if not, add it from the components menu. For example: You have a horse. You want the horse to spawn dust particle effects at its feet, each time it moves. You can do this with the Effects Builder. Or, say you want your enemy AI to play the shooting animation each time it shoots. You'd use Effects Builder to create a new effect, name it "Shoot animation" and insert the animation name. Then, on the Ranged Attack Controller, type "Shoot animation" into the "On Fire One Shot" box. Now, each time the AI shoots, it will refer to the Effect name you inserted (case sensitive).

3 | QUICK START: CREATE NEW BLANK A.I.

Adding AI to a raw model takes a few clicks:

1) Drag your RAW model into the scene. Can be anything – animal, car, aircraft, humanoid, etc.

2) Select your model. Go to:
TOOLS → AI Designer → Create Blank AI → Create New Blank AI Setup

3) A new GameObject will be created in your scene with basic AI components. Your original raw model will be placed as a child object under it. Select the new GameObject to modify it. You MUST manually select the new GameObject because Unity does not change your mouse cursor after clicking the menu item, which may lead to confusion about which object is currently selected.

4. Your raw model is now AI-enabled! Now, you MUST remember to do these next few steps right afterwards (to save you from headaches later when strange errors arise):

> a) Select _Scan Position, drag it to your model's Eye level, a bit in front of it. You can also optionally child this to your model's head bone.

> b) Drag _Collision Sensor Position to chest level, a bit in front.

> c) Drag _Base Position to feet level. If later the AI stops moving even when it should, try setting this a bit higher.

> d) Click the New Agent and find the IDENTITY component. Set the Faction.

> e) Find the AWARENESS component. Set Enemy Factions and Friendly Factions (can leave blank – no need to add agent's own faction)

> f) Find the Animator component. Fill in the animation controller. For out-of-box

humanoid animations, use our included controller, "Advanced Combat Controller"

5) That's it. You can modify any other setting on the agent before playing the scene. If it has the WANDER Controller attached, it should automatically wander around.

4 | QUICK START: CREATE NEW A.I. FROM TEMPLATE

AI Designer comes with several templates for different general types of AI. These templates come with Brain logic filled in for you and other configurations (such as Movement).

The steps are almost exactly the same as the previous chapter, creating a blank AI. Except you'd do the following:

1) Drag your raw model into scene.

2) Select your model, then go to the menu:

Tools→ Create AI From Template → [Select a template]
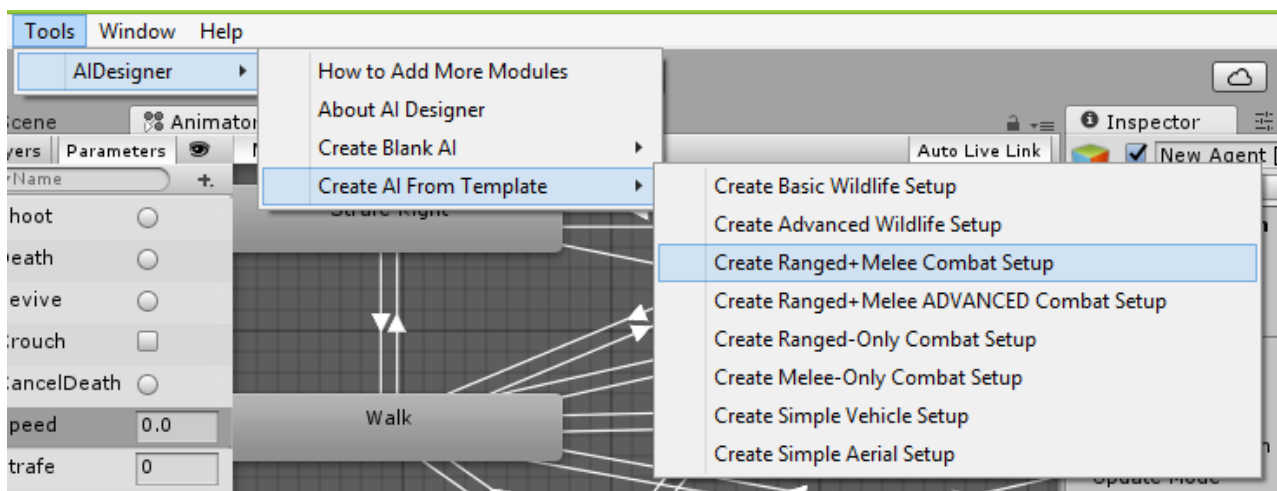
5 | QUICK START: CREATE NEW A.I. FROM EXISTING A.I. (DUPLICATE SETTINGS)

Let's say you created a Swordsman AI character and you spent a lot of time configuring its logic and movement parameters, targeting, melee setup, etc. Now you want to create a new AI, Spearman. Why go through all of that again? You can easily duplicate the Swordsman character, swap the model and Animation avatar, and simply rename it to "Spearman"!

1) You already have an AI character setup. Drag it into your /Assets/ folder to save it as a Unity prefab (in case you accidentally overwrite it).

2) In your scene, duplicate the character.

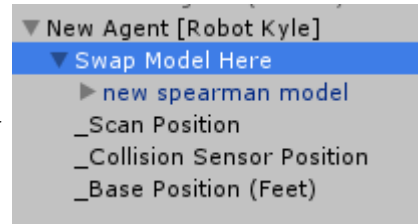3) Under "Swap Model Here", delete your old model. You may want to pull out any attached weapon models before deleting it.
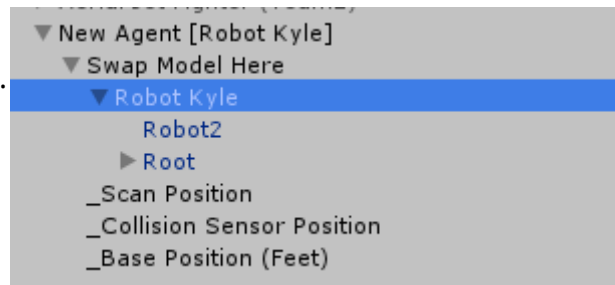
4) Insert your new RAW model under here.

5) Click your new raw model and DELETE any attached Animators, Colliders, etc. Make sure its blank.

6) Click the main AI parent gameobject "New Agent" and find the Animator. Insert your new model's Avatar.
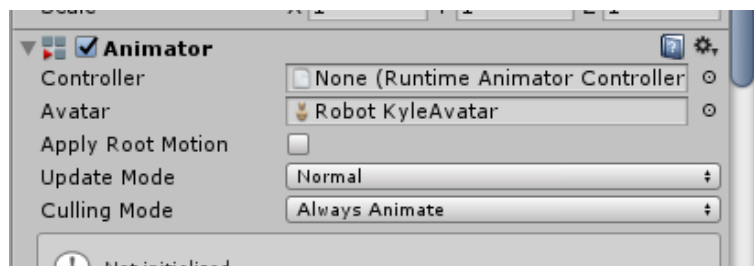
7) If your new model is a different size, find the Capsule Collider (or other collider) and change the settings to make the collider fit the model.

8) Finally, Rename and Drag the new AI object into your Assets folder to create your new AI

character.

You modify factions in 2 places:

1) <u>Identity Component:</u> Every character, AI, or object in your scene that you want targeted must have an "Identity" component attached. This includes your human players. This component has a slot named "Faction". Just type in the faction name – case sensitive. **The object also must have a collider that isn't a trigger for the AI to "see" it.**

To add:

- Select your GameObject (again, this doesn't have to be AI character. Could be anything like a simple cube – see the Attacking Simple Cube demo scene)

- Go to Components Menu → AI Designer → Identity → Add Identity

- In the "Faction" slot, enter the faction name (e.g. Team1 or Team2 or Kingdom3)

- Add a collider (not a trigger) to this object.

What happens now? Let say you entered "Team1" for the Faction. Now, whenever an AI in the scene targets "Team1" as its enemy faction, this object will be attacked.

2) <u>Awareness Component:</u> Every AI must have the Awareness component. Here, you set up Enemy Factions and Friendly Factions for this AI. Unlike the Identity component, this should **only be added to AI-enabled objects.**

Careful: Make sure not to add the AI's own faction into its Enemy Factions list. This is a common cause of strange errors when the AI starts targeting itself.

7 | ENABLE MELEE & RANGED ATTACKS

HOW TO ENABLE MELEE & RANGED ATTACKS? Enable Melee/Ranged attacks in the Target Controller.

Simply check/uncheck the boxes: "Enable Ranged Combat" and "Enable Melee Combat". Your character must also have the attached Melee Attack Controller for Melee, and a Ranged Attack Controller for ranged attacks.

HOW TO SET ATTACK RANGES? **On the Target Controller,** find 2 settings: "Shoot Range" and "Melee Range". The red editor color draws the boundaries of the ranges in your Editor for your convenience.   **Note: Attack ranges are set on the TARGET CONTROLLER, and not on the Melee / Ranged Attack controllers themselves. Why? Because the TARGET Controller makes decisions on WHO to target next, and therefore, it needs to know the range to check for melee or ranged attacks.**

Programmer Note: These 2 ranges can be accessed your scripts (float shootRange, float meleeRange), as part of the GetComponent<AIDesigner.TargetController>() component.

15

8 | THE EFFECTS BUILDER

The Effects Builder is our powerful and flexible system of creating effects on AI and *any* object in your scene without any code. We're discussing the Effects Builder early on because later chapters (Melee/Ranged attack setup) may use it to play animations.

Effect Builder should be attached to every object you want to use it on. It allows you to play multiple random audio effects, spawn particle effects, play Mechanim Animations, create physics effects, and more. It is added automatically default to most AI configurations from the main menu.
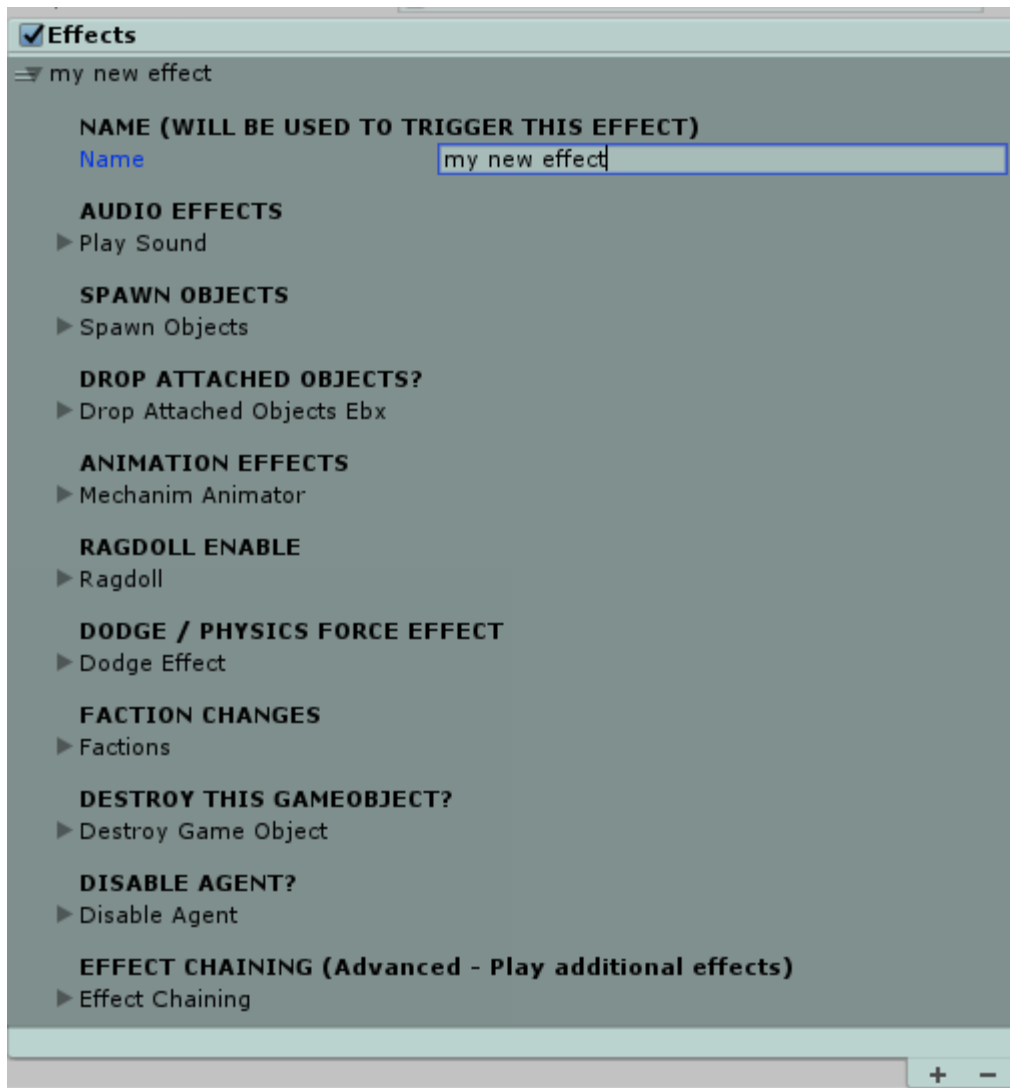
The best way to explain how it works is by example.

Example 1: You want the AI to drop its rifle to the ground when it dies. You'd use the Effects Builder to do that easily, without any code:

1) In Effects Builder, create a new effect - click "New Blank Effect". Name it something like "Death effect"

2) Under "Drop Attached Objects" click "Enable", click "Auto Add Rigidbody" and insert the your rifle model. The model must be attached as a child to your AI character, usually on its Hand bone. Auto-adding the Rigidbody allows the rifle to drop to the ground, based on your scene's Gravity settings.
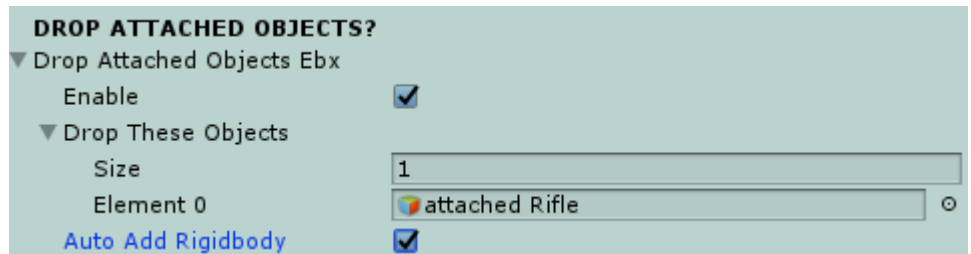
3) That's it for dropping objects. You most likely also want to play a death animation as well, so do this under "Animation Effects":

Click "Enable" on Mechanim Animator. Click "Play Animation State". Add the animation state name from Mechanim Controller. Set the CrossFade time to something low like 0.3 (do not use 0 or animations will be choppy). You can add more animation state names into these slots and Effects Builder will automatically pick a random one to play.
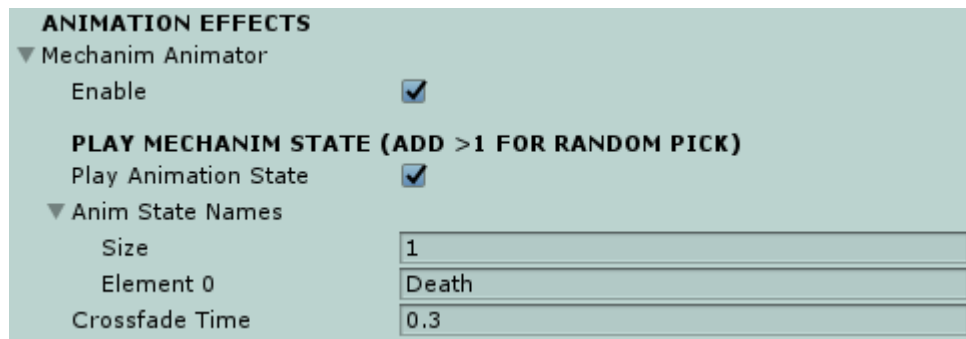
Now you're finished setting up the Effects Builder for the death effect. However, Effect Builder does not know WHEN to play that effect. So you have to trigger it.

Go to the AI's Health component. Scroll down and find the "{Effects Builder Integration}" section.

For this final step, add your new Effects Builder name into the slot, Effect On Death:

{ EFFECTS-BUILDER INTEGRATION }
Effect On Death   Death effect
Effect On Hit
Effect On Revive
Effect On Damage Avoided
Effect On Health Regen
Effect On Shield Regen
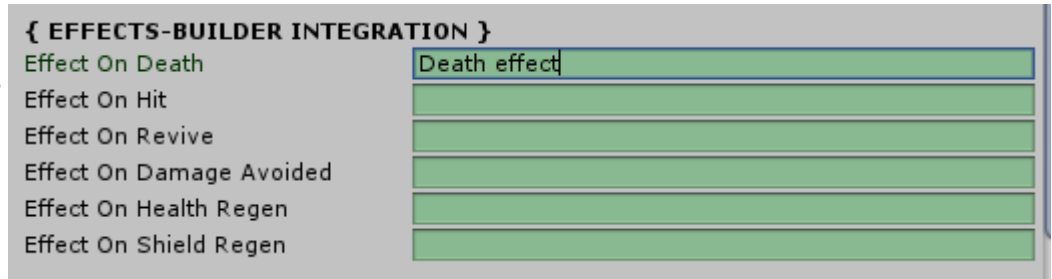
Now the effect will triggered by the Health component whenever the AI is killed.

Example 2: You want to play a shooting animation each time the AI shoots.

1) Add a new effect in Effects Builder. Name it something like "Shoot animation"

Shoot animation

NAME (WILL BE USED TO TRIGGER THIS EFFECT)
Name   Shoot animation

2) Under Animation Effects – Mechanim Animator – Play Mechanim State (same method as Example 1):
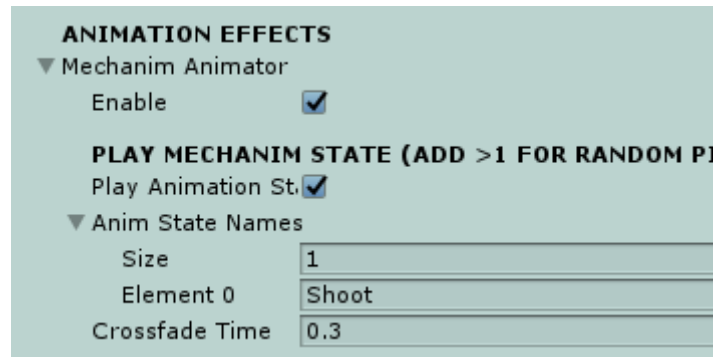
Add the animation state name, "Shoot" (assuming that's the name of the shooting animation STATE in your Mechanim setup)

Refer to your Mechanim Animation controller for the state names → → →
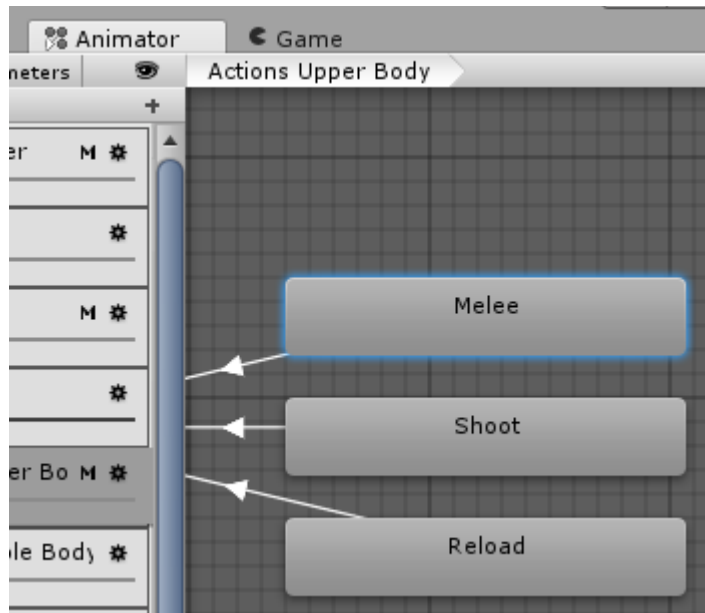
Now, your Effects Builder is set up for shooting. Like the death effect, you just have to trigger it.

On the AI, find the Ranged Attack Basic or Ranged Attack ADV controller. There should be a section called "{EFFECTS BUILDER}" Insert your effect name into the slot "On Fire One Shot"

9 | FOOTSTEPS AUDIO MANAGER

To add footstep sounds to your AI character, do the following:

1) Select your character, go to menu:

Component Menu → Footstep Manager
→ Add Footstep Manager

2) Insert your audio clips into the Default Footstep Clips section. Add as many as you'd like.
They will be played sequentially in order, or at random (Set Play Mode)

3) Set the speed to play the footsteps.
To do this, open up Play Speed Rules.

Add a new rule. This lets you set the min and max speed for the rule to trigger.

The Foot Step Interval is the amount of seconds to wait between each footstep sound.

For example, let's say you want the play footstep sounds at 1 step per second, when your character is walking. So set minSpeed to 0, maxSpeed to 1, and set the interval to 1.

But when your character runs, you want faster footstep sounds, at 2 steps per second. So set the minSpeed to 1, and maxSpeed to 3. Then set interval to 0.5 (half a second) to wait between steps. This makes it play faster. You can experiment with this to get the footsteps right for your specific character. The speed parameter is retrieved from the AI's attached Movement Controller.
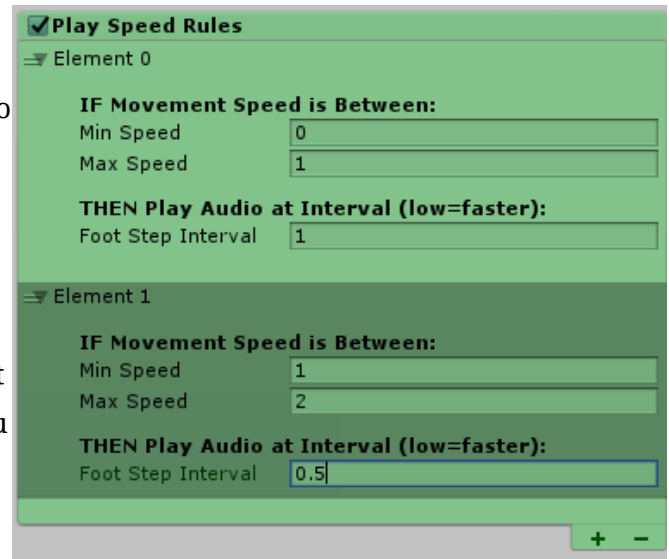
10 | SETUP MELEE ATTACKS

Melee attacks are driven by (1) Your weapon model, (2) A collider on the model, (3) Your Effects Builder animation, and (4) A melee damage component attached to the weapon itself.

So, you do the following steps:

1) Drag your raw weapon model into the HAND bone of your AI character. This ensures that when your character plays the Melee animation, the weapon sticks to the Hand.

2) You might need to spend some time rotating and positioning the weapon to fit the hand.
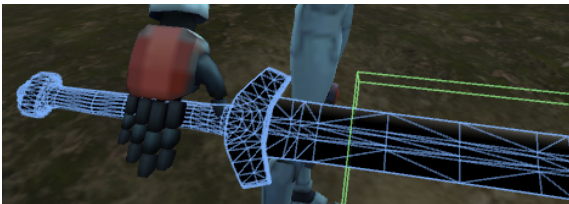
3) Add a box collider to the weapon. This is the Damage Area. If it's a sword model, position the box collider so it covers the blade part only. **Make sure to check "Is Trigger" or the AI will have strange movement behavior.**
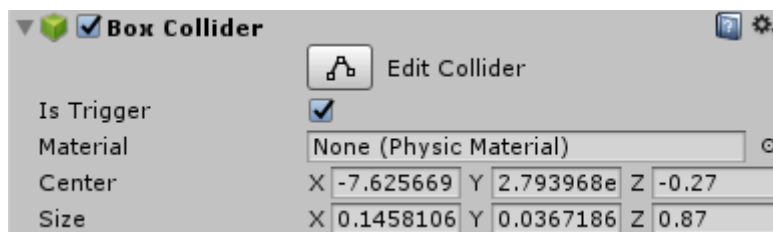
Fit the box collider around the melee weapon similar to this. Let the collider protrude a bit in front of the weapon.
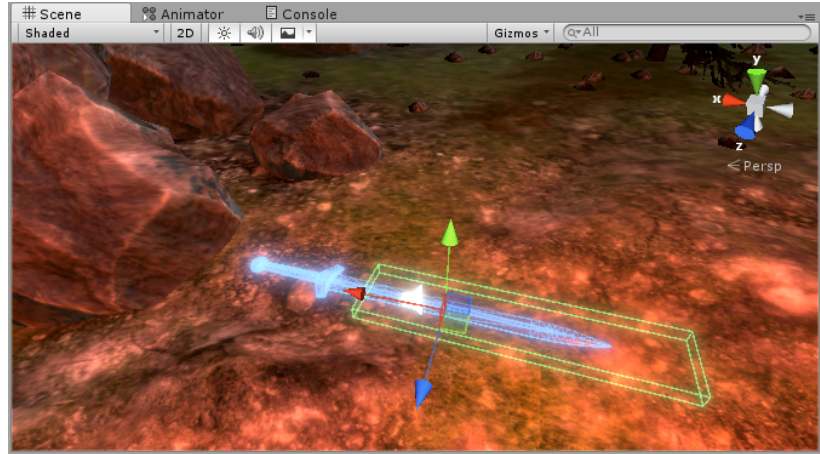
4) Select the weapon and add the "Melee Weapon" component:

Component Menu → AI Designer → Damage Behaviors → Melee Damage

5) Drag your main AI prefab into the "Self" slot to prevent the weapon from damaging the AI himself (unless you want it to self-damage).

6) Add a particle or other effect prefab into the On Hit Effect slot. Note: This is NOT a prefab in your scene, but an effect in your project folder. This will spawn when the weapon hits an object.

7) Add a sound effect to play when the sword hits an object. Drag an audio clip to the Sound Effect slot. Note: AI Designer automatically added an Audio Source to your weapon. Do not modify or add a sound to it.

8) That 's it. Leave "Damage Enabled" unchecked. It's for runtime debug usage.

Melee Weapon (Script)

AI DESIGNER

PRO

Script	MeleeWeapon

Send # Damage?

Damage	○────────	15

Ignore Self (Melee Attack Setup)

Self	Swordsman Bot (Team1)

Effects

On Hit Effect	On Hit 1
Sound Effect	human_footstep_02

Send attacker info to victim?

Send Attacker Info	✔

This is auto enabled at runtime to cause damage

Damage Enabled	

We're almost done setting up Melee attacks.

9) Go to your main AI prefab and add a Melee Attack Basic controller, if it's not already attached.   **Component Menu → AI Designer → Combat → Melee Attack Basic Controller**

Most of the templates automatically add this.

10) Now, drag your weapon into the "Melee Weapon Model" slot:
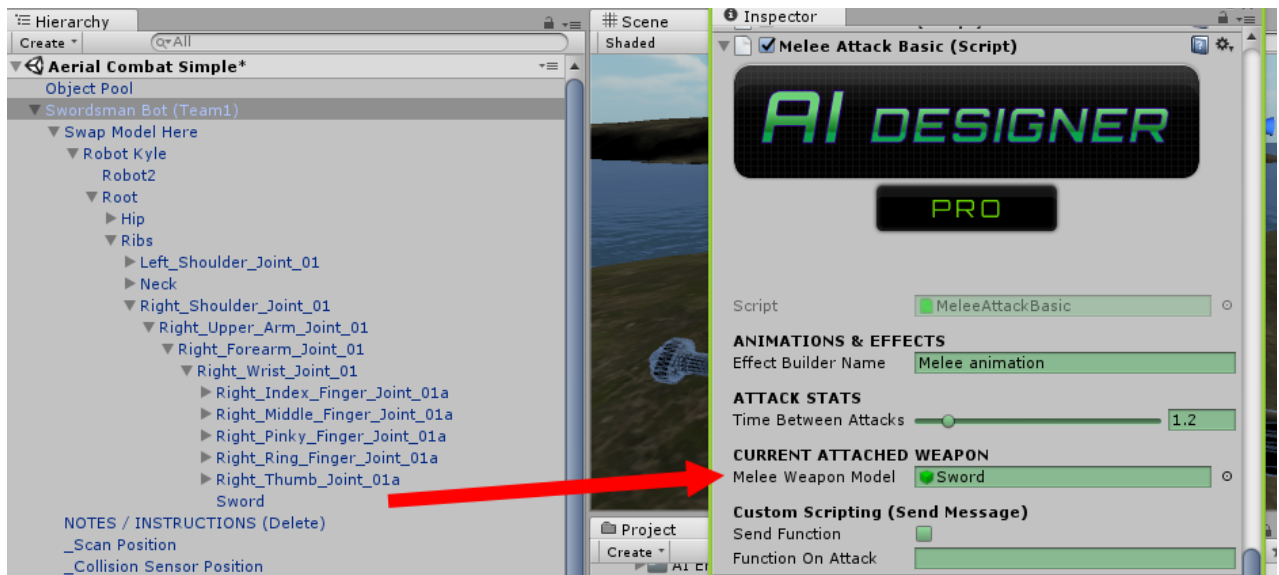
This step is important – otherwise the melee weapon damage function will not activate.

You now need to create an Effects-Builder effect for the Melee animation, which will tell the AI what melee animation to play.

So go to the Effects Builder, create a new effect called "Melee animation", insert your Mechanim Animation state name, and go back and insert the effect name into the Melee Attack controller (like in above screenshot).

11 | SETUP RANGED ATTACKS

For ranged attacks, you first need to set up the bullet, then add the Ranged Attack controller to the AI, and finally insert the bullet prefab into it.

1) Create a bullet:
The package comes with 3 demo bullets created for you and you can use those for prototyping. To make your own, here's how:

> a) Create a new game object, name it "My New Bullet"

> b) Add any bullet models, trail renderers, or particle effects as child objects under that.

> c) Click the main bullet object and go to menu:
> **Components → AI Designer → Damage Behaviors → Send Damage on Impact**
> This adds damage functionality.
> If you're using AI Designer PRO version, you can add Send Damage on Impact ADV, which allows for splash damage (Area of effect) and grenade mode.

> d) Click the main bullet object and go to menu:
> **Components → AI Designer → Weapon Scripts → Projectile**
> This adds motor functionality – it drives the bullet forward when fired.
> Note: This is an Editable C# script. If you change the script name, you can no longer add it from the Component menu.

> e) Drag the bullet into your Assets folder to turn it into a prefab.

> You're done with bullet setup.

2) Add a Ranged Attack Basic controller to your AI (if not already attached) . On the Pro version, you can add Ranged Attack ADV.

Components → AI Designer → Combat → Add Ranged Attack

3) Create a blank gameobject as a child to your AI, name it "_Shoot Position" Bullets will be spawned here. Position it a bit in front of the AI's body or the bullets might damage himself. If you're using Pro version, you can add the Ranged Attack ADV controller which lets you add unlimited Shoot Positions (for shotguns, miniguns, etc,).

4) Drag this "_Shoot Position" to the "Shoot Position" slot of your Ranged Attack controller.
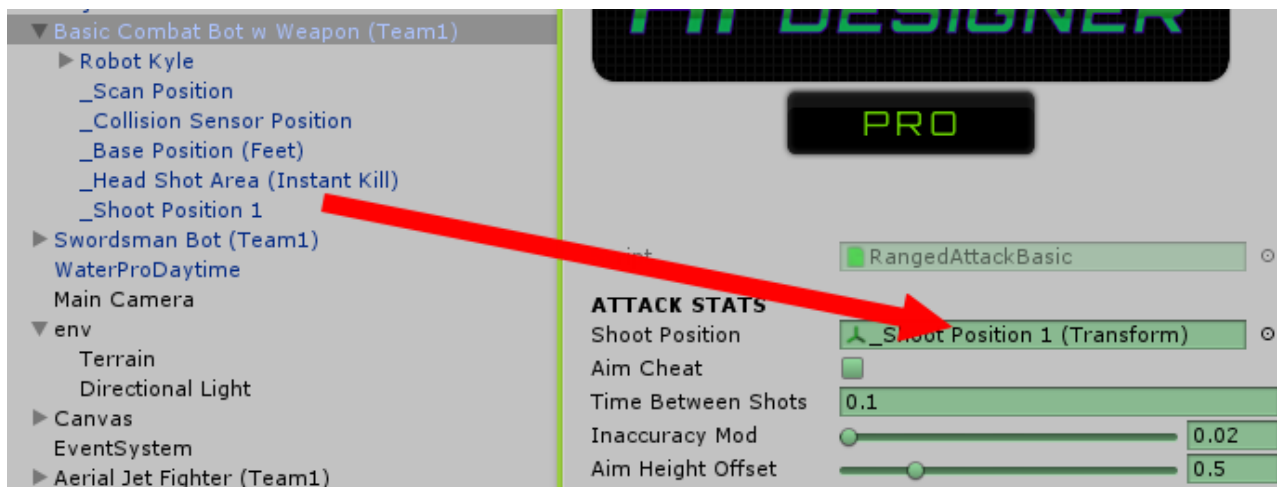
5) Insert the bullet prefab you created earlier into the "Bullet to Fire" slot:
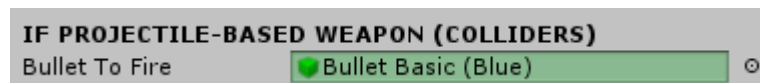
That's the basic setup.

If you want animations to play while shooting, you can create a new effect in Effects Builder and insert the effect name into the Ranged Attack controller's slot, "On Fire One Shot" It's recommended you read the EFFECTS-BUILDER section before proceeding.

12 | MOVEMENT ANIMATIONS

**HOW TO SET MOVEMENT ANIMATIONS?** Locomotion animations (walking, running) are first set in your Mechanim Controller. We provide an example controller "AI Designer Advanced Combat Controller". In your Mechanim controller, you'll need a float parameter called "Speed" (or anything you'd like). Set it to 0. Then create 3 animation states: Idle ← → Walk ←- → Run : The transitions depend on the speed parameter, so if Speed < 0.01, go to Idle animation, and if speed > 0.01 < 2, go to Walk, and if speed > 2, play run animation. The Movement Controller on the AI can automatically update this Mechanim parameter with its current speed. Simply go to the Movement Controller, find "Animator Speed Name" and insert the same Mechanim controller speed name into it, and check "Update Animator"
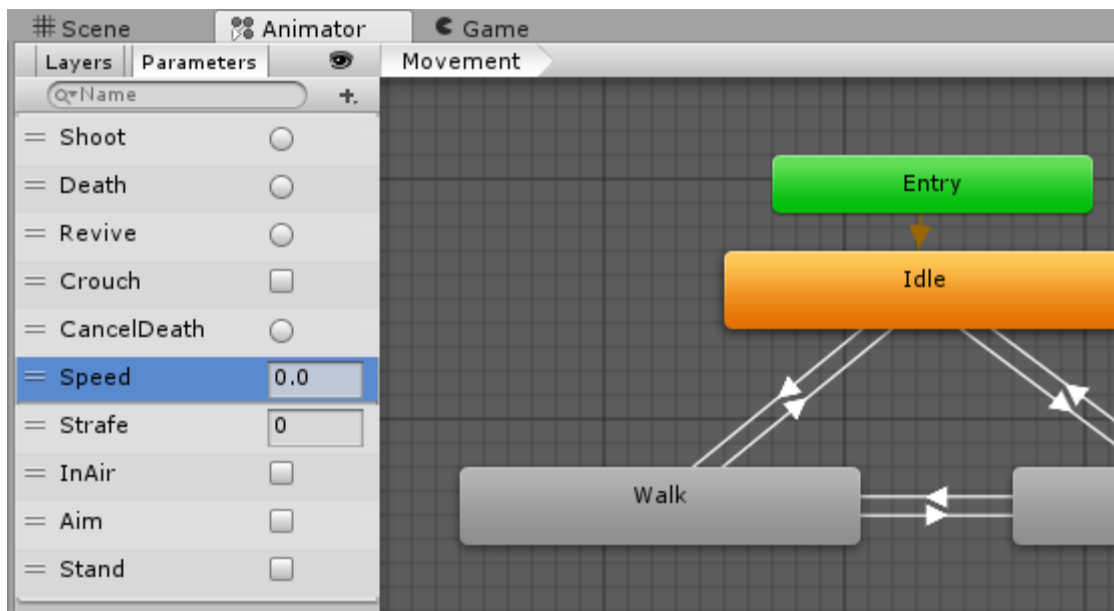
13 | MOVEMENT SETTINGS

The Movement Controller has many configuration settings.

Set speed, rotation speed, acceleration, stop distance:

MOVEMENT PARAMETERS
Top Speed — 36
Acceleration — 1
Brake Deceleration 0.01
Do Not Rotate In Air ☐
Target Speed 0
Rotate Speed — 1
Constant Y Rotation ☐
Stop Distance — 2

Pathfinding Method:
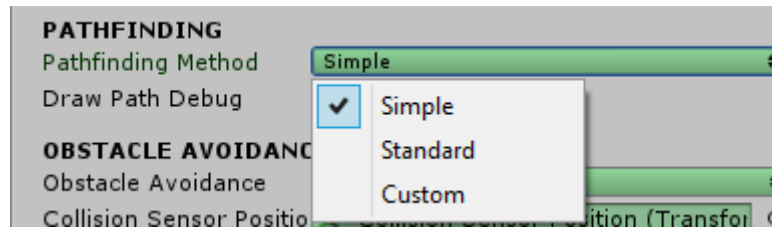
PATHFINDING
Pathfinding Method Simple
Draw Path Debug
 ✓ Simple
OBSTACLE AVOIDANC Standard
Obstacle Avoidance Custom
Collision Sensor Positio — ition (Transfoi ⊙
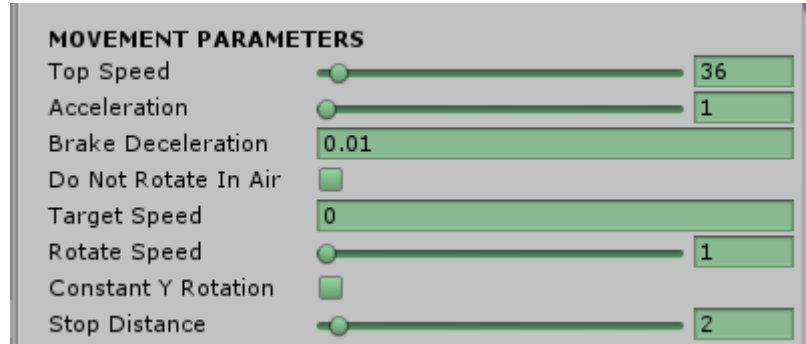
Simple pathfinding means obstacles are not taken into account when moving. Light on CPU. Good for very large battle scenes or open-world, terrain scenes. Does not require NavMesh baking.

Standard pathfinding blends our own pathfinding methods with Unity's built-in A-star pathfinding and **requires NavMesh-baking your scene.** For more info on Unity NavMesh, see Unity documentation. **NavMesh Agent component is NOT required and should NOT be added.**

Draw path debug: If using Standard Pathfinding, it will draw the AI's path, as a red line in the Scene Editor during gameplay. So switch over to the Scene Editor when you play the game to see this.

Obstacle Avoidance: Moves out of way of collisions. Requires a blank

OBSTACLE AVOIDANCE
Obstacle Avoidance Low
Collision Sensor Positio Position (Transfoi ⊙
Detection Distance None 5
 ✓ Low
MOVEMENT NOISE DI EMY AI)
Send Noise High
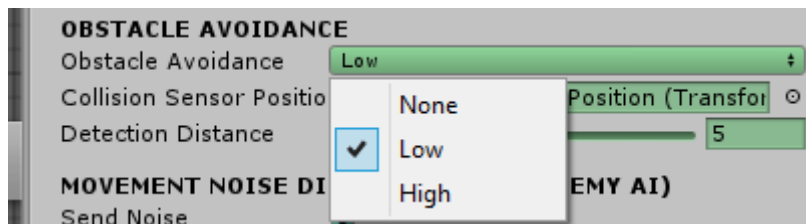
GameObject child'd to your character, placed at Chest level and inserted into the "Collision Sensor Position" slot.   If you're using Standard Pathfinding, set this to None. If using Simple Pathfinding, you can use this for a rough collision avoidance implementation since Simple Pathfinding does not have obstacle avoidance by default. If your AI is moving strangely or not moving at all, try setting this to NONE. Note: Setting it to HIGH with large numbers of AI in your scene may cause heavy CPU usage.

If your AI becomes overturned (upside down) it can automatically flip itself over, or stop moving altogether. **You usually don't need this unless your AI is a vehicle** (it will look silly for an

overturned vehicle to keep moving so checking this will either stop it from moving or flip it back up)

Send noise to alert other AI while moving.

Send Noise Interval – sends noise alerts each # seconds whenever the AI is moving. We don't recommend setting

this to a low number (frequent noise alerts) as it can slow down CPU.

Base Noise Distance – the minimum distance to send noise.

Noise Speed Multiplier – this multiplies BASE NOISE by current SPEED. So, the faster the AI moves, the more bigger noise it sends (bigger noise distance).

14 | HEALTH

Health, armor, shields are all set in the Health Controller. It also comes built-in with a health and shield regenerator. The Health Controller integrates with the Effects Builder, so you can input your Effect names into the slots (Effect On Hit, Effect On Death, Effect On Revive, etc.)

God Mode: Make the AI invincible – take no damage. Good for testing a new AI character against swarms of existing AI enemies. Also good for making AI companions in a RPG game that never die.

*Programmer Note: You can also replace this entire component with your own custom implementation if you're a coder. Check the box, "Use Custom Health Code" and input the name for your damage function. Do NOT remove the Health Controller. Each time it receives damage, it will use Unity SendMessage on your function name with **additional parameter of (float damageAmount)** so your function will need to take in 1 argument. Your custom function code must be attached to the AI gameobject.*
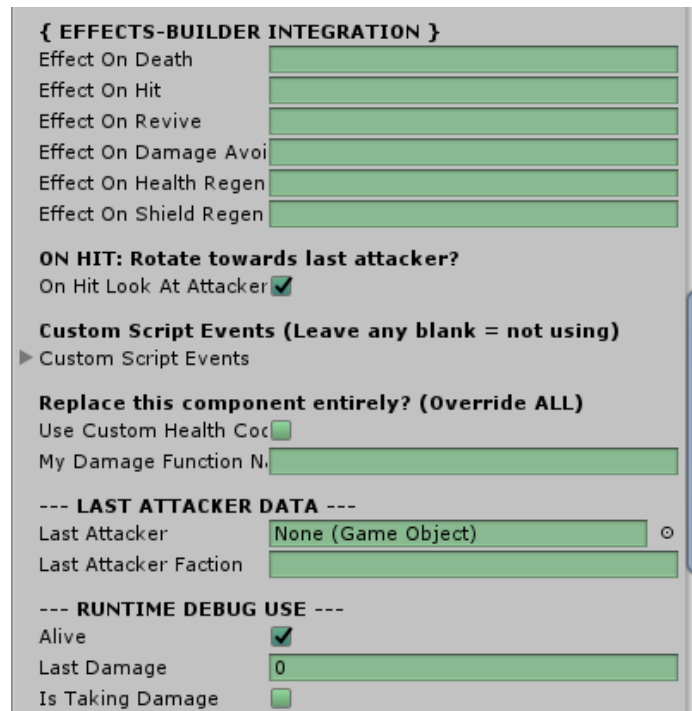
15 | THE BRAIN COMPONENT & ADDING LOGIC

Every AI character has a Brain component, where all the central logic happens. While you configure the AI's parameters using its Controllers (movement speed, attack range, etc.), you put it altogether using the Brain component.

The Brain component lets you visually build AI actions without any coding. It is stuffed with power and features so it may seem daunting at first, but as you explore it, you'll realize that workflow is quite simple but the number of possibilities is great.

The Brain component comes as 2 main parts:

AI Action section

Flags section

The AI Actions section holds logic for the AI – when to perform certain actions based on conditions and triggers.

The Flags section holds a set of Flag names for the AI's components to communicate with each other, or with your own scripts. This is for advanced integrations only and most use-cases don't need to touch this.
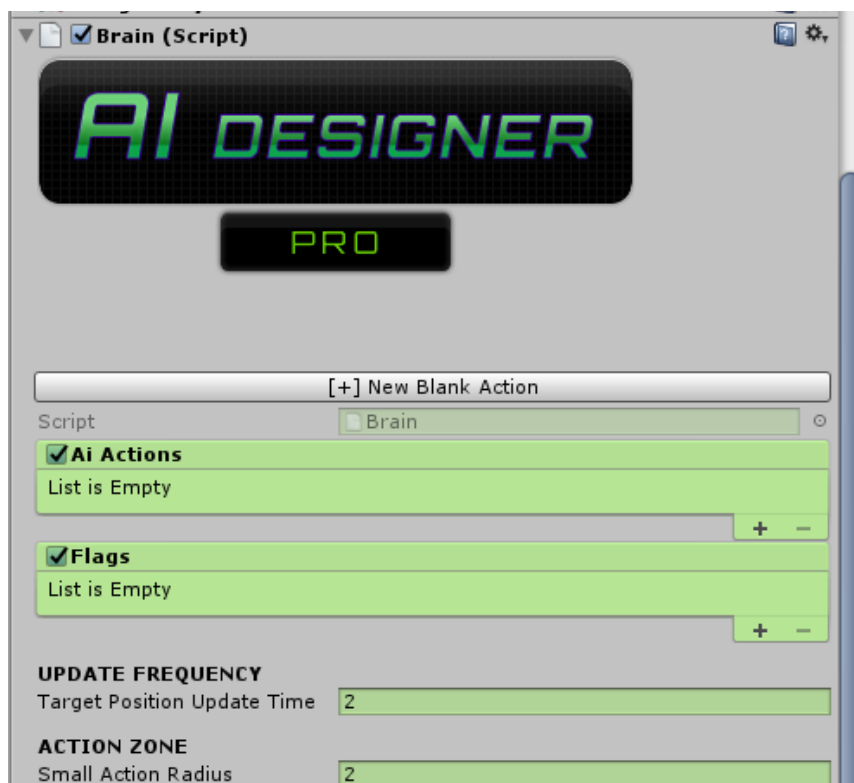
AI ACTIONS

Clicking "New Blank Action" will create a new action group.

Name it something unique like "Chase enemy" that you can understand. If you have script integrations, you can refer to this action group by this name (case sensitive)

During development, you might want to temporarily turn off this action, just check "Disable This Group"

How Often to Check Conditions?

If you want the AI to run this action each tick (if conditions are met), then check "Check Each Tick". If you only want this to run once, click "Check Once". Only pick one.
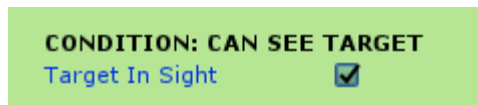
CONDITIONS & ACTIONS

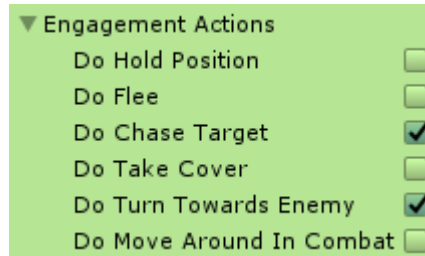Each AI Action group is split into 2 sections: CONDITIONS and ACTIONS.

As you might guess, the ACTIONS are what you want the AI character to do. The CONDITIONS are parameters that tell the it WHEN to perform the actions.
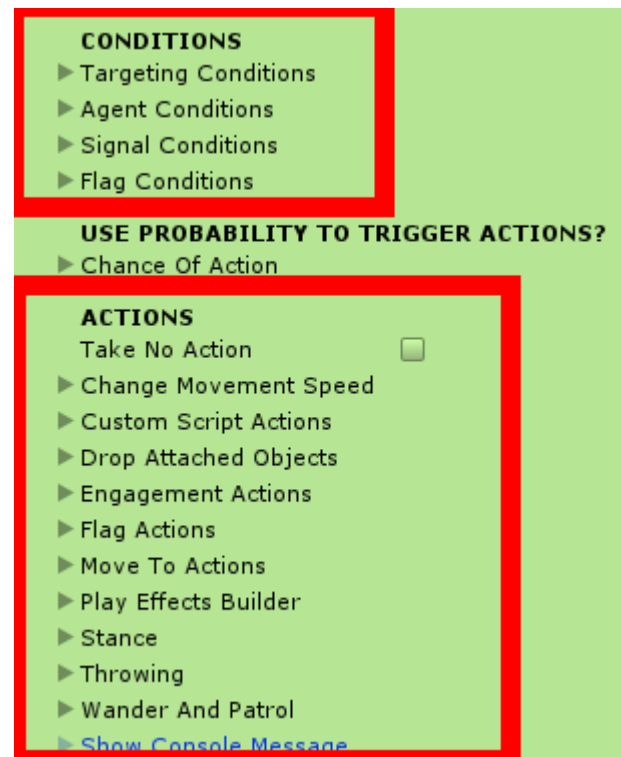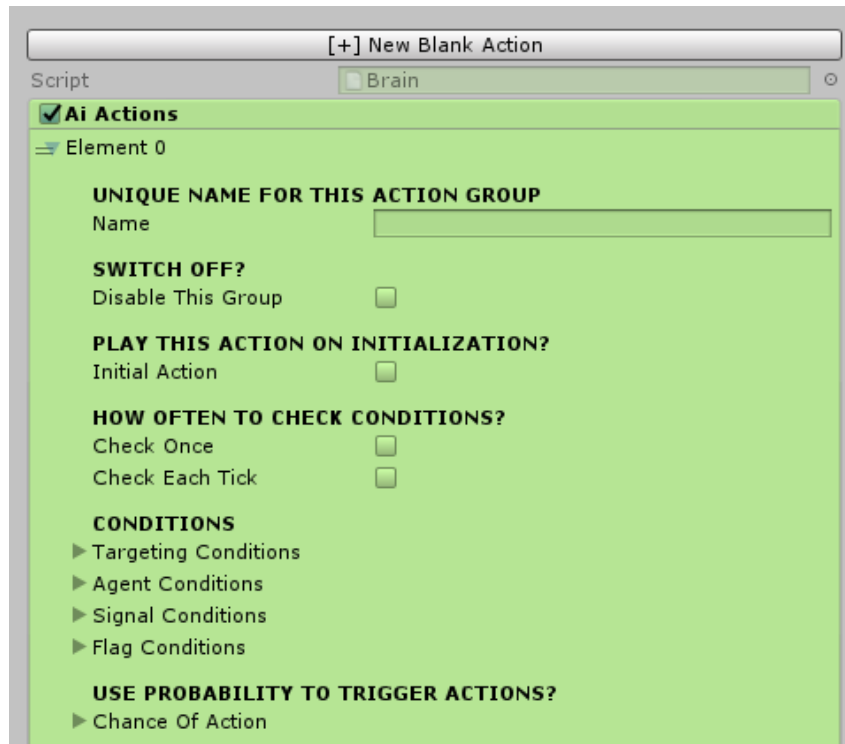
For example, if you want the AI to chase enemies on sight, you'd set the following condition:

And the following action:

Why is "Do Turn Towards Enemy" set? Because whenever an enemy is within range, the AI now DOES NOT HAVE TO look at the enemy or the point it's moving to. Why? Let's assume this: Your AI is sandwiched between a health pack he desperately needs and an enemy. The enemy is in front of your AI. A health pack is behind your AI. The AI needs the health pack behind him. But the enemy is in front of him. So: Should the AI run to pick up the Health pack AND shoot at the enemy at the same time? If yes, the AI should always look at the enemy, shoot at him, AND run BACKWARDS to get the health pack. (Run backwards while shooting forwards). BUT, if no, then the AI will TURN AROUND to look at the health pack while running to it and cannot shoot at the enemy.

So, this setting "Do Turn Towards Enemy" tells the AI to look at the enemy, whenever its perform this action, even if the AI is running the other way.

That's it for adding a "Pursue enemy" behavior.

16 | INTRODUCING THE FLAG SYSTEM

This AI system was meant to be used without any coding. However, we provided some very powerful and flexible features for programmers also.

Continuing our previous example with the Brain component... Let's say you have a third-party script and you want to know WHEN a specific AI is chasing enemies, meaning, when it has activated the CHASE behavior you just created.

Easy: Add a new flag on the AI, name it something like "I_AM_CHASING_ENEMY"
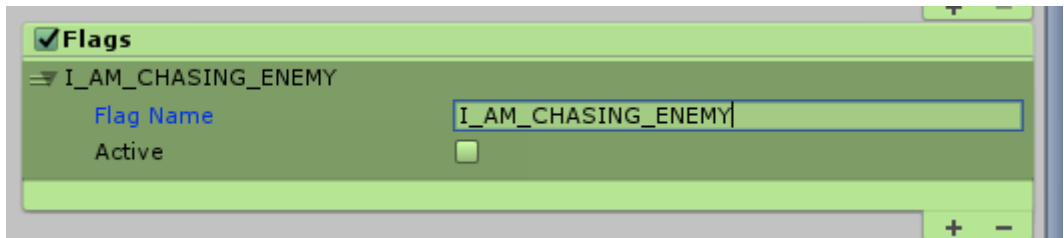
Then under Actions, tell the AI to activate flag whenever it's chasing enemies:
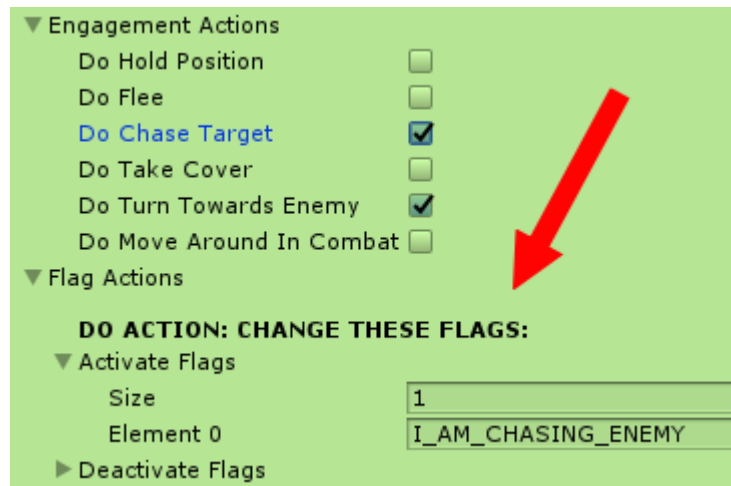
Now, go into your script and add the following line:

```
if (GetComponent<AIDesigner.Brain>().GetFlag("I_AM_CHASING_ENEMY") == true){

        // do something...

}
```

That's it. You should put that in a Coroutine or InvokeRepeating to check for Flag activation.

You can also activate or deactivate a flag in order to trigger certain actions. Let's say you want the AI to move to its home base building using your script.

Create another flag, "MOVE_TO_HOME_BASE"
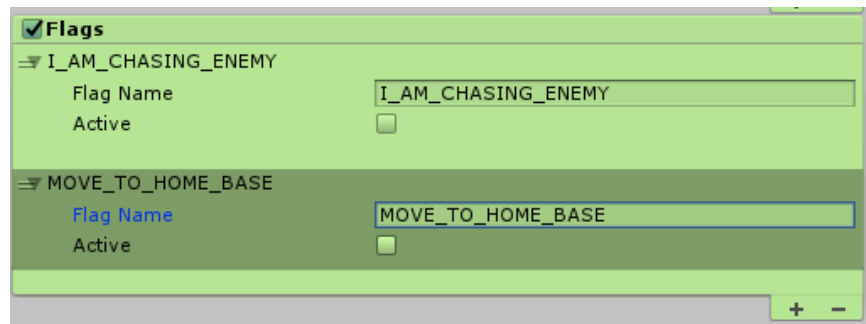
Create a new AI Action called "Go Home"

Add a new Flag Condition (same flag name you just created):
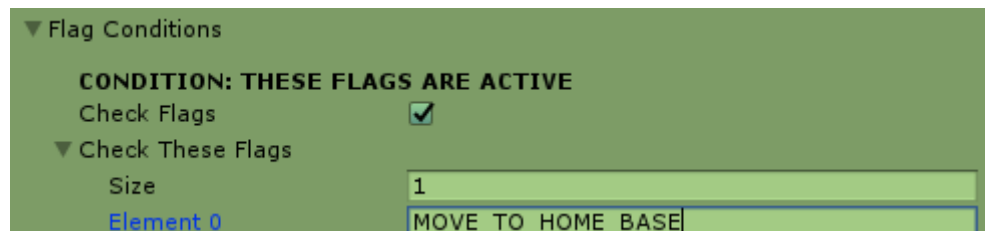
Under the Actions section, open up "Move To Actions" and click "Do Move To Object". Drag your GameObject in scene into the slot. When triggered, the AI will move to this object.
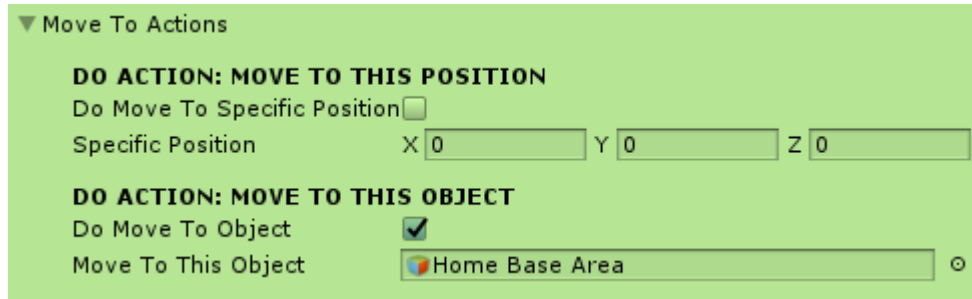
Finally, in your own script, add the following:

GetComponent<AIDesigner.Brain>().SetFlag("MOVE_TO_HOME_BASE", true);

Now the AI will activate the flag "MOVE_TO_HOME_BASE", which will automatically trigger the Action group for moving to your specific object.

Note: You can also DE-ACTIVATE a flag by using false:

GetComponent<AIDesigner.Brain>().SetFlag("MOVE_TO_HOME_BASE", false);

and this will stop the AI from moving to that specific object.

These are just simple examples of how to use the Flag system with third party assets and scripts. You can imagine the kinds of scenarios that this would come in handy.

Global Flags: Global flags use the same concept as regular flags, except global flags are shared by ALL agents in a scene. In your Brain actions, you've seen Flag conditions and "Global" Flag conditions – that's what these refer to.

To set up global flags, go to the AI Designer / Prefabs folder and drag the **"AIDesigner Global Flags"** prefab into your scene. Then add as many flag names as you'd like. They "usually" can be the same names as your non-global flags, but it's good practice to use different names.

17 | CULLING SYSTEM FOR LARGE BATTLES

AI Designer comes with a built-in culling system. It can automatically deactivate AI functions and renderers when too far from the main camera. In large battle scenes, this can significantly increase performance. Sometimes, you don't want the AI play out its behavior when your player can't even see it. In those cases, it's best use the built-in culling system.

1) Select any AI Designer character.

2) Go to Component Menu → Culling and Performance → Add Culling System

You should get this:

If you want to disable renderers (graphics) on your character, check Disable Renderers and type in the range to disable (from main camera)

If you want to temporarily disable all AI functionality and CPU usage, check Disable Agent and set the range to do it.

If an agent is disabled and the main camera comes back into range, it will be reactivated and should continue its previous operations.

18 | SETUP WANDER BEHAVIOR

This assumes your AI character is already setup using the Tools menu → AI Designer, either using [Create blank AI] or from a template. The templates usually have the Wander behavior already setup for you. To do it from scratch, follow these 3 steps:

1) Select character, go to Component menu → AI Designer → Controllers → Wander Controllers

2) Edit parameters:

Min / Max Path Change Time: The random (min/max) amount of seconds to wait before changing to a different random path.

Idle Start Time: Every # seconds, the AI will stop wandering and go idle. Set this to 0 to make the AI move nonstop.

Idle Time: # seconds to wait when it goes idle

Wander Type:
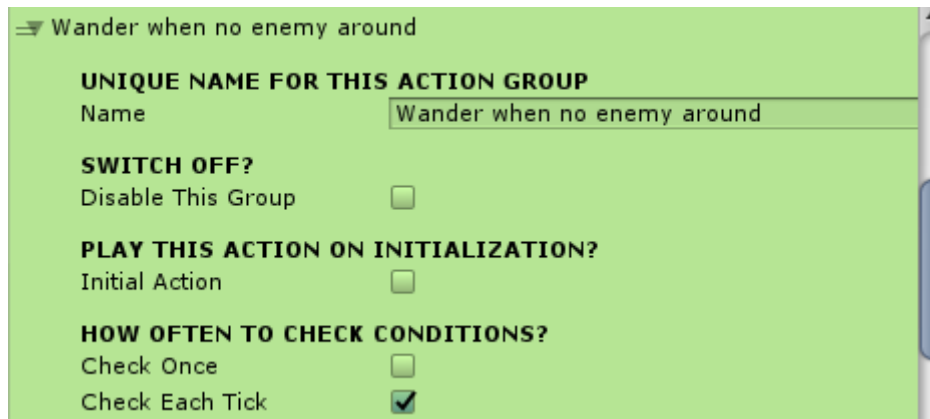**Wander from Self Position** – make the AI pick a random wander position a distance from its current position. When it arrives at that destination, it will pick another random point, but from that new destination.
Wander from Custom Object – make the AI pick random wander position near a custom object (fill in Custom Object)

– see next page for further instructions -

3) Final step: You need to tell the AI to actually use the Wander Controller and WHEN to use it.

It needs to be activated from the Brain component. So go to the Brain and add the following action:

Wander when no enemy around

UNIQUE NAME FOR THIS ACTION GROUP
Name   Wander when no enemy around
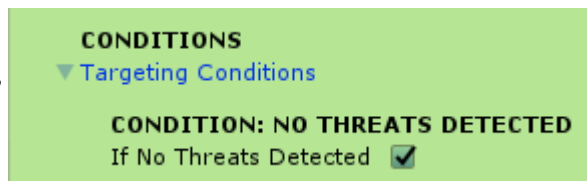
SWITCH OFF?
Disable This Group   ☐

PLAY THIS ACTION ON INITIALIZATION?
Initial Action   ☐

HOW OFTEN TO CHECK CONDITIONS?
Check Once   ☐
Check Each Tick   ☑

Under TARGETING CONDITIONS, select "If No Threats Detected"

Leave RANGE MODE to "Do Not Check"
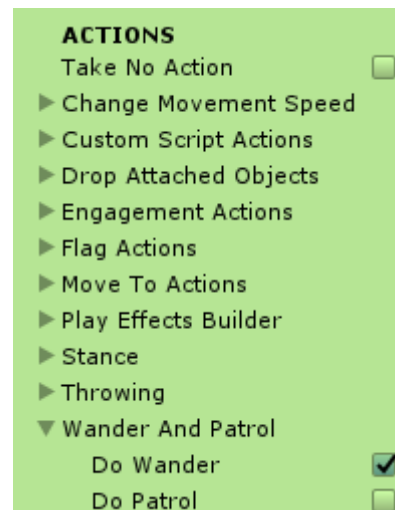
CONDITIONS
▼ Targeting Conditions

CONDITION: NO THREATS DETECTED
If No Threats Detected ☑

Scroll down to ACTIONS section and find "Wander and Patrol" at the bottom.

Check "Do Wander".

ACTIONS
Take No Action   ☐
▶ Change Movement Speed
▶ Custom Script Actions
▶ Drop Attached Objects
▶ Engagement Actions
▶ Flag Actions
▶ Move To Actions
▶ Play Effects Builder
▶ Stance
▶ Throwing
▼ Wander And Patrol
    Do Wander   ☑
    Do Patrol   ☐

That's it. Now the AI will start randomly wandering whenever there are no enemies around.

19 | SETUP PATROL & WAYPOINT PATHS

The Patrol Controller lets you define a specific path for the AI to follow. This is great to create guards on patrol, scripted scenes, drawing race tracks, and more. When on this path, the AI will usually not deviate unless enemies are in sight. Basically, you drag the Waypoint System prefab into your scene, set up the waypoints, and tell the Patrol controller to process the waypoints.

1) Drag a Waypoint Group prefab into your scene. (found in Assets / AI Designer Pro / Prefabs). You'll see 3 waypoints in your scene. If you were to use this untouched, the AI will follow the path of these 3 waypoints. Click on each individual waypoint (may need to click twice) to select and move them around.

You can add unlimited waypoints by clicking the main Waypoint Group object and clicking "Add Waypoint" in the inspector.
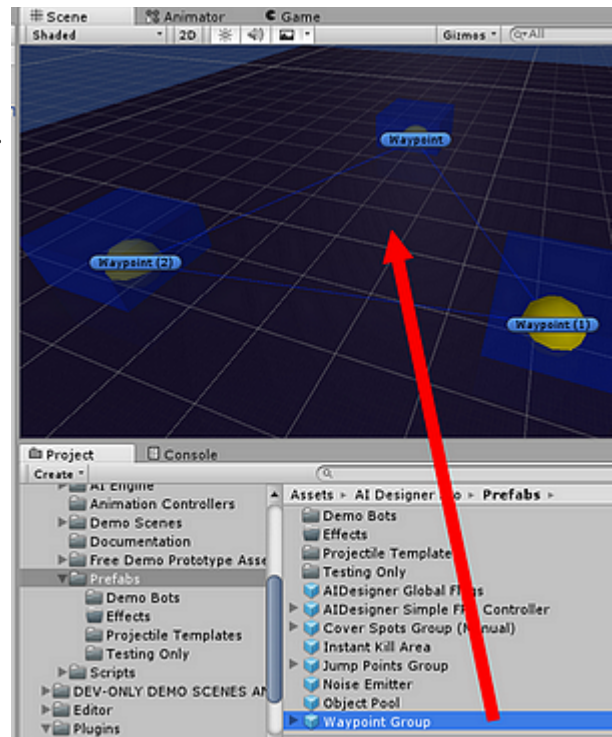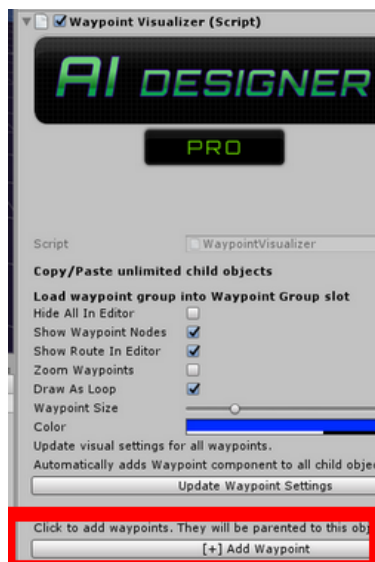
Place each waypoint at ground level, with about half the yellow circle showing above ground, and half underground.

NOTE: If the waypoint is placed too far from ground, the AI will move to it but will act strangely – run in circles, playing walk animation but not moving, etc. When this happens, simply adjust the Y-position of this waypoint.

2) With waypoints set up, add the PATROL controller to your character:
Component Menu → AI Designer → Controllers → Patrol Controller

3) Drag the Waypoint Group in your scene into the "Waypoint Group" slot of the Patrol controller:

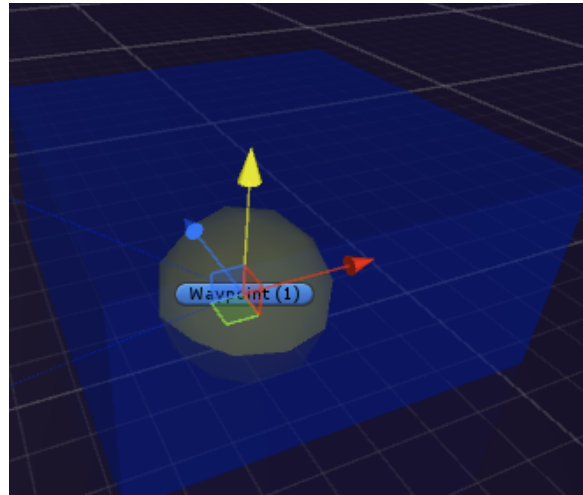This tells the AI which waypoint group to process. You can have multiple waypoint groups in your scene, with each used by different characters. So if you have a castle game with 2 towers, and guards patrol Tower 1, and another set of guards patrol Tower 2, you would create 2 Waypoint Groups, and assign one to one set of guards, one to the other.
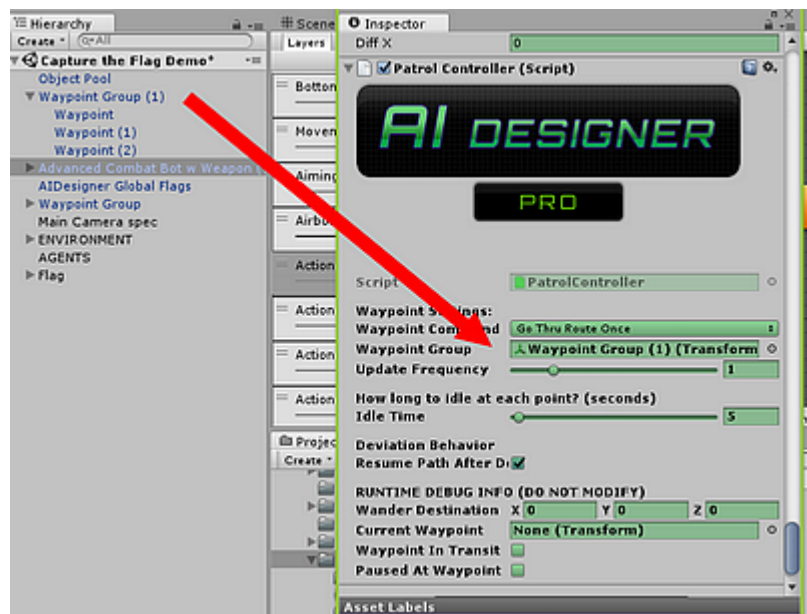
4) With the Patrol controller setup, you need to tell the agent WHEN to patrol. So, like the Wander setup, go into the BRAIN component and create a new action:

Name it something like "Start Patrol if no enemy detected" - because that's what we're going to tell it to do: Only patrol if there aren't any enemies nearby.
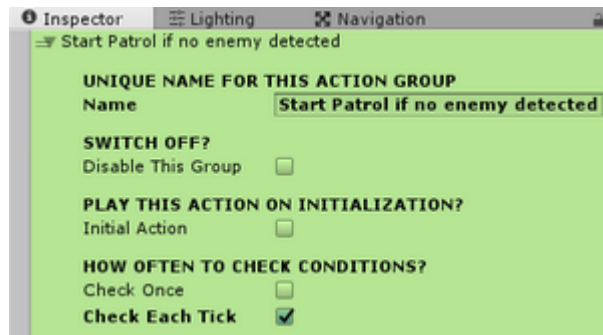
Set the TARGETING CONDITION to "If No Threats Detected"

Under ACTIONS, find Wander and Patrol and click "Do Patrol":
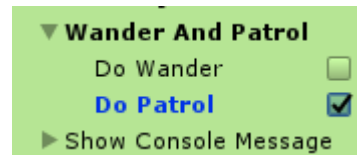
That's it.

Note: If you have another action that plays "Do Wander" it MUST be disabled! "Do Patrol" and "Do Wander" cannot work together because the AI cannot wander randomly and patrol a specific path simultaneously. If you accidentally leave "Do Wander" unchecked, the AI may behave strangely, or not move at all.

20 | SETUP FLEE BEHAVIOR

(Pro Only)

Note: This doc section is being worked on. BUT, the feature is available. To see how it's set up, see the demo scenes folder.

21 | SETUP COVER-FINDING BEHAVIOR

Cover behavior gives the AI the ability to seek out hiding spots to take cover. The hiding spots can be dynamic (decided at runtime) or manually-placed by you.

<u>**TO MANUALLY SET UP COVER SPOTS:**</u>

(If not using manually-placed spots, but you let the AI figure it out on his own, skip to next page)

1) **Drag the Cover Spots Group prefab into your scene** (Assets / AI Designer Pro / Prefabs)
You should see 3 green markers:

2) Like the Patrol/Waypoint setup, you simply click and **move each Cover Spot to your desired location,** with about half the yellow circle sticking out of ground level and half under it. Make sure it's a reachable location for your characters, or there will be strange behavior.

3) Each Cover Spot can be rotated – this will be the rotation your character will follow after reaching it. The rotation is represented by the RED line sticking out of the cover spot – it acts as an arrow.
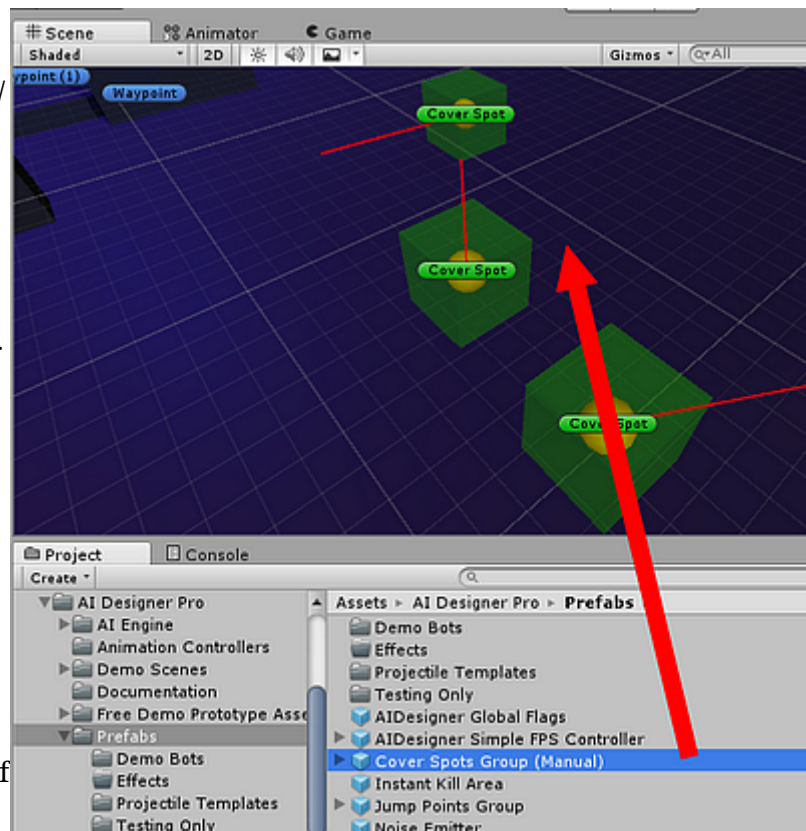
Example Usage: You want the AI to take cover behind a box. Once he gets there, he should turn around with his back towards the box. To do this, move a Cover Spot to one of the sides of the Box and rotate the Cover Spot to face outwards.
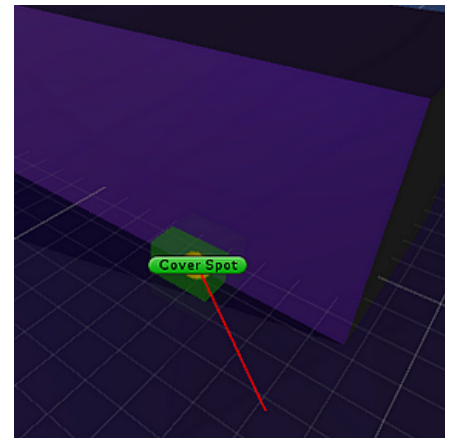
4) If you're manually-placed cover spots are set up, it's time to set up your AI.

Select your character, go to Component Menu → AI Designer Pro → Controllers → Cover

Note: if you manually placed cover spots, each cover point must be tagged as "CoverSpot" in the Unity editor:

This is the major difference between setting up Patrol Waypoints and setting up cover spots – you never load the cover spots into the AI itself – it automatically scans and processes all points tagged as "CoverSpot"
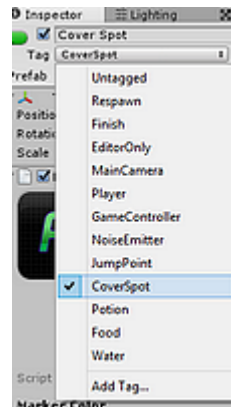
5) Setup the parameters on the Cover Controller. Proximity means the distance to find nearest cover spot.

If there aren't any cover spots within this distance, the AI will not go to cover.

Size Preference – the general height of cover. You might wander why you can't set a specific height. That's because if the system were to check for specific heights during gameplay, it can severely bog down your players' CPU with too many calculations. For the most part, leave this setting as "Let AI Decide"

Cover Location: Dynamic means the AI will pick a spot, even if you don't have any Cover Spots setup. However, the behavior is unpredictable, so if you have time, it's recommended you use Dynamic AND Manually-placed cover spots.

If you select Manual, you must have cover spots set up in your scene. Otherwise the AI won't do anything.

For the final step, you need to tell the agent WHEN to look for cover:

6) Create a new action in the Brain component:

Under actions, find Engagement Actions and click "Do Take Cover"

This section shows ONE method of making the AI jump. This method can be done in both the Lite and Pro versions of AI Designer. *Note: The Pro version has an advanced method of jumping that lets you place jump points in your scene to tell the AI exactly WHEN to jump (e.g. on rooftops).*

You can make the AI jump by using the "Dodge Effects" portion of the Effects Builder. You can also use this to make him dodge left/ right, forward backward, as well as up (jump) down.

We will tell the AI to jump during certain conditions (e.g. during combat, when enemy is within melee range, when health is low, etc.). Basically, you create an effect in Effects Builder that propels the character up (under Dodge Effects). Then in the Brain component, create a new condition (health is low) that triggers this effect name. That's it, really.

Let's create a behavior that tells the AI to jump back when its health is low:

1) Go to the Effects Builder and create a new effect, name it something like "Jump":

2) Open up "Dodge / Effect", click "Enable"

3) Set "Back Force" to 200,
"Up Force" to 300
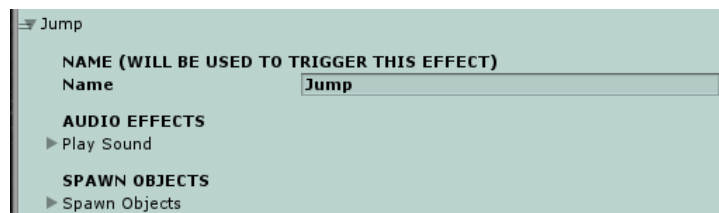
(optional): You can open Mechanim Animator section and input a jump animation.

4) Now go to the BRAIN component. Add a new action group. Call it "Jump Away"

5) Open "Agent Conditions" and click "Check My Health".
Set "Min My Health" to 0, "Max My Health" to 50.
This means: Trigger this action when Health % is lower than 50%

6) Scroll down to Actions, and open up "Play Effects Builder":
Click "Do Play Effect"

Set "Effect Name" to "Jump" (exact same name you used in Effects Builder – Step #1)

7) Now, we don't want him jumping all the time, when health is below 50. That would look strange. So let's add probability of this happening:

Open up "Chance of Action". Click "Enable Chance". Set "Chance" to 25.
Now there's a 25% chance of jumping back when health is low. This random chance roll happens every 1-2 seconds

(see next page for screenshots)

That's it for this tutorial.

23 | JUMPING ROOF-TOPS & MORE

(PRO ONLY)

AI Designer Lite and Pro both have jumping, but only the Pro version lets you place Jump Points in your scene to tell the AI exactly where to jump (e.g. over rooftop). For example, you can place a series of jump points at the edges of rooftops. When the AI comes near it, it will automatically jump. The rotation of the jump is determined by the direction your jump point faces (represented by red line)

SETUP:

1) Drag the Jump Points Group prefab into your scene. (Assets / AI Designer Pro / Prefabs)

2) Like waypoints, drag each jump point to desired location, such as the edge of a rooftop.

A thin red line is drawn from each point. That represents the direction of the jump. You can rotate the jump point to desired direction.
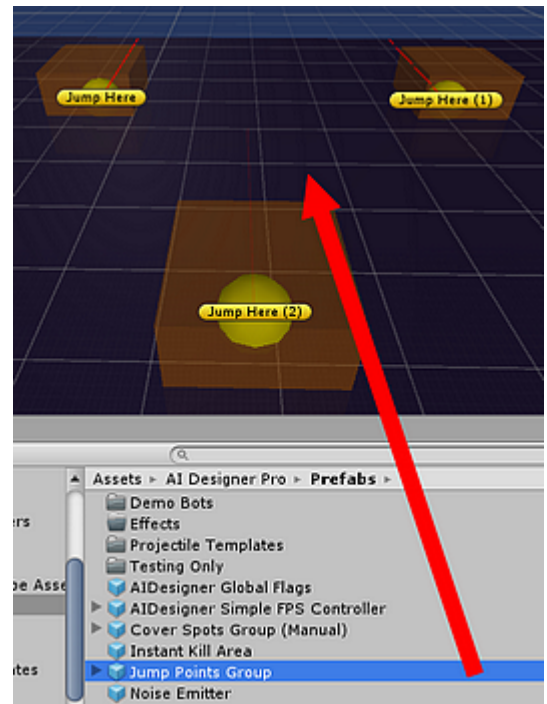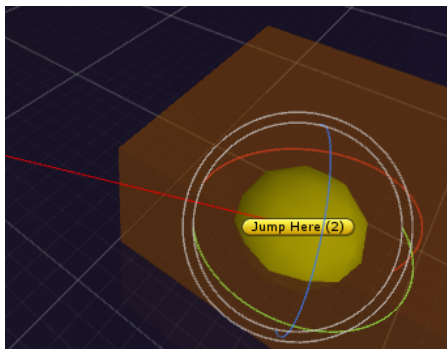
3) After you've finished placing each jump point, select your AI and **add a Jump Controller:**
Components Menu → AI Designer Pro → Controllers → Jump Controller

4) Setup the parameters:

Up/Forward Force: the physical force exerted for the jump – this is affected by Rigidbody gravity and other physical forces in your scene

Jump Delay Time: the # seconds to wait before actually jumping – this needs to be tuned with your animation

Dist to Jump Point: How far should the AI be from a jump point, to detect it and jump?

Jump in Point Direction: If checked, the AI will jump in the direction of jump point (the rotation / dir of red line you set in the editor). If unchecked, the AI will jump in the direction it's currently facing which may lead to fun but unpredictable behavior.

OnJump / OnJumpLand: the name of the Effects Builder effect. You should create a new effect in Effects Builder. Name it something like "Jump" and set the Animation to your jump state.
Then insert the effect name into OnJump.   OnJumpLand can be used to trigger an Effect Builder that spawns dust particles at the feet upon landing.
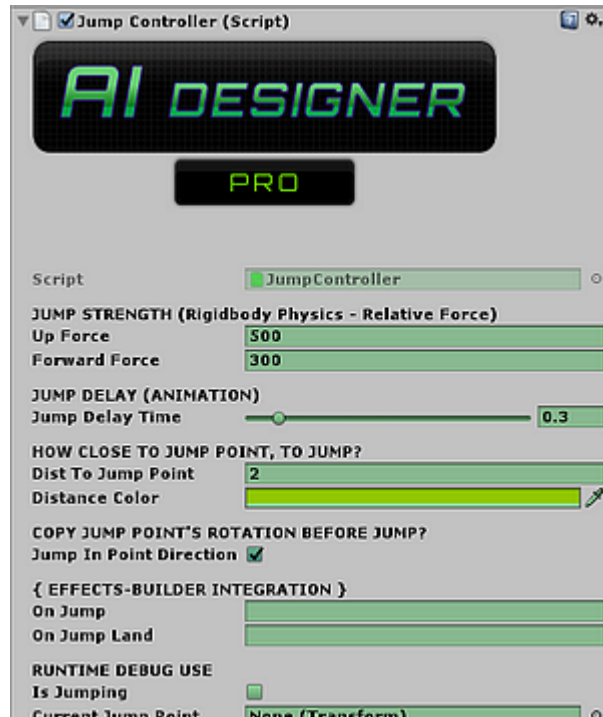
Effects Builder example   → → → →
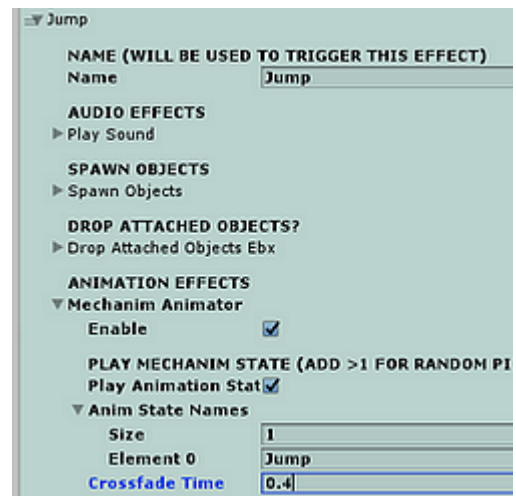
24 | BURST MOVEMENT

The Burst Movement Controller gives AI the ability to suddenly "charge" ahead for a set amount of time. There's no complicated setup – simply add the Burst Movement Controller!

Components Menu → **AI Designer Pro** → **Controllers** → **Burst Movement**

You set the amount of seconds to auto burst (or use Flag triggers). Then set the burst speed and acceleration. That's it.

To see this in action, play the Demo Scene: Burst Movement (Charging)

25 | MOVEMENT POINTS / GETTING TIRED

You can simulate AI getting "tired" after moving for a while, and stopping to wait to recharge. This is great for wildlife or other biological units.

 This is done by the Movement Points Controller:

Components Menu → AI Designer Pro → Controllers → Movement Points

You can set total points, spending rate, and auto-recovery parameters.

To see this in action, see the Demo Scene: Movement Points (Get Tired)

26 | PROCEDURAL ANIMATIONS

(PRO ONLY)

The Procedural Animator component blends in procedurally-generated animations with your own provided animations. Together, they create more variation.

If paired with the Advanced Range Attack module, your character will twist and turn slightly towards his target. The Basic Range Attack does not have this feature and the character will always point straight ahead.

The Procedural Animator is currently available only for Chest areas – the other areas are planned to come in a future update.

(see next page for settings)

USAGE:

Select your character and go to menu:

Component → Procedural Animation → Add Procedural Animator

SETTINGS:

Allow Bend Up/Down:

This allows this controller to procedurally bend your character's chest / torso area up/down. If you're using the Advanced Ranged Attack module, it will twist the torso to aim up and down. If aiming down, it should look like the character is bowing.

Allow Bend Left/Right:

This allows this controller to bend the torso/chest area left and right. If using the Advanced Ranged Attack controller, it will bend and slightly twist the torso to the side, left/right to aim at the target.

Random Procedural Idle

Auto Play Random: Check this to allow auto procedural animations

Time to Change: The amount of seconds to wait before changing to a new procedural pose

Advanced Settings:

These settings are usually not touched but are included for your flexibility. They're good for extreme cases, such as simulating Zombies. (See the Zombie demo scene)

27 | NEEDS & GOALS TUTORIAL (Simulate Hunger and more)

{ PRO ONLY FEATURE }

<u>**INTRO**</u>

AI Designer Pro comes with a powerful system of Goals and Needs to drive complex behaviors. This is great for animals / wildlife, village-based NPCs that simulate daily life, or in more complex action games such as Capture the Flag.

We can only give a basic, general tutorial on this because there are so many ways to use this with a variety of game types and it's not possible to cover every single use-case in this documentation.

<u>**TUTORIAL: Simulate Hunger**</u>

1) First, make sure your AI is already set up with basic components and parameters. If it's a combat AI, make sure its combat capabilities are fully operational. If it's a neutral wildlife animal, make sure it can wander. The point is, adding Goals and Needs should be one of the LAST steps you make to your AI, after everything else is done.

2) Set up and define NEEDS:

a) Add the Emotion Controller (Components → Controllers → Emotion Controller)
b) Open the Needs section, add a new Need. Set the Var Name to "Hunger". Set the value to -100.

What does this mean? The Var Name of "Hunger" defines this need to represent

"Hunger".

The value of -100 means the AI has a "Hunger" level of -100, meaning, it is definitely NOT hungry. So anytime there's a NEGATIVE value, then the AI does NOT have this need. Therefore, if the need is positive, then the AI has the need. So if the value goes from -100 up to +50, then the AI should be hungry. Now set the minIncreasePerSec and maxIncreasePerSec parameters. This means that every second, the AI will increase its Hunger level by this amount, randomly picked from the min/max range. So if you set 5 to 15, then every second the AI will increase the Value of "Hunger" by anywhere from 5 to 15. The higher you set this, the faster the AI will become hungry. For wildlife animals, if you're simulating realism, you might want to set this to a small number like 0.3 to 2. Otherwise, the animal will find food every few minutes, which isn't realistic.

c) Now, we'll define WHEN to tell if the AI is actually hungry. Why? Remember, this is being done on the Emotion Controller. Everyone has different levels of FEELING hungry. Some people feel hungry even when they're not. Others do NOT feel hungry, even when their bodies are out of food. So, tell the AI when it should FEEL hungry by opening up the "Need Conditions" section and adding a new slot. Set the minVal to 15 and maxVal to 100. That means when the AI's hunger level is between 15 and 100, it will FEEL hungry. So when its below 15, it will NOT feel hungry.

d) Now, add a flag to activate when it feels hungry, such as "HUNGER". This flag will be used by the Goal Controller (we'll get to that in a bit), by the Brain actions, and by external scripts. Remember this flag name.

You're now done setting up the Needs portion.

3) Add the Flag name to the Brain's flag list.

4) Now we tell the AI what to do WHEN it becomes hungry. If you have your own implementation / code, you can just ring the

Flag system from your scripts to check if the flag "HUNGRY" is active. If you're not using your own code, you can use the Goal Controller.

1) Add the goal controller to your AI:

Components Menu → AIDesigner → Controllers → Goal Controller

2) Create a new goal, name it something like "Find food"

3) Click "Repeating Goal" because we want the AI to find food again the next time it becomes hungry.

4) Set the priority – if you have other goals, the one with the highest priority will be processed first

5) Set the Goal Type to "Find" because we're going to find food.

6) Set the Dist to Target – this is simply the distance to the goal object at which the AI will mark the goal as "Completed"

7) Now, open up "Goal Target" and type "Food" into the Tag slot. We're telling the AI to search for all objects tagged as "Food". - This uses the Unity Tag system. You can use any tag you'd like, like "Water", "Health Potion", "Gold Coins", etc. Just make sure the tag is added to your project and the target objects are tagged.

8) Tell the AI what to do upon moving to the food. Obviously, he should eat it. But that also means his Hunger level should DECREASE. So open up "On Target Reached". Open "Add to Needs" and create a new slot. For Var Name, type "Hunger" (same var name of your Needs / Emotion Controller). For Add Value, type a NEGATIVE number. This is the amount to add to hunger levels, so if you want to DECREASE hunger, add a NEGATIVE number. So type in -150. That means when the AI reaches the food, it will decrease 150 from his hunger level and making him no longer hungry.

9) This goal action – searching for objects tagged "Food" - isn't going to play on its own. It needs something to trigger it. So open up Goal Triggers.   Under "This Flag Active", type in "HUNGRY". This is the flag name you selected previously. Basically, when the flag "HUNGRY" is activated, this goal will activate as well, triggering the AI to search for objects tagged as "Food"

That's it for Goals and Needs.

One final thing you might want to do is go back to your Brain and review all of your Action groups. You now have an additional condition (HUNGRY) to consider.

For example, if you previously had the AI wander or patrol waypoints, you need to add a new condition to that to make it stop doing the action if the HUNGRY flag is activated.

There are many examples of this in the Demo Scenes folder. See the various goals and actions demos and the Agent setups in those scenes.

28 | EMOTIONS & FEELINGS

{ PRO ONLY FEATURE }

To enable Emotions and Needs: Select your AI character, go to:

Component Menu → AI Designer Pro → Controllers → Emotion

The Emotion Controller handles:

1) Emotions & Feelings management for the AI it's attached to.

2) Needs management (any need you define, such as Thirst, Hunger, etc).

3) Real time, live updating of feelings for nearby actors (live-sight feature)

(e.g. if AI hates a nearby actor, it's self-feeling of "friendliness" will decrease)

The Emotion Controller can be used by:

1) Dialogue Controller for emotion-triggered dialogue **(see demos)**

2) Goals Controller for Needs-driven goals and actions **(see demos)**

3) Any of your custom code **(see Emotion dialogue demo)**

Available Emotions & Feelings:

1) Friendliness

    -100 to 0 = Not friendly to hate

    0 = Neutral

    0 to +100 = Friendly to love

2) Happiness

    -100 to 0 = Unhappy to Very Sad (or Angry, depending on your use case)

    0 = Neutral

    0 to +100 = Happy to Elated

3) Fear

    - 100 to 0 = Not afraid to no fear

    0 = neutral

    0 to +100 = Afraid to extremely fearful

Coding / No-Coding Operations:

Coding Required: The Emotions/Feelings management only manages the AI's current feelings and requires C# coding on your part to fully integrate and use in your projects. It gives you a simple API to increase/decrease Friendliness/Happiness/Fear during gameplay. You can then query the Emotion Controller for the AI's current feelings to support the rest of your game code and logic.

Coding Not Required: You can use the Dialogue Controller to set min/max feeling values (from -100 to +100) that will trigger specific dialogue (see emotion dialogue demo).

You can also use the Needs portion of the Emotion Controller to set Goals and actions without any code (see Goals and Actions demos related to Thirst and Hunger)

Memory

The AI can store memories of the actors it meets that includes data on its happiness, friendliness, and fear. This is feelings the AI has towards the Actor, and not the feelings of the Actor itself. This memory is used by both the Dialogue Controller and any custom code you have. In future updates, we're thinking to add a few more examples or expand this user manual on this memory subject. We might also add a few more features to make the AI exhibit more human-like memory elements.

Emotion Realtime Updates / Live-Sight Feature

The Emotion Controller includes a built-in real time update feature. This allows the AI to constantly update its feelings based on nearby actors. For example, if it hates nearby actor, it will update its "Self-Happiness" and "Self-Friendliness" to negative values, based on the level of hatred stored in its memory.

Note: The "Live-Sight" feature (real time feeling updates) can be CPU intensive, and is only recommended to be enabled for a few companion characters that follow the player around, or a few key quest characters. You should not enable this on every NPC in a town.

29 | THROWING & GRENADES

{ PRO ONLY FEATURE }

Setting up throwing behavior is pretty simple.

1) Attach the Throw Controller to your AI.
Components Menu → AI Designer Pro → Controllers → Throw Controllers

2) Add a "Throw position" blank gameobject to your AI. Drag it a bit far away from the AI body / collider so that the object you throw don't collide and bounce off the AI himself.

3) Edit the Throw Controller's settings, insert Throw position into the throw position slot, and insert the prefab for object to throw. It's advised that your object has a collider and Rigidbody with Gravity enabled (or your object won't drop back down after thrown).

4) Edit the Forward/Up Forces – these are physics-driven forces based on Unity physics.

5) Now, you have to tell the AI when to activate the Throw Controller, so go into the Brain and add a new Action. Name it something like "Throw" and go down to "Throwing" action. Check the "Do Throw" action. In the conditions, you can set when to activate, such as when target is within 5-20 range. **Example: See the Arena shootout demo and click the main godmode AI – this bot has grenade abilities.**

When the Brain activates the Throw controller, it will start a timer process that will throw an object every amount of seconds (you set the min/max range and it will pick a random time).   When the Brain does not have the set of conditions you set for throwing (e.g. enemy is not within that range you set), the Throw Controller is automatically deactivated.

30 | VEHICLES

Vehicle behavior is made for both cars and aerial jet-like units.

Setting up vehicles is easy and similar to all other AI setups. Drag any vehicle model into your scene, select it, go to the Tools → AI Designer menu, and click "Create Vehicle" from the Templates.

What this does is turn your vehicle model into a vehicle-AI enabled GameObject with all the necessary settings, minus animations.

Required Vehicle Setting: Vehicle behavior requires the following setting on the Brain (this is already done for you if you used the Template menu)
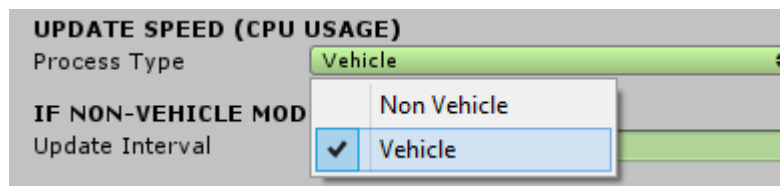
Why? Because vehicles are typically faster than usual agents, require smoother turns and less stopping. This requires vehicle agents to use special processing by the AI Designer engine. This translates to heavier CPU usage than non-vehicle AI, so try not to have too many vehicles in one scene.

Another thing to consider is to disable both Movement Noise and Obstacle Avoidance/Collisions on vehicles to make operations less CPU intensive.

Wheels: You can set up wheels to automatically spin, based on speed (the faster the speed, the faster they spin). This is set in the Movement Controller. Scroll down to the Wheels section. Expand the wheel slots and drag each wheel into it. The wheels must be detached from the car model and must be able to be separately rotated. See the racing demo for a demo (when you click each car, you can see that 4 wheels are set up).

Setting up wildlife is easy and similar to all other AI setups. You can view our Youtube video on our Asset Store page on how to set this up.

Basically, drag any animal model into your scene, select it, go to the Tools → AI Designer menu, and click "Create Wildlife" from the Templates.

"Basic Wildlife" template: Creates a basic agent setup that can wander and flee when attacked.

"Advanced Wildlife" template: Creates an advanced agent setup that can wander and has Goals and Needs. It's pre-setup for you with Hunger and Thirst needs, and to automatically look for food (tagged as food) and water (tagged as water) objects when any need is unfulfilled.

32 | STANCE MANAGER

The Stance Manager lets your AI switch between unlimited stances that you define. It automatically adjusts colliders (e.g., making collider smaller if crouching), sets new movement speed (e.g., move at slow speed if crouching), and enables Animator Bools. If your AI needs a Crouch or Prone stance, you would use this.

For most cases, you do not need this. Jumping, attacking, and most movement animations are usually not handled by this unless you want the AI to move while crouching.

Usage:

1) Select your AI and add the Stance Manager:
Components Menu → AI Designer → Controllers → Stance Manager

2) Add a new stance. Name it something like "crouch stance"

3) If you want colliders to adjust smaller when crouching, set the new collider stats here and enable "Change Collider Setting"

4) If you want to update your Mechanim Animator's bool value, enter a bool name. This bool will be **enabled** when this stance is played. For example, if you want the Animator to go from Stand-Walk to Crouch-Walk, set a
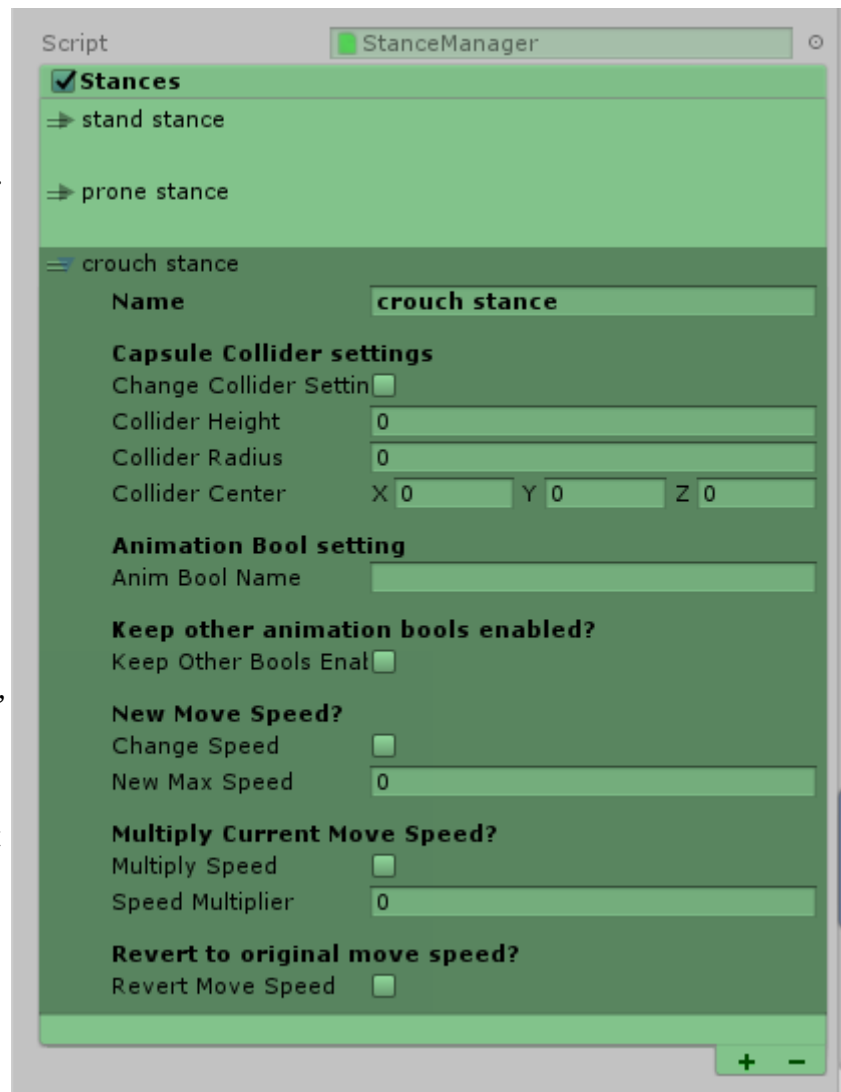
transition that checks if a bool "isCrouching" is true.

5) If you want the AI to move at a different speed (e.g., slower when crouched), enter a new speed and enable "Change Speed".

6) If you're modifying an original stance, like a "Standing" stance, you might want to revert speed back to the original, so enable "Revert Move Speed."

7) You're done setting up the Stance Manager. Now we have to tell the AI when to play this particular stance. So go into the Brain and add a new action. Call it something like "Crouch 20% chance"

8) Go down to Probability settings and type in 20. We want the AI to crouch 20% of the time.

9) Go to Actions, and find "Stance". Input the name of the Stance you just created (our example it's "crouch stance" and enable Stance.

10) That's it. You probably want the AI to go back to Standing after a while. In that case, go back to your Stance Manager and create a new stance called "Stand". Then, go back to the Brain and add an action called "Stand 70% chance" and do the same as you did for the crouch action.

You can create more complex behavior by inputting further conditions (e.g., when target is within 5-20 range) to trigger the crouch stance.

33 | THE SPECIAL PROTOTYPING CAMERA

AI Designer ships with a special camera system for quick prototyping and testing AI behavior, without having to spend time to add your own implementation into the scene.

It allows you to quickly switch between camera views during gameplay (press 1, 2, 3).

Select your scene's main camera.
Click [Add Component] from the bottom of the screen. Type in "prototype cam" and select "Prototype Camera".

Usage: Drag an AI character from your scene into the "Target" slot. It's recommended you set the AI to "GodMode" (Health component) so you can spend more time watching it in case it gets killed.

It supports the following views:

AI POV Vision: The camera will automatically lock onto the AI's Awareness component and lets you see what the AI sees, in first person mode.

Follow Target: The camera will follow the AI in third person mode.

Spectator Target: The camera will watch the AI like a security cam, but will not follow it. Camera stays in same starting position.
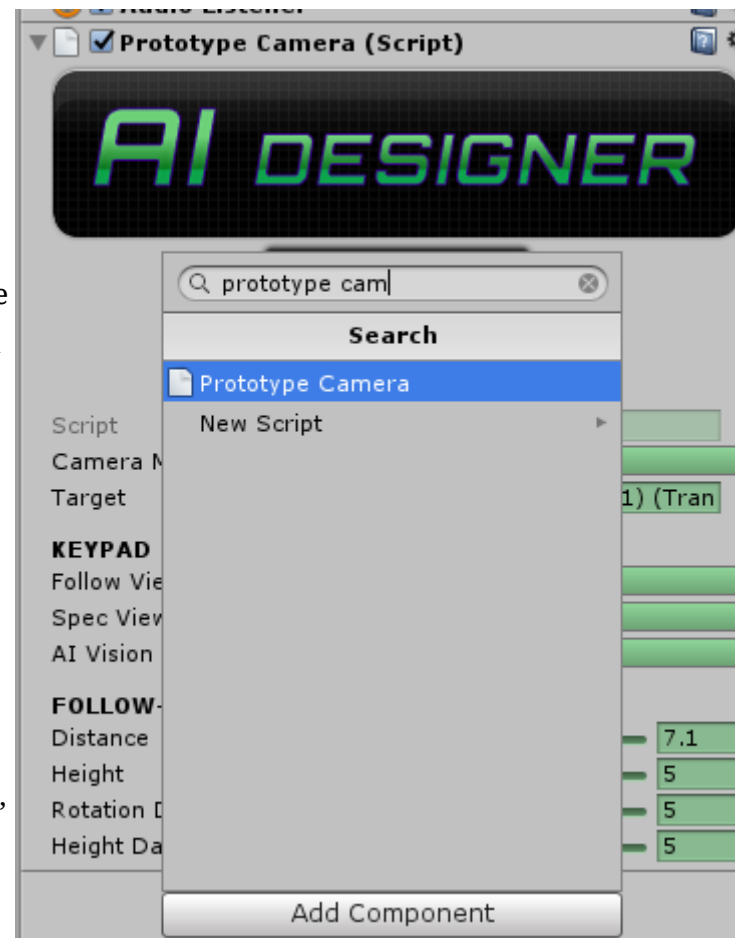
Free Look: You can use the mouse to look around. Camera stays at its starting position.

34 | RAGDOLLS

The new version 0.95+ update includes a new component: The Ragdoll Animator. This component includes enhancements for ragdoll management and bug fixes and takes care of things in the background without much effort on your part. To see how it works, check out the demo scene: "Ragdoll Demo" in folder "Demos_Ragdoll Death Effects".

To enable ragdoll effects on your character, do the following:

1) Setup AI as usual.

2) Click on your AI (parent gameobject) and setup Ragdoll using Unity's own tools.
GameObject Menu → 3D Object → Ragdoll

3) Click on your AI again, then add the new Ragdoll Animator:
Component Menu → AI Designer → Procedural Animations → Ragdoll Animator

4) Set "Start Mode" to Ragdoll Off. If you set it to Ragdoll On, it will immediately activate the Ragdoll on start. Only set this to Ragdoll On if you have custom scripts that will handle ragdoll procedural motions. Otherwise, leave it Off.

5) Now, go to Effects Builder. If you don't have a "Death" effect, add one now.

Disable any animation actions if any exist.

Under "Ragdoll Enable", check this:
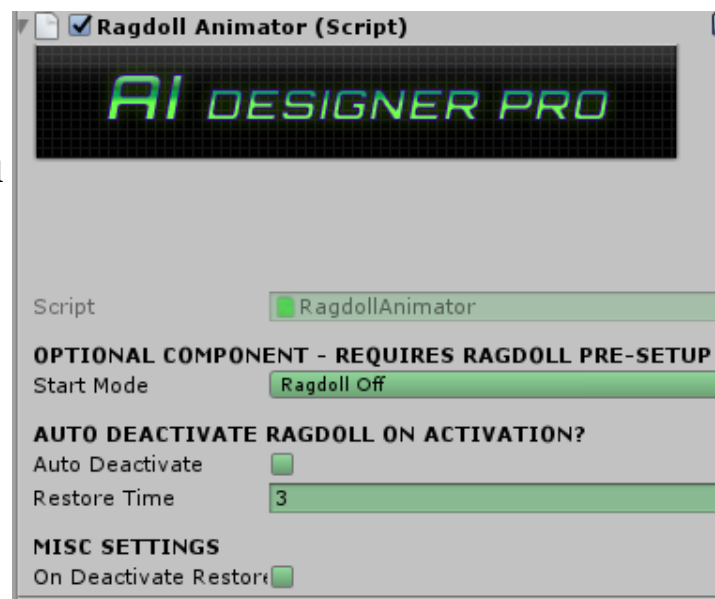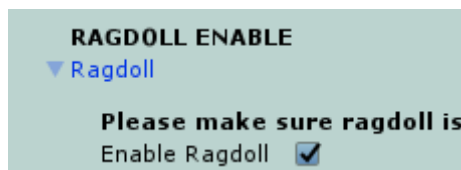
Like standard setups, add this Effects Builder effect name into the Health controller:

{ EFFECTS-BUILDER INTEGRATION }
Effect On Death   Death
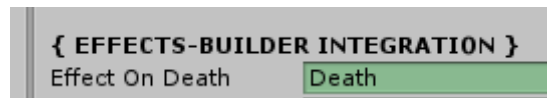
Now, when the AI is killed, instead of playing an animation, it will activate its Ragdoll and generate a procedural effect.

OTHER RAGDOLL CONTROLLER SETTINGS

Auto Deactivate: This auto shuts off Ragdoll (restoring Mechanim Animation Controller) after a few seconds when the Ragdoll is activated.

Restore Time: When Ragdoll is activate, wait this amount of seconds before deactivating Ragdoll.

Misc Settings --

On Deactivate Restore Position:
When ragdoll is activated and then deactivated, restore the character's position to its original state *before* ragdoll activation.

Ragdoll Animator (Script)

AI DESIGNER PRO

Script   RagdollAnimator
OPTIONAL COMPONENT - REQUIRES RAGDOLL PRE-SETUP
Start Mode   Ragdoll Off
AUTO DEACTIVATE RAGDOLL ON ACTIVATION?
Auto Deactivate
Restore Time   3
MISC SETTINGS
On Deactivate Restore
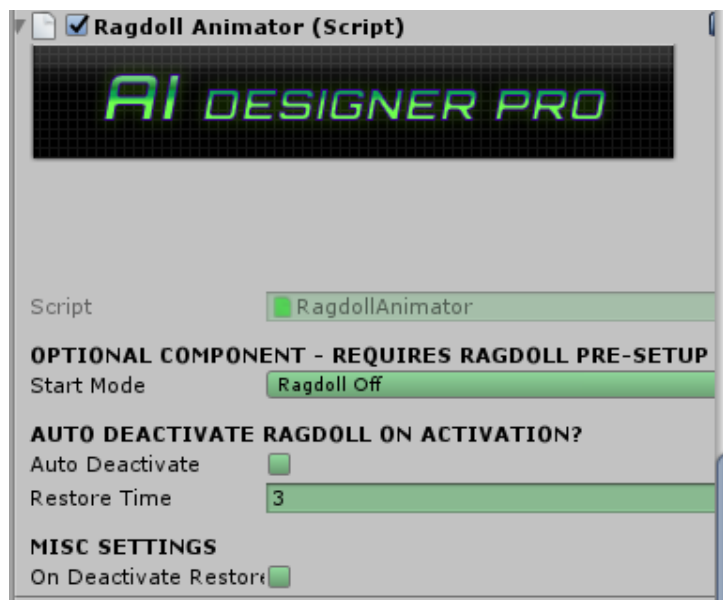
35 | DEBUGGING / TESTING TOOLS

AI Designer ships with some useful debugging tools:

Show Active Action State Labels:

On every Brain component resides a checkbox 'Show Active Labels'

Click this box to print a useful list of current active actions at the top of the AI's head. When your main camera comes within 40 distance, the labels will show up. You can use this to check what the AI is currently doing.

Print to Console for each Action:

On the Brain component, each action group has an option to print a custom text into the Unity Console:

Here, when this action group is triggered, it will print "enemy running away!" in the Unity console. If it's printing too much and clogging up CPU, click "Print Once Only"

While AI Designer is a complex system, its Third-party integration is actually quite straightforward. You can send damage to AI bots with one line of code or simply attach the included Damage On Impact component to your existing bullet prefabs. Additionally, many components has a slot called "Custom Function Name" - enter the name of your script's function and it will be called using Unity SendMessage().

HOW MUCH KNOWLEDGE / EXPERIENCE IS NEEDED?

If you're not a coder, you need an understanding of basic Unity non-programming concepts, how and when to add certain components, how the Mechanim animation system works, colliders, and so on.

If you're a coder and intend to code most of your game, you need a basic understanding of how Unity's own API works – this is a requirement of any product you buy that you intend to integrate with scripting. You need to know about Unity's SendMessage implementations since most important controllers have a SendMessage function. You need to have a beginner's knowledge of C# (we **do not** recommend Javascript beyond prototyping, for long term game development). This documentation contains information regarding AI Designer's public methods and variables that you can integrate with your own scripts.

UNITY'S SEND-MESSAGE IMPLEMENTATION:

Unity allows multiple assets/scripts to communicate using the SendMessage system. AI Designer has many components that fire events, such as onMove or onDeath. You insert your attached function names into these slots. Whenever you see a slot called "Custom Function" you can insert it.

For example, the Movement Controller has the following events where you can insert your function names:

Integrating Movement:
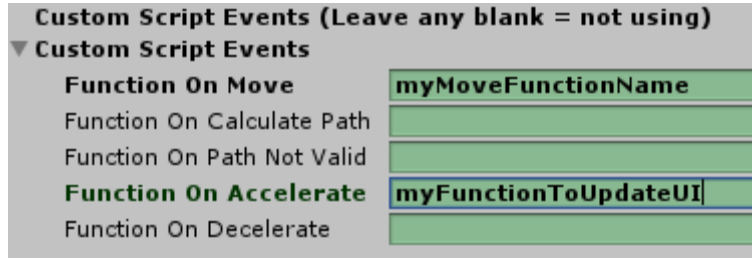
Integrating Projectile Damage: On the "Damage On Impact" component that is attached to every bullet, there's a slot to insert your custom damage function:
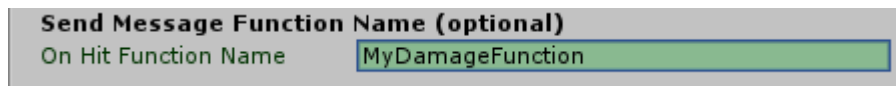
Here, when the bullet hits something, it will use Unity SendMessage() to call the function name "MyDamageFunction" on the VICTIM it hits, meaning, the Victim object must have a script that has a function named "MyDamageFunction".

To receive the damage amount, your function should be built like this:

void MyDamageFunction(float amount){
        // process 'amount' ... do something here...

}

Integrating Hit-Scan Damage: On the Ranged Attack ADV controller, there is a slot under Hit-Scan settings where you can set custom function name: (this works the same as above – it's called using Unity SendMessage() system.
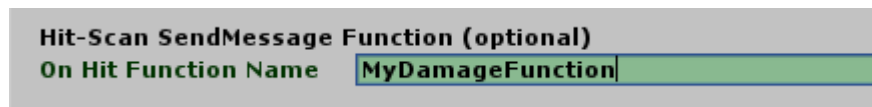
When the AI shoots and hits something, it will call MyDamageFunction on that object with damage amount, same as above.

Integrating Melee Damage: On the "Melee Weapon / Melee Send Damage" component attached to the weapon itself, there's a slot to insert custom function name:
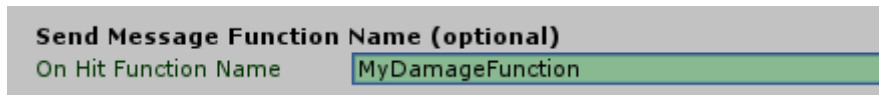
Each time the weapon swings and hits an object, it will call MyDamageFunction() on that object with damage amount, same as above.

HOW TO INTEGRATE WITH MY OWN PLAYER CONTROLLER?

Your most important question is most likely: How to make the AI attack my player? How to make my player damage the AI?

To make the AI attack the human player, or *any* object in your scene, that object needs these 2 things attached:

1) A collider (not a trigger) on the top parent Game Object

2) The "Identity" component from AI Designer. Simply select your player prefab, go to Component Menu → AI Designer → Identity

Then, Set the faction on the Identity component. So if your AI targets the "Team1" faction, set it to Team1.

SEND DAMAGE TO AI WITH CUSTOM RANGED ATTACKS

To allow your player to damage AI, you should already have a shooting script on your player prefab.

If you're shooting projectiles and not using hit-scan, meaning, your player's bullets literally have to fly across the screen to hit something, then all you have to do is attach the "Damage On Impact" component to your bullet.
Simply select your bullet, go to:

Component Menu → AI Designer → Damage Behaviors → Send Damage on Impact

Make sure your bullet has a sphere collider set to trigger. Now whenever your bullets hit an AI, it will cause damage. This allows your own scripts and AI Designer damage senders to exist on the same bullet object.

But what if your weapon system uses hit-scan (raycast hits), instead of projectiles? Easy. Go into your shooter script and find your hit-scan function. When your hit-scan hits an object, check if that object has the attached Health component like this:

```
if (victimGameObject.GetComponent<AIDesigner.Health>() != null){

        // this object has a health component... do your stuff here...

}
```

Then, you need to use AI Designer's damage API: TakeDamageWithData(float damage, GameObject attacker)

So, you'd do this:

```
if (victimGameObject.GetComponent<AIDesigner.Health>() != null){

        victimGameObject.GetComponent<AIDesigner.Health>().TakeDamageWithData(50, me);

}
```

SEND DAMAGE WITH CUSTOM MELEE ATTACKS

If your human player uses Melee attacks, there are multiple ways to damage AI. Most likely you're using an implementation similar to ours – the character plays a Melee-attack animation and the sword swings towards a target. When the sword hits (using a collider), it detects the object that was hit. In this case, you need to modify your script to do the following:

1) When the sword's collider hits an object, check if the Game Object has a Health component attached:

```
if (victimGameObject.GetComponent<AIDesigner.Health>() != null){

        // this object has a health component... do your stuff here...

}
```

Then, you need to use AI Designer's damage API: TakeDamageWithData(float damage, GameObject attacker)

So, you'd do this:

```
if (victimGameObject.GetComponent<AIDesigner.Health>() != null){

        victimGameObject.GetComponent<AIDesigner.Health>().TakeDamageWithData(50, me);

}
```


Here, 'me' refers to the GameObject of your human player – the one sending the damage

The reason you're sending over the gameobject of the attacker is so the AI knows who just attacked him.

37 | API REFERENCE

INTRO

This section is for programmers who want to integrate AI Designer with custom code. Remember, for most functions, you do not need to resort to coding since AI Designer takes care of many things for you under the hood. If you need to do something advanced, first, think if you can already do it with AI Designer's tools. If not, then look up this list of global functions.

Rules:
- API was created assuming you have beginner knowledge of C#
- All public methods use standard Unity GetComponent for access
- All public method names capitalize each word like .FunctionName instead of .functionName
- If you understand basic C# and how to use GetComponent, you're good to go.

OBJECT POOL

The included Object Pool system is fast and modular and can be used in projects that does not require AI Designer characters. AI Designer automatically integrates its capabilities with the Object Pool without coding on your part. If you wish to use it by itself, simply use the API (just add and setup Object Pool set up in scene):

Usage: AIDesigner.ObjectPool.FunctionName()
Recommended usage:

```
AIDesigner.ObjectPool objectPool;
objectPool = (AIDesigner.ObjectPool) FindObjectOfType(typeof(AIDesigner.ObjectPool));
GameObject bullet = objectPool.GetObject(GameObject myObject);
```

GetObject(GameObject myObject)

Returns a GameObject from the pool of the requested object type. This object must be set up in the Object Pool in your scene first.

ReturnObject(GameObject myObject)

Use this in place of Unity's Destroy. When you want to remove something from the scene, such as a bullet that has hit its target, simply call objectPool.ReturnObject(bulletObject). You don't need to do anything else. This function automatically puts the object back into the pool and hides it from your players. It will still be in the scene though, and is not destroyed in memory. You can retrieve it back using GetObject(object Type). This method is usually faster than destroying the object from scene / memory and minimizes lag.

HEALTH COMPONENT

Usage: gameObject.GetComponent<AIDesigner.Health>().FunctionName()

TakeDamageWithData(float damage, GameObject attacker/damage sender)

Sends damage to this attached Health component. Armor and Shields, if any, are automatically deducted from the damage you send. The minimum damage is 0.

The second argument, GameObject attacker, is the gameobject of the object sending the damage. This is useful for deathmatch games where you're keeping score of who killed who.

SetNewArmor(float amount)

Sets a new armor amount. Useful if your character uses different kinds of equipment / armor.

SetNewHealth(float amount)

Fully heals character and sets health to amount. If this character is already dead, then this revives him. This is useful in an RPG game where a character "levels up" and gains new health.

SetNewShield(float amount)

Fully recharges shields and sets shield to new amount. Useful if your character equips a stronger shield device.

Revive()

Fully heals Health and Shield, and if character is dead, revives him.

HealHealth(float amount)

Heals the character for amount. Will never heal beyond 100%.

HealShield(float amount)

Recharges shield for amount. Never goes beyond 100%.

SetNewHealthRegen(float amount)

Sets new health regeneration rate

GetHealthPercentage()

RETURNS FLOAT: a number between 0-100 of the remaining health %

GetHealth()

RETURNS FLOAT: current remaining health (not percentage)

AmITakingDamage()

RETURNS BOOL: is the character currently taking damage?

GetLastAttacker()

RETURNS GAMEOBJECT: the object of last known attacker. Useful for keeping score of who killed who

Kill()

Immediately kills the AI character.

AmIAlive()

RETURNS BOOL: is the AI currently alive?

BRAIN MODULE & FLAG SYSTEM

Usage: gameObject.GetComponent<AIDesigner.Brain>().FunctionName()

GetFlag(string flagName)
RETURNS BOOL: Is this flag active? If flag does not exist, or if you spelled incorrectly (case sensitive), then it will always return False.

SetFlag(string flagName, bool trueFalse)
Set to true or false to activate / deactivate the flag

GetGlobalFlag(string globalFlagName)
RETURNS BOOL: Is this global flag active?

SetGlobalFlag(string globalFlagName, bool trueFalse)
Set true/false to Activate / deactivate flag

EnableConditionGroup(string actionName)
Enables an action group on the Brain component. When enabled, the brain will check the action group's conditions to decide when to trigger the associated actions.

DisableConditionGroup(string actionName)
Disables an action group on brain component. If disabled, the brain will stop checking the conditions in that group. So if you want the AI to stop Chasing targets and your action group name is "Chase enemies" then use DisableConditionGroup("Chase enemies")

EnableAllConditionGroups()
Like EnableConditionGroup except it enables everything

DisableAllConditionGroups()
Like DisableConditionGroup except it disables everything. This essentially pauses your AI's brain module and in most cases, your AI will stop working until you enable something again. Ranged attacks, animations and environment awareness scanning will still run, though.

SetConditionGroupMoveToObject(string actionGroupName, GameObject targetObject)
In the MoveTo Actions section, there is this handy behavior:
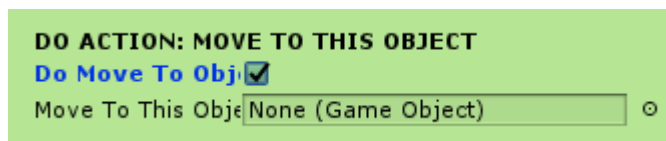
This function basically sets the target object for that behavior, at runtime, on the action group name you specify.
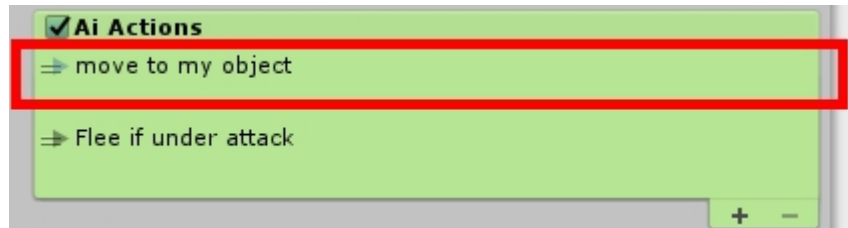
The first parameter, "ActionGroupName" is the name of the action group in the AI's brain component:

So in this case your function call would be:

SetConditionGroupMoveToObject("move to my object", targetObject)

SetConditionGroupMoveToVector3(string actionGroupName, Vector3 destination):

Similar to the above function, this one sets a move-to paramter for a specific action group. In this case, it sets the Vector3 destination of:
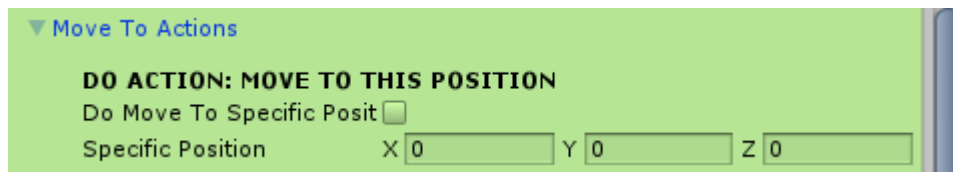
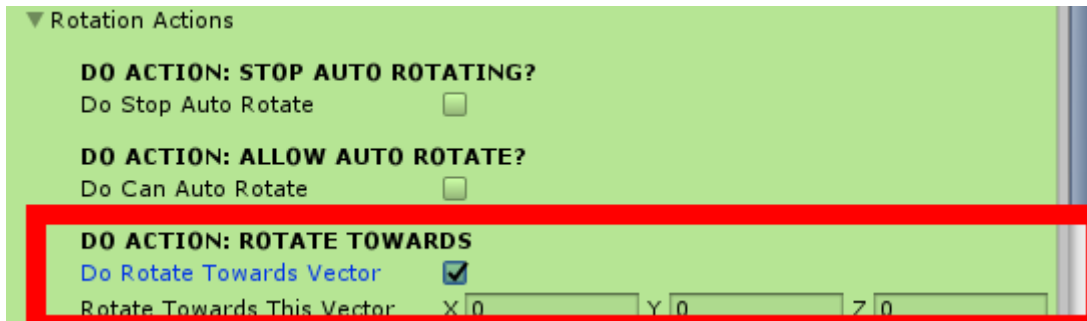So, say you have an action group named "Move to HQ":
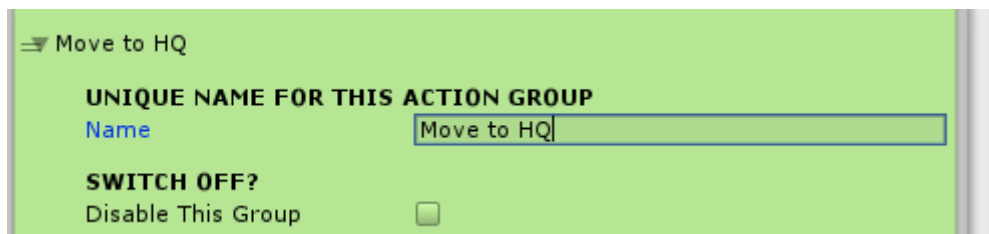
Your function call would be:
SetConditionGroupMoveToVector3("Move to HQ", new Vector3 (150, 5, 500));

SetConditionGroupRotateToVector3(string actionGroupName, Vector3 rotateToDestination):

Similar to above, this function sets the rotation to rotate towards for the following action:

Make sure "DoRotateTowardsVector" is checked prior to using this function.

This will enable Custom Rotations on the Movement Controller. This means that when this is enabled, auto rotations will not fully finish (e.g., rotating towards a self-selected target).

Freeze()

Freezes the entire AI character – all AI functions will be paused, including animations and attacks. Basically, it will look like the character is frozen in time, especially if his animation was in the middle of a pose.

This has a lot of creative uses, from making your own debugging system (freeze a bot during gameplay for testing) or simulating special magic abilities that freeze a character.

Unfreeze()

Unfreezes the character – resume all operations and animations.

RANGED ATTACKS & SWAPPING WEAPONS AT RUNTIME

Usage: gameObject.GetComponent<AIDesigner.RangedAttackBasic>().varName()

or (pro version)

Usage: gameObject.GetComponent<AIDesigner.RangedAttackAdvanced>().varName()

The following are all global variables that can be changed at runtime. They're useful if you're implementing your own weapon / inventory system and want to swap out bullets on the AI.

float timeBetweenShots

The amount of seconds to wait before firing another shot. Set this to a low number if switching to a machine-gun type weapon, and set it to a high number for slower sniper-style weapons.

float inaccuracyMod

A number added to the AI's targeting that on-purposely throws off his aim, to simulate different accuracy levels

gameobject bulletToFire

The gameobject reference to the bullet type to fire. If switching weapons at runtime, you can simply change this to another bullet type.

Gameobject muzzleFlash

A particle object to spawn when shots are fired. Can be changed at runtime according to weapon type.

RANGED ATTACK PRO-ONLY FUNCTIONS:

Usage: gameObject.GetComponent<AIDesigner.RangedAttackAdvanced>().varName()

These functions are available for Ranged Attack Advanced:

SwitchWeaponToHitscan()

Changes weapon type to hitscan

SwitchWeaponToProjectile()

Changes weapon type to projectile-based

float hitscanDistance: distance to fire hit-scan ray

float hitscanMinDamage, float hitscanMaxDamage

When the AI fires a shot using hit-scan, it will calculate a random damage between min and max values. To keep constant value, make them the same number.

JUMP CONTROLLER

Usage: gameObject.GetComponent<AIDesigner.JumpController>().functionName()

Jump()
Forces AI to jump, according to the jump settings you specifed in the Editor.

RAGDOLL ANIMATOR

This controller is new in v0.95+ and handles Ragdoll effects, doing most of the grunt work for you. Your character must already have a ragdoll set up (see Unity documentation).

Usage: gameObject.GetComponent<AIDesigner.RagdollAnimator>().functionName();

Activate()
Enables Ragdoll effects. This will block all animations on your character.

Restore()
Disables Ragdoll effects. This will enable animations on your character as well as give back control. This works in conjunction with position restoration (See the Ragdoll chapter).

TARGET CONTROLLER

Usage: gameObject.GetComponent<AIDesigner.TargetController>().functionName();

SetCustomTarget(GameObject target)
Commands AI to target specific game object. The AI will engage this target according to the settings you set in Brain, Awareness, Target Controller, and Melee / Ranged attack controllers.

RestoreTargetStrategyToOriginal()
You usually call this *after* you call SetCustomTarget() and want the AI to stop targeting your specific target and return to its original programming (engage self-selected targets on its own).

WANDER CONTRLLER

Usage: gameObject.GetComponent<AIDesigner.WanderController>().functionName()

GetWanderDestination()

Returns Vector3 position of the AI's current selected wander destination.

ChangePaths()

Forces the AI to select a new random Wander destination.

Pause()

Removes permission to wander.

Resume()

Gives back permission to wander.

EMOTION CONTROLLER

Usage: gameObject.GetComponent<AIDesigner.EmotionController>().functionName()
int TypeOfName is always: 0 (char name), or 1 (faction name), or   2 (gameobject TAG)

CreateNewMemoryEntry(string nameOfObject/Faction/Tag, int typeOfName)

Creates a new entry in the AI's memory for future storage, regarding this item.

What is this item? The item can be:
- An Actor (based on Char Name in its attached Identity component)
- A Faction tag (based on Faction in its attached Identity component)
- A Unity tag (based on GameObject tag)

Why is this useful? In order for the AI to keep a record of how it "feels" towards a particular actor, faction, or tagged object, it first needs an entry created for it in its memory.
Returns nothing.

DoesMemoryExist(string nameOfObject/Faction/Tag, int typeOfName)

Checks if an entry for the charname/faction name/tag name exists in AI memory.

Returns true/false.

GetFriendliness(string nameOfObject/Faction/Tag, int typeOfName)

Get's the current "Friendly" score that the AI feels towards a particular object/faction/tag.

Returns float from -100 (most hostile) to +100 (most friendly)

GetHappiness(string nameOfObject/Faction/Tag, int typeOfName)

Get's the current "Happy" score that the AI feels towards a particular object/faction/tag.

Returns float from -100 (most angry/sad) to +100 (most happy)

GetFear(string nameOfObject/Faction/Tag, int typeOfName)

Get's the current "Fear" score that the AI feels towards a particular object/faction/tag.

Returns float from -100 (most NOT afraid) to +100 (most afraid/fearful)

IncreaseFriend(string nameOfObject/Faction/Tag, float amount, int typeOfName)

Increases "Friendly" score to a particular charName, faction name, or tag name.

A negative amount means a DECREASE in score.

Returns nothing.

IncreaseHappy(string nameOfObject/Faction/Tag, float amount, int typeOfName)

Increases "Happy" score to a particular charName, faction name, or tag name.

A negative amount means a DECREASE in score.

Returns nothing.

IncreaseFear(string nameOfObject/Faction/Tag, float amount, int typeOfName)

Increases "Fear" score to a particular charName, faction name, or tag name.

A negative amount means a DECREASE in fear, positive means INCREASE in fear.

Returns nothing.

IncreaseNeed(string nameOfNeed, float amountToAdd)

Increases "Need" in a specific need. So if you have a Need called "Thirst", calling the following function will increase "Thirst" by 50 (the bigger the increase, the "thirstier" it gets):

gameObject.GetComponent<AIDesigner.EmotionController>().IncreaseNeed("Thirst", 50);

Returns nothing.

UTILITY FUNCTIONS

Utility functions exist for your convenience. You call them directly. Nothing is added or set up in your scene. To use them, your script must be attached to the AI or other object that you want these methods to affect.

Usage: AIDesigner.Utils.FunctionName();

IsDestinationInFront(Transform myself, Vector3 targetPosition)
Returns BOOL: True if *targetPosition* is in front of the *myself* Transform; False if behind.

Example usage:

if (AIDesigner.Utils.IsDestinationInFront(transform, enemy.transform.position) == true){

      // enemy is in front of me

}

ReturnNearestGameObject(GameObject[] arrayOfObj, Vector3 myPosition)
Returns GameObject that is nearest myPosition, given an array of GameObjects

ScaleVector3(Vector3 targetVector, float scaleAmount)
Returns Vector3 that is scaled by scaleAmount from targetVector

THERE ARE MANY MORE FUNCTIONS TO USE!

THIS SECTION IS STILL BEING UPDATED!

38 | SUPPORT & UPDATES

DOCUMENTATION UPDATES: **www.AIBotSystem.com**

Go to the section for AI DESIGNER PRO – there will be a link to updated documentation.

Most of this user manual is complete, but we plan a few more sections and tutorials since not every single feature has been covered yet (though all features are available in the package)

Website Note: Our domain host has been giving our domain strange issues lately, so if the website is down, refresh it a few times and it should come back up.

If the website is down, send us an email to let us know so we can help you specifically.

Current buyers will receive future free updates, such as updates to this documentation, even after the Beta price increases to full price. You can find video tutorials on our asset store page, or on our website.

**aibotsystem@gmail.com** or **www.AIBotSystem.com**

Please send all bugs / problems to our email.

Technical Support Note: We are happy to provide free support for our products only. We cannot give free tutorials on how to use Unity itself. If your technical support request might require us to send you an updated package, please include your Unity Asset Store Invoice #.

If you're developing a serious project, especially a commercial one, you'll obviously want to know what features are performance-heavy and which are not. Please read this section before venturing further into your development so you can tackle performance problems early on (or eliminate performance issue altogether).

EMIT NOISE: The "Emit Noise" feature of many components, especially in the Movement Controller is very CPU-intensive. **Note: "Noise" in this case does not refer to sound effects.** It refers to the noise data received by AI (e.g., a gun fires and alerts AI in another room). Your game doesn't even need to play any sounds for the AI to receive noise data. This is performance-heavy because every action (a single movement, a footstep, a single gunshot) can emit noise data (uses CPU) and alert every nearby AI, that must receive and process this data (using up more CPU). Think about how much CPU must be used if every AI or player in your scene emitted noise data on every footstep (may be 100s) and every gunshot (may be 100s). If you absolutely must use noise data (e.g., stealth game), try to limit it to the human player only and maybe a few AI companions.   For most cases, disable "Emit Noise" in the Movement Controller.
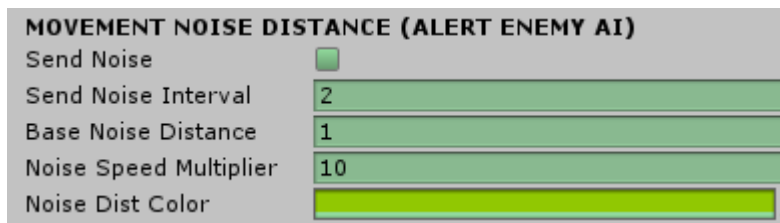
USE OUR BUILT-IN CULLING SYSTEM: If you have large battle scenes, or a large scene where you don't need every AI to be active all the time, use AI Designer's built-in culling system. The culling system allows each AI character to turn itself off, or hide its renderers (graphics) when it gets too far from your Main Camera (a camera in your scene tagged as Main Camera). This let's you place more AI agents in a scene and still have nice FPS. To use

this, pick the AI you want this on, and go to Components → AI Designer Pro → Performance & Culling

SELECT THE APPROPRIATE PROCESS TYPE: Each AI's Brain component has a setting under the AI Actions and Flags sections. This setting is called "Update Inverval". If your AI is non-vehicle (people, animals) then select "Non Vehicle".
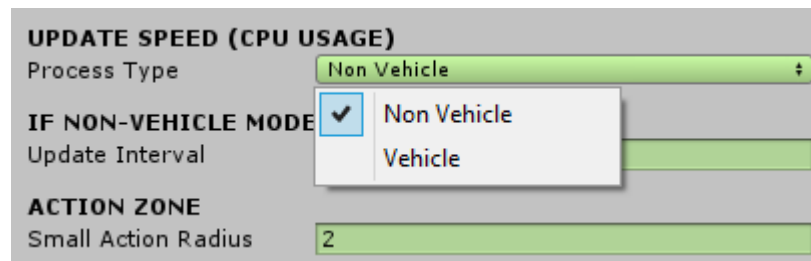
Likewise if your AI is a vehicle (car, jet) select Vehicle. The reason this is important because the "Vehicle" setting uses more CPU power. Vehicle AI requires very fast, real time processing due to the faster vehicle speeds (such as for racing games). Therefore, Vehicle-based AI processes information faster and more often than Non-Vehicle AI. The "Vehicle" setting does not use the "Update Interval" parameter under it.

Note: If using the "Vehicle" setting, probability-based AI actions may not work properly. Please use "Non Vehicle" setting for proper probability-based actions.

For performance-optimized AI that can support larger battle scenes, select "Non Vehicle" and pick a number between 1 and 3 (2 is the default). The higher the number, the **less often** the AI will process information, resulting in less CPU use and game lag. If you set it too high, the AI will be very slow in reacting, making it seem like it's just standing there doing nothing. If you set it too low, the AI will react too often and may exhibit strange behaviors. So unless your game is getting a lot of lag, leave the default setting to 2.

OPTIMIZE THE "RESPONSE TIME DELAY": On the Awareness component, there's a setting at the bottom called "Response Time Delay". This is the delay the AI will take in seconds, before updating its environment data. If your scene has a lot of AI and players, set this to a higher number, like 2-5. Let's say you have 100 NPCs in a large world scene. What would happen if you set the Update Time to a low number like 0.5? Your game may lag because every 0.5 seconds, the AI will need to update it's surrounding environment and process data, potentially for all 100 NPCs. Now obviously, if the AI is only near 10 NPCs, it only needs to process data for 10, but **potentially,** it will need to process 100 simultaneously. Now imagine if all 100 NPCs had to **potentially** process data on all 100 NPCs, every 0.5 seconds. That's a lot of CPU usage! Your human players can be unpredictable, so play on the safe side and set the Update Time to around 2 to 5.

TURN OFF OBSTACLE AVOIDANCE: If your AI uses the "Simple" pathfinding setting on its Movemenet Controller, it has an additional feature available to it: Collision Avoidance. This is set on the Movement Controller.   This tells the AI to try to steer away from an obstacle in its way.   "Simple" pathfinding is very useful, especially in large battle scenes or on open-world terrains and minimizes CPU usage. However, once you enable Obstacle Avoidance, "Simple" pathfinding may lose its performance benefits. If you have lot of AI in a scene, enabling this feature may be CPU intensive due to the fact that every AI must calculate for collisions all the time.
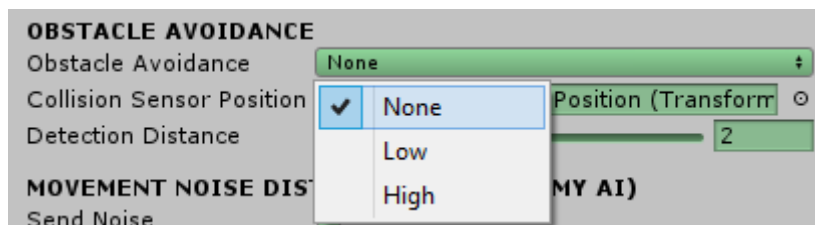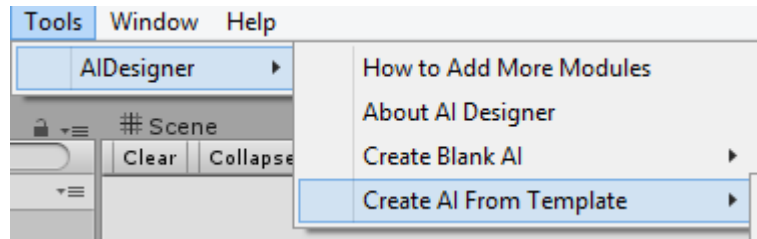

If your scene is very complex with many objects and rooms, we recommend you instead pick "Standard" pathfinding, which requires a baked NavMesh for your scene.
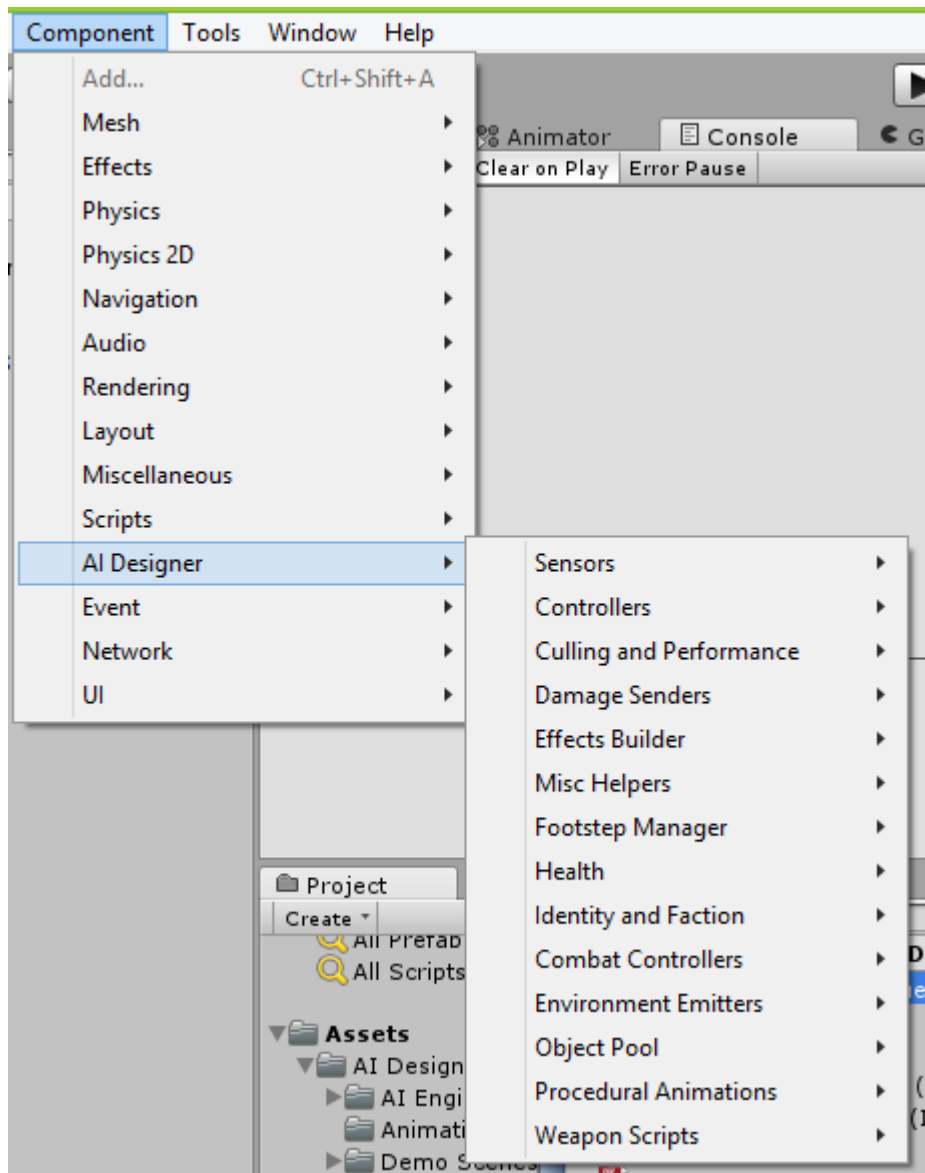
40 | EDITOR TIPS & TRICKS

This is a special section that quickly summarizes how to do certain things quickly. It's recommended to read this section so you don't waste time!
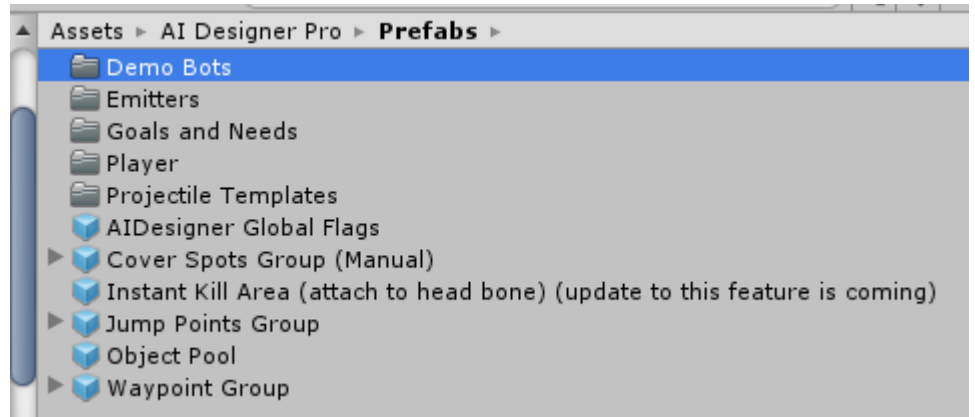
Where to add AI or Templates?

Tools Window Help

AIDesigner ▶

Scene

Clear Collapse

How to Add More Modules

About AI Designer

Create Blank AI ▶

Create AI From Template ▶

(see next page for more)

Where to add AI components?

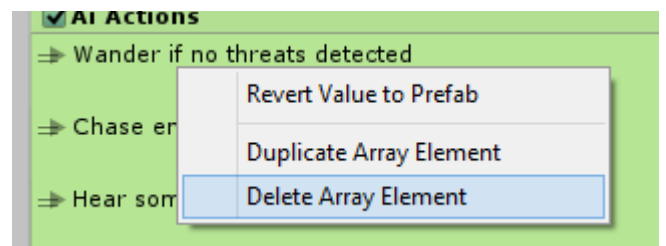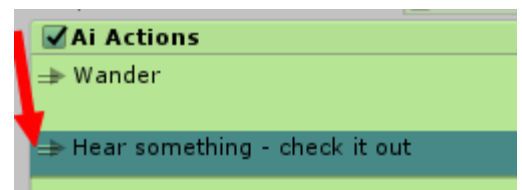## How to Quickly Drop in Bots for Prototyping?

Prefabs / demo bots

How to delete an action in Brain?

2 ways: First, select the action. You can select it by clicking it a bit off to the left side (see image).

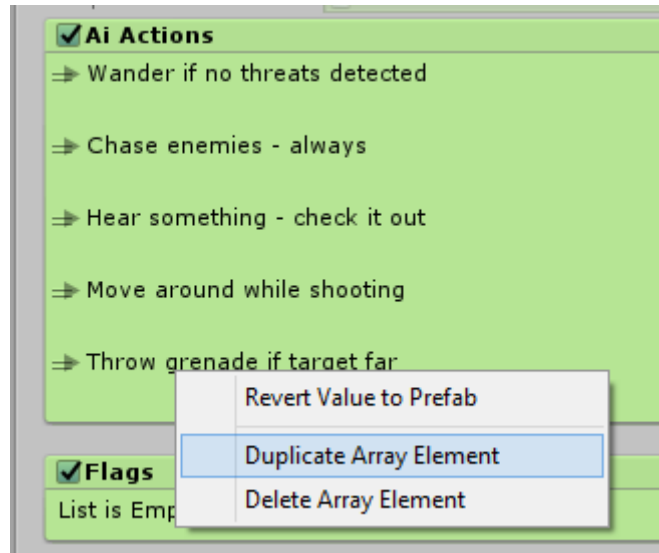Next you can either Right-Click it and click Delete.

Or, click the – minus button on the bottom right.

(go to next page for more)

How to Copy Brain Actions? Copy Flags?

METHOD 1: Right click the action, click
"Duplicate"

METHOD 2:

Click the [+] sign,

and the last action

will automatically

be copied into a new one.

How to Reorder Brain Actions?

How to Show "Debug Labels" on Top of AI?

On the Brain, check this box:

SHOW ACTIVE ACTIONS IN GAME? (DEBUG TOOL)
Show Active Labels ☑

Then when you play in game, the AI will show Brain actions on top of it's head like this:

Note: Some actions that run on probability will STILL show up, even if it's not supposed to show up (the random roll was off)

Jump Away
Approach Target
Goto Stance 2
Goto Stance 1

How to Transfer Actions/Settings to a New Bot?
(without having to re-do everything)

Easy. Simply click the AI gameobject in scene. Right-click and hit "duplicate"

Then swap out the model and animation avatar and rename the bot.

How to Set Editor Colors?

(e.g. color of the sphere drawn in editor, representing Sight range)

Many Controllers have an "Editor color" setting. Simply click on it and set your custom color.