Blaze AI Documentation

Thank you for purchasing **Blaze AI**. For any questions and support you can email me directly at:
pathiralgames@gmail.com

Discord server here

Browse **Pathiral**'s packages here

About:

Blaze is a *Blaze-ingly* light, modern and comprehensive enemy AI engine. If you have enemies in your game no matter the genre, Blaze will definitely be of service to you and your game. It's been inspired by several AI systems of many modern games making Blaze a powerhouse in it's features, functionality and workflow.

It supports many sub-systems such as integrations with audio, animations, distractions system, attack, patrolling in different states, visions, emotions, local avoidance, reactions and a whole lot more.

Index:

Getting Started (for video *here*):

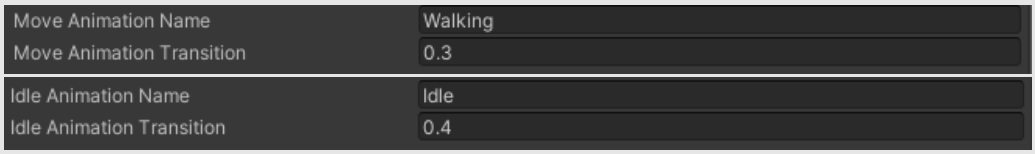
1. Click on your character object and add component “BlazeAI” script. You’ll find that it adds some required components.
2. At the far bottom of the script inspector you’ll find a button “Build Agent”, click it!
3. Now the character object has been built in a new hierarchal structure that Blaze is able to work with.
4. Make sure you set the added character controller, navmesh agent and capsule collider components to the radius and centers you want that fits your character.
5. Insert a plane into your scene. Open the Navigation tab and click bake.
6. Set the *ProxyOffset* inside the inspector to a good value that makes the debug sphere (green sphere *in scene view*) outside the carved boundries of the character.
7. Choose the ground layers in the inspector of Blaze. Make sure it has the same layer as the plane we baked in step 6.
8. Now if you start game you’ll notice your game object is moving randomly around the navmesh.
9. AND THAT’S IT TO BUILDING YOUR CHARACTER! Now all that’s left is to fill in the blanks of the inspector with animations, audios, waypoints and so on. We’re going to cover them all in this documentation later. **If you’re going to use animations, make sure to set the animator controller property and avatar in the Animator component and check Use Animations in normal state/alert state/attack state in Blaze. Also Don’t forget to set your animation files inside the Animator itself. We’ll cover this also so keep reading.**

In the next few pages we'll be covering first **Animations** and **Audios** independently and how they work when you find such properties as it's really important, then we'll start from the *top* of the inspector to the bottom.

Followed by the **public methods** and **properties** to use in programming *(page 21)*

Animations

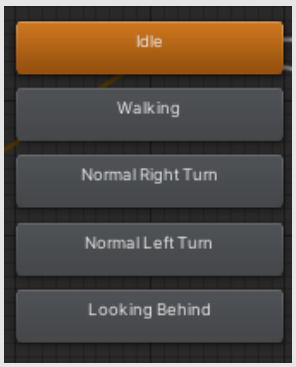
In many parts inside the inspector you'll find that you need to set the **Animation Name** and **Animation** T**ransition** of a certain thing. Like these inside the normal and alert state:



Now what is meant by these?

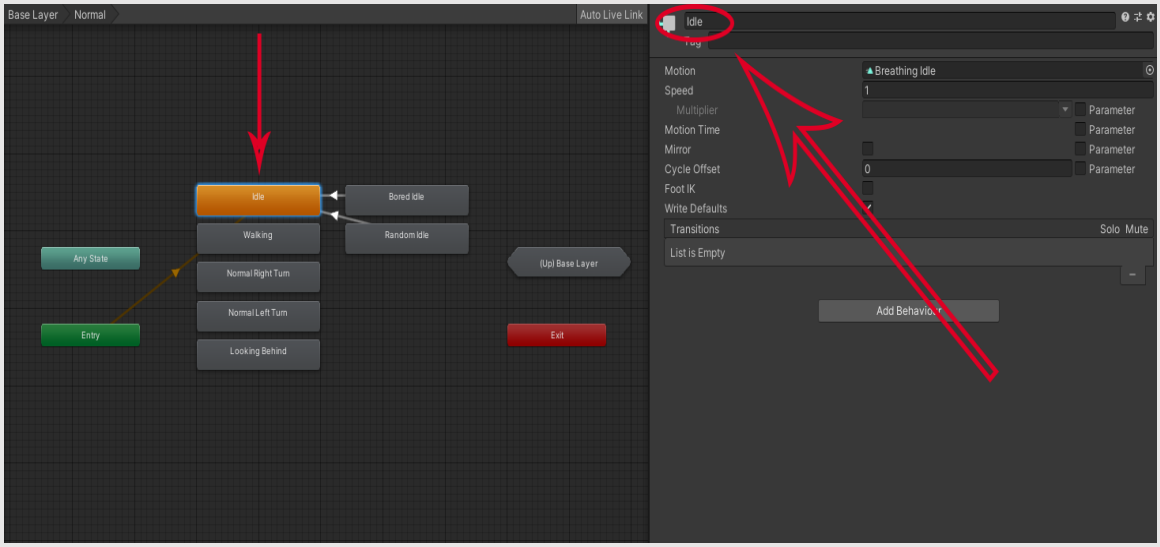
Let's start with the easy one. The Animation Transition as self-explanatory as it is, is simply the amount of time from a current animation to the animation in question (*the animation name*). In other words**: the blend/transition time.**

The Animation Name on the other hand is what is known inside of the Unity Engine as the *Animation State Name.* Which is this:



It’s the name of your animation inside the Animator. That’s it!

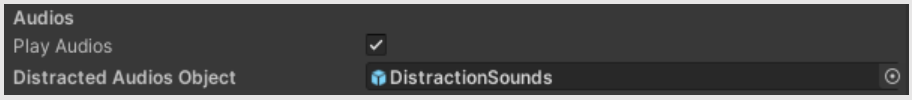
So my idle animation is called **Idle** so I set the *Idle Animation Name* property to **Idle**. They have to match. Same thing with Movement and everything else. You simply enter what the animation is called inside the Animator. You can change the animation state name inside the Animator like this



By clicking on the animation and then changing the name from the top right. But, remember you have changed the animation state name so you’ll have to change it’s name from Blaze AI to match the Animator.

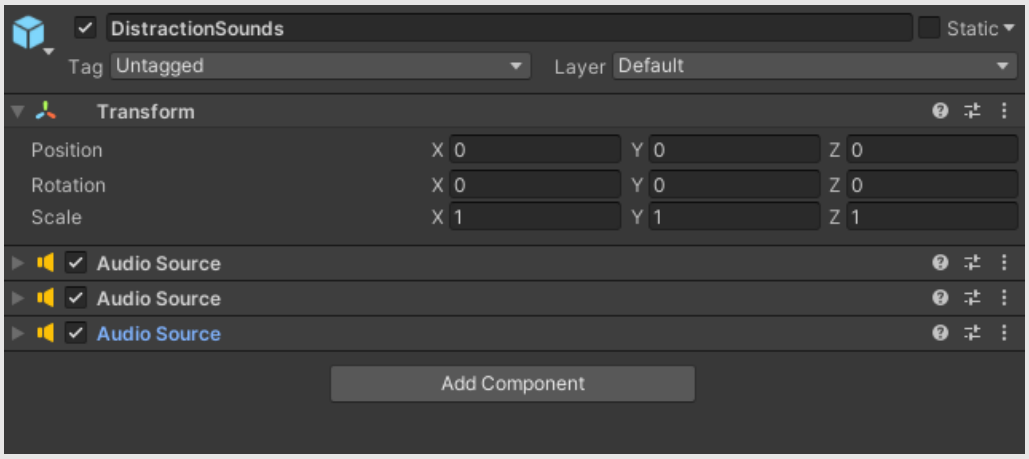
Audios

In the Audios sections of the inspector you'll come across properties like this:



Let's take this previous image as an example. This is from the distractions class. If you choose to Play Audios (checked), the audio to play will be randomly chosen from multiple audio sources from inside a game object. If there is no game object set the process will be ignored, if only one audio source is added then that only one will be played.

So as you can see the **Distracted Audios Object** is given a game object called **DistractionSounds**. This is what this game object looks like:



It has nothing other than 3 Audio Sources. So Blaze will randomly choose an audio source out of this 3 and play them when necessary. You can obviously set the audio source settings as you please.

So whenever you come across audios know that you'll have to provide **an empty game object with multiple audio sources, if non is**

provided, process will be ignored, if only one is set then that only one will be played each time.

Now that we covered Animations and Audios lets start from the top of the inspector.

Pathing

Ground Layers – Choose the layers that are the ground.

Path Recalculation Rate – The smaller the number (0.3) the higher the quality of navigation and obstacle avoidance but costlier on the CPU. It all depends on your game and target hardware. Suggested range is 0.5f – 1f.

Path Smoothing – If enabled, your character will curve towards corners making it look more realistic rather than turning at every angle.

Path Smoothing Factor – The strength of the path smoothing, the bigger the number the wider the curve. TAKE NOTE: path smoothing is never perfect and too much path smoothing can cause unwanted behavior and movement. Suggested from 1f-2f.

Path Finding Proxy – A game object built by Blaze that is responsible for path finding. This is set by default when you Build NPC. So leave it as it is. But, it's public nonetheless for the extreme cases of problems.

Proxy Offset – set the z position of the proxy that will pathfind. This should be outside the barriers of the navmesh obstacle.

Enable Gravity – If set to true the character will be pushed downwards by gravity.

Obstacles

Avoid Facing Obstacles – If set to true, a ray will be generated with a certain distance and if it detects an obstacle will stop the NPC. This is to avoid making the NPC standing too closely face-forward to an obstacle making the NPC look dummy.

Obstacle Ray Distance – Set the distance of the obstacle avoidance ray.

Obstacle Ray Offset – Position the ray relative to the character.

Obstacle Layers – Set the layers that are the obstacle. If the ray hits any of those layers then will be detected as an obstacle.

Avoidance

Tags To Avoid – During rare cases, sometimes the patrolling NPCs hit and get stuck to each other. In order to avoid this, set this list to the tags of the NPCs with the BlazeAI script you want to avoid. What will happen is if two get too close, one will stop and wait while the other walks by and the one stopped will then resume. Looking more realistic. Most probably all of your enemies have the same tag of “Enemies” or something of the likes. So just add this tag to the list, and if there’s something else moving in the map that might cause the NPCs to get stuck it would be best to add it’s tag to this list also but that would need trial and error and depends on each game.

Waypoints

The patrol routes of the NPC

Instant Move At Start – If enabled, the AI on game start will move instantly to the first waypoint. If disabled, will start idle.

Waypoints – A *vector3 array* where you add all the waypoints/patrol routes of your character. The points should be in world space and not local space.

Waypoints Rotation – Set the rotation to turn to when reaching waypoint. They are shown as small red squares in the scene view. The AI will face the red square on each waypoint. If 0 turning will occur.

Loop – If enabled, the patrol waypoints will be looped, meaning when the AI reaches the last point, it'll go back to the first waypoint and so on. If not set and reaches the end point then the AI will stay in idle state in that point indefinitely.

Randomize – If enabled, the waypoints set will be ignored and random points will be generated dynamically at run-time during patrol. Making it impossible to figure out where the AI will go next.

Right/Left Turn Anim Normal/Alert – Set the animation names for turning in each direction and each state. Used to turn at distractions or in patrolling.

Use Movement Turning – turns before moving to face the correct path.

Movement Turning Senitivity – how sensitive you want the AI to be to correct itself by turning.

Vision

Video explanation of vision here.

The agent's cone of vision that detects the surroundings.

Layers To Detect – Add all the layers that you want to detect in your world. That includes enemies and even obstacles. Anything you don't add will be seen right through.

Hostile And Alert Layers – Add the layers of the hostiles (enemies) and the alerts (gameobjects that will turn the AI to alert).

Hostile Tags – Add the tags that you want the agent to attack. Since in most games enemies attack only the player. Then add only the player tag here. You can also make the enemies attack each other by adding their tags here. If your game requires so. **The hostile needs to have a collider.**

Alert Tags – Add tags that when seen will turn the NPC into alert alert state such as tags of dead bodies or open doors that need to be closed. This makes your game highly interactive and immersive as you can react to unlimited things with specific audios and animations even.

Vision During Normal State – Set the cone angle and sight range when the NPC is in normal state.

Vision During Alert State – Set the cone angle and sight range when the NPC is in alert state. It's always best to increase the sight range and cone angle a little bit during alert state to make the game more challenging and realistic as the enemies now are more alert and know the player is there.

Vision During Attack State – Set the cone angle and sight range when the NPC is in attack state. Remember that probably 99% of

the time you would want the this cone angle to be 360 degrees. Because you want the NPC to always know you're there during a fight or else if it's something like 90 degrees and the NPC goes in for a punch and you dodge or roll behind him, then you're no longer in his cone of vision (90 degrees) and so it will fallback to patrolling in alert mode.

Sight Level – The Y-offset of the vision. Everything below the cone will be seen/detected and everything above it won't (unless max sight level is set)

Max Sight Level – This is the maximum sight level. Anything above it will not be seen everything below it (between it) and the vision cone will be seen/detected.

Head – Optional: add the head gameobject of the character, this will automatically fix rotation of the vision according to the head as well as the sight level. If empty, the vision will be fixed to project forwards of where the script is attached (body of character).

Pulse Rate – Amount of frames to pass each time before running vision. The lesser the number, the more accurate the vision but more expensive. The bigger, the less accurate but easier on the CPU. Remember, the frames are fairly negligible so accuracy isn't that big of a measure. Recommended to have it at 5.

Show Normal Vision, Show Alert Vision, Show Attack Vision – This is for debug only and if enabled will show in editor view the vision cones for easier debugging.

Normal State

Normal state is the normal patrol where the NPC hasn't seen any hostile or alert tag object.

Use Normal State On Start – If enabled, the character will start being in the normal state. Either this or **Use Alert State On Start** can be used.

Move Speed – The speed of movement in normal state.

Rotation Speed – The speed of rotation in normal state.

Wait Time – The time of idle before moving to the next waypoint.

Randomize Wait Time – If enabled, the wait time will be randomized between two values. If disabled, the below property will not be read.

Randomize Wait Time Between – Randomize the wait time between two values.

Use Animations – If enabled, animation properties will be read and used.

Idle Animation Name – Animation state name to trigger when idle in normal state.

Move Animation Name – Animation state name to trigger when moving in normal state.

Use Random Animations On Idle – If enabled, when the NPC reaches a waypoint and waits in idle state. There will be a 50% chance that it plays a random idle animation from the below property. Like stretching or ties his shoe or checking his gun. These

are all random idle animations you might want to play to give a more realistic look.

Random Idle Animation Names – A string array for adding all the animation state names you want to play. One will be automatically chosen at random and played.

Enable Scripts – If enabled, when the NPC is idle the script on idle will be enabled and script on move will be disabled and vice versa.

Enable Script On Idle – MonoBehaviour, in normal state this script will be enabled when the NPC is idle and disabled when moving.

Enable Script On Moving – MonoBehaviour, in normal state this script will be enabled when the NPC is moving and disabled when idle.

Play Audios On Patrol – If enabled, this NPC will play a random audio every set interval when in normal state.

Play Audio Every – The amount of time in seconds to pass before choosing an audio to play in normal state.

Alert State

Alert state is when the NPC sees a hostile or alert tag object and notifies others of his alert state.

Use Alert State On Start – If enabled, the character will start being in the alert state. Either this or **Use Normal State On Start** can be used.

Move Speed – The speed of movement in alert state.

Rotation Speed – The speed of rotation in alert state.

Wait Time – The time of idle before moving to the next waypoint.

Randomize Wait Time – If enabled, the wait time will be randomized between two values. If disabled, the below property will not be read.

Randomize Wait Time Between – Randomize the wait time between two values.

Use Animations – If enabled, animation properties will be read and used.

Idle Animation Name – Animation state name to trigger when idle in alert state.

Move Animation Name – Animation state name to trigger when moving in alert state.

Use Random Animations On Idle – If enabled, when the NPC reaches a waypoint and waits in idle state. There will be a 50% chance that it plays a random idle animation from the below property. Like reloading his gun or aiming up and down in alert

state. These are all random idle animations you might want to play to give a more realistic look.

Random Idle Animation Names – A string array for adding all the animation state names you want to play. One will be automatically chosen at random and played.

Return To Normal State – If enabled, After a certain amount of time, the NPC will return to normal state.

Time Before Returning Normal – The amount of time to pass in seconds before returning to normal state.

Returning Duration – The amount of time to pass in seconds the NPC playing the return audio, return animation and staying idle before being completely in normal state.

Use Animation On Return – If enabled, an animation will be played when returning to normal.

Animation Name On Return – Animation state name you want to play when the NPC is returning to normal state.

Play Audio On Return – If enabled, a random audio will be chosen to be played on return to normal state.

Enable Scripts – If enabled, when the NPC is idle the script on idle will be enabled and script on move will be disabled and vice versa.

Enable Script On Idle – MonoBehaviour, in alert state this script will be enabled when the NPC is idle and disabled when moving.

Enable Script On Moving – MonoBehaviour, in alert state this script will be enabled when the NPC is moving and disabled when idle.

Play Audios On Patrol – If enabled, this NPC will play a random audio every set interval when in alert state.

Play Audio Every – The amount of time in seconds to pass before choosing an audio to play in alert state.

Attack State

When the npc finds a hostile enemy it attaches to the enemy a script component; ***Blaze AI Enemy Manager*** *at runtime. This script is the npc scheduler which makes npcs attack one by one. If you want to further improve or increase the enemy numbers attacking and you're a programmer you should take a look at this script.*

If the script already exists then it won't be added. Adding this script manually before hand in editor time on your player allows in controlling the interval of when to make the NPCs attack. So keep this in mind.

If the targeted enemy suddenly changes tag to a non-hostile tag the npc will not attack.

When an agent is in attack state, there are two sub-states that will be triggered at some point.

The first being the **idle-attack state** meaning, the NPC is waiting for it's turn (or timer) to attack.

The second being the **actual-attack state** meaning, it's the NPC's turn to attack so it'll actually move to the attack distance and when reached will deliver the hit.

Cover Shooter Options:

Cover Shooter – A bool to set either true or false. If true, the cover shooter attack behavior tree will be implemented. Which is basically finding cover, attacking, finding cover again and so on.

Cover Layers – The layer mask of the obstacles you want to detect as covers.

Hide Sensitivity – From -1 to 1. The smaller the number the better the hiding spot but may be harder to find such good spot. Best from -0.25 to 0.

Search Distance – The radius of searching for cover from current npc position. This can't be bigger than *Distance From Enemy* property. Clamped if so.

Min Obstacle Height – The minimum height of obstacles to be eligible for covers. The height is measured using collider.bounds.size.y. Use *GetCoverHeight* script to get cover height.

First Sight Chance – Set the chance of attacking when seeing a hostile for the first time. Can be set to 3 options. Take Cover, Always Attack and Randomize. If set to Take Cover, the npc will always go to cover first when sees a hostile. If set to Always Attack, the npc will always attack first. If set to Randomize, there's a 50/50 chance of either taking cover or attacking. Making this option the most immersive and realistic. This is the most recommended option.

Cover Blown State - Set the state when a hiding npc gets it's cover blown. Meaning, an enemy can see it. Can be set to 3 options. Take Cover, Always Attack and Randomize. If set to Take Cover, the npc will move to another cover. If set to Always Attack, the npc will always attack when cover is blown. If set to Randomize there's a 50/50 chance of either taking another cover or attacking.

Attack Enemy Cover – Should the npc open fire always on enemy's cover only, always attack the enemy itself or randomize. This will only be taken into account if enemy is actually hiding behind an obstacle, if not, the enemy itself will naturally be attacked.

Cover Animations:

High Cover Height: The height for the high cover animation. Anything equals this value or *greater* will be considered as a high cover. The height is measured using collider.bounds.size.y. Use *GetCoverHeight* script to get cover height.

High Cover Animation: The animation name for the high cover.

Low Cover Height: The height for the low cover animation. Anything equals this or *smaller* will be considered as a low cover. The height is measured using collider.bounds.size.y. Use *GetCoverHeight* script to get cover height.

Low Cover Animation: The animation name for the low cover.

Rotate To Normal – If set to true, the npc will rotate to match rotation of cover normal, meaning it's back will be on the cover. If set to false, no rotation will occur but instead take cover in the same current rotation.

Use Scripts – Do you want to trigger certain scripts when in high or low covers? This is useful to change something like the character controller height depending on cover type.

High Cover Script – Set to a MonoBehaviour that will be enabled when in high cover and disabled when out of cover.

Low Cover Script – Set to a MonoBehaviour that will be enabled when in low cover and disabled when out of cover.

Cover Animation Transition – Set the transition time from current animation to low/high cover animation

Distance From Enemy – The distance between the NPC and the enemy where the idle-attack state will be triggered and the NPC will be waiting for it's turn to attack the enemy.

Attack Distance – The distance between the NPC and the enemy when delivering the attack.

Layers Check Before Attacking – Sometimes you want to make sure a certain layer isn't between the agent and it's target before attack. Example: setting it to the other agent layers to avoid friendly fire.

Call Others – Will call other agents when in attack state.

Call Radius – The radius to search for other agents to call.

Agent Layers To Call – The layers of the agents to call.

Call Others Time – The amount of time in seconds to pass before calling others. As long as the agent is in attack state this will keep running every time this value has passed.

Receive Call From Others – Get called by other agents.

Attack In Intervals – If enabled, this NPC will not be part of an enemy group waiting for its turn to attack. But, will attack in intervals. This is best for ranged enemies. When they throw spears for example every certain amount of time at the player from a long distance. Not waiting for their turn.

Attack In Intervals Time – The amount of time to pass in seconds before attacking. Will only be read if **Attack In Intervals** is set to true.

Randomize Attack Intervals – If set to true, the **Attack In Intervals Time** property will be randomized between two values after each attack.

Randomize Attack Intervals Between – Randomizes the attack intervals time between these two values after each attack.

Move Backwards – If enabled, when the enemy gets too close to the NPC. It will move backwards. What is meant by *too close* is the distance between the NPC and the enemy is less than what is set in **Distance From Enemy** property. Meaning the enemy is trying to get close to the NPC.

Move Backwards Dist – If less than this value the npc will move backwards.

Move Backwards Attack – If enabled, the NPC will be eligible to attack while backing up. Best for ranged enemies like those

throwing spears. If the enemy gets too close they can backup and throw spears.

Move Speed – Movement speed in attack state.

Move Backwards Speed – Movement speed while moving backwards. This is only read and used if **Move Backwards** is set to true.

Use Animations – If enabled, animation state names will be read and triggered.

Idle Animation Name – The name of the animations state name of the attack-idle state (NPC waiting for it's turn to attack), in other words the fight stance.

Move Forward Animation Name – The animation state name to trigger when NPC is moving to attack position and running after enemy.

Move Backwards Animation Name – The animation state name to trigger when NPC is backing up from enemy.

Move Backwards Attack Animation Name – The animation state name to trigger when NPC is backing up and attack.

Attack Animations – The animation names to play on attack. One will be chosen at random on each attack. If only one is set, then that one will always be played.

Always Look At Enemy – keep rotating to enemy even on attack.

Attack Duration – An array that is automatically set according to the attack animations size. Set the duration of the attack for each animation. It's the attack time before backing up.

Attack Script – When this enemy attacks, this attack script will be enabled. In this script you should code your actual attack whether it's shooting a gun or throwing spears or whatever it is you want your actual attack to be. The attack code is up to you and this is where it will be and what will be triggered.

Emotions

This is a feature of Blaze where it simulates certain emotions in the NPC. Gives you full control to simulate basically any given emotion with audio and animations.

Surprised

Surprised is triggered when the NPC is in normal state and sees a hostile enemy for the first time. Thus gets surprised. You can play an animation where the NPC is actually surprised, scared or shocked or you can play something like a raging battle cry for example as he alerts others. It's all up to your design.

Use Surprised – If enabled, the NPC will use the surprised emotion when sees an enemy for the first time in normal state.

Surprised Duration – The duration to pass in seconds before being in complete attack state and finishing the surprised emotion.

Use Animations – If enabled, the surprised emotion will read and trigger animation state names.

Surprised Animation Name – Animation state name to trigger when in surprised emotion.

Use Audio – If enabled, when in surprised emotion a random audio source will be selected and chosen from the game object.

Distractions

Distractions is interrupting an NPCs patrol and forcing him to look at the distraction direction and even move to that location to check for hostiles. **Triggering the distraction programmatically will be discussed in properties and public methods section. Setup distractions tutorial here.**

Distractions are made using the **Blaze AI Distraction** script component. Inside this script is a public method **TriggerDistraction()** which will do all the work for you for distracting the NPCs. So use this for distractions.

Always Use – If enabled, this NPC will be eligible to get distracted.

Auto Turn – If enabled, this NPC will auto-turn to distraction direction. Always best to use it with animations.

Turn Speed – If **Auto Turn** is enabled, this is the speed of the turn to the distraction direction.

Turn Reaction Time – The amount of time to pass in seconds before turning to distraction direction. It's best to leave a little gap of 0.3-0.5 seconds. To give the realistic look of not suddenly turning to distraction the moment it happens.

Turn Alert On Distaction – If enabled, this NPC will turn to alert-state if distracted. All depends on your game design whether you want to enable this or not.

Move To Distraction Location – If enabled, this NPC will go check the distraction location. **If the distraction location is on a valid NavMesh.**

Check Distraction Priority Level – When a group of enemies are distracted and all of them have **Move To Distraction Location** checked, only **one** will be chosen to check that location. The one chosen will be based on this priority level. The NPC with the highest priority is the one that will move to check the distraction location. If two or more have the highest priority, then the one with the smallest distance to the distraction is the one that will be sent to check. Sometimes you want a certain type of a strong enemy to have the highest priority to check distractions, to make the game more challenging maybe? Again, depends on your game.

Move To Distraction React Time – How long after being distracted do you want the NPC to start moving to check location?

Checking Time – The amount of time to spend in the distraction location checking.

Use Turn Animations – If enabled, will use the turning animations in the waypoints class, if set.

Distraction Check Animation – If enabled, when the NPC reaches distraction location will play a special animation.

Distraction Check Animation Name – Animation state name to play when NPC reaches distraction location.

Enable Script – If enabled, a script will be enabled when distracted.

Distraction Script – A script to enable when NPC is distracted. This script will be disabled when the NPC is no longer distracted and goes back to patrolling.

Play Audios – If enabled, an audio will be played on distraction.

Hits

Hit is when the agent is in a complete vulnerable state and pauses it's current state and resumes it after a certain duration. To actually trigger the hit state programmatically, it will be discussed in the ***properties and public methods*** *section.*

Hit Duration – The amount of time in seconds for the NPC to be in a hit state. Being completely vulnerable and playing hit animation before resuming it's previous state.

Cancel Attack If Hit – Choose to either have the attacking agent's attack be cancelled if it got attacked midway or not. If set to false the agent will resume it's attack after hit.

Use Animations – If enabled, animation state names will be triggered during hit state.

Animation Name – Hit animation state name to play when hit. You can dynamically change this name to make a certain hit animation on specific parts of the body. To access programmatically:

```
GetComponent<BlazeAI>().hits.animationName = "head hit";
```

Anim Play Gap – Amount of seconds to pass before hit animation can be played again.

Use Audios – If enabled, a random audio will be played when hit.

Death

Simply put, it's dying. Blaze AI gets disabled to prevent unnecessary over-head and plays a death animation with audio and script to enable.

Use Animation – whether or not to use animation state name in death.

Animation Name – the death animation state name. You can access this programmatically and change it according to the way of death like:

```
GetComponent<BlazeAI>().death.animationName = "head shot";
```

Use Audio – Play a random audio on death.

Enable Script – If enabled, on death a script will be enabled.

Script To Enable – The MonoBehaviour to enable on death.

Public Methods and properties

All classes/variables in the inspector can be accessed programmatically using the camel-case convention. Making only the first letter small case.

For example: Path Recalculation Rate -> pathRecalculationRate

Move Speed inside of Normal State class = normalState.moveSpeed

Here are handy public variables and methods to call.

state – returns a **State** enum of either (State.attack, State.alert, State.normal or State.hit)

waypointIndex – returns the current index of the waypoint

distracted – returns a bool, either true or false whether the NPC is distracted or not.

enemyToAttack – returns a game object of the enemy the NPC is currently attacking or targeting.

checkEnemyPosition – if NPC has been alerted by another of an enemy position, this is where you can get the position of that enemy you're going to.

Death() – this will trigger the death system. The agent will play the death animation, audio and enable a death script and also disables Blaze AI permanently.

Undeath() – this will return the agent to being alive.

Hits(GameObject) – this will trigger the hit system, the agent will be hit and vulnerable for a certain time. If **Hits()** is called without

providing a game object the enemy will not know where the enemy who hit it is and will simply continue in alert state. If you call **Hits(GameObject)** with providing a game object of who hit it. After the hit state is done, agent will run to that gameobject's position to check for the enemy there.

CallAgentToLocation(Vector3 location, float secondsUntilMove, string animationToPlay, string stateToTurn) – calling this method will make the AI run to the passed **location** argument. You can also pass seconds to pass before moving and even an animation name to play before moving. If no animation passed will play the idle state animation of the current state. Last argument - the **stateToTurn** – you can set the state you want the AI to switch to when moving to the location. StateToTurn takes a string of either: "normal", "alert" or "attack". Take note: you can't call this function when the AI is already in attack state.

Additive scripts – These are extra scripts provided that increase the usability of Blaze.

BlazeAIEnemyManager – this is added by Blaze when it sees a hostile tag, if it's already added to the hostile object then Blaze will not add it again. This script component is the enemy scheduler that makes the Blaze NPCs attack the hostile one at a time. You can add this before-hand in editor time to be able to control the interval of attacks of NPCs on this script (hostile enemy) or improve the code to make numerous npcs attack at the same time.

BlazeAIDistraction – add this script to any game object you want to make as a distraction. Trigger this distraction programmatically using **TriggerDistraction()**. This is how it's triggered in full:

```
GetComponent<BlazeAIDistraction>().TriggerDistraction();
```

This will make any Blaze script be distracted and look at the distraction trigger source direction. You can obviously within the function of triggering the distraction also play an audio. This will simulate or look like as if the npc has heard a sound and got distracted by it. Sound distractions, this is how it works in games.

BlazeAIRandomEnemyDistance – Add this script to whichever NPC you want and it'll automatically (on start) randomize the **Distance From Enemy** property between two values. To give the NPCs a more genuine look when they're surrounding the player with randomized distances within the same (somewhat) range

BlazeAIGetCoverHeight – Add this script to any cover obstacle and it'll print you it's height in the console log.

BlazeAICheckLocation – Use this script to make all agents within a radius run in attack state to check a location. Can be used with something like explosions. Add this to a gameobject, set the radius and do: *GetComponent<BlazeAICheckLocation>().Trigger()*

BlazeAIFall – Add this script to your agent with the BlazeAI component to add falling capabilities.

Browse all of **Pathiral**'s packages here