

# PATHOS

User Manual

## What is PathOS?

PathOS is an end-to-end, lightweight framework for simulating player behaviour. PathOS agents approximate player navigation in a game's world, and can be viewed in real-time or recorded for later visualization. Agents can also be customized to mimic different player motivations.

PathOS is built for Unity, and designed to operate on top of your existing game projects, requiring no instrumentation or modification of game assets or code.

Happy playtesting!



# Contents

- Quickstart Guide . . . . . 1
- Project Set-Up . . . . . 2
- Level Markup . . . . . 3
  - Game Entity Types. . . . . 4
- Runtime Interface . . . . . 5
- Agent Customization . . . . . 7
  - Custom Profiles. . . . . 7
- Batch Simulation. . . . . 8
- Data Recording & Visualization. . . . . 9
  - Viewing Agent Paths . . . . . 9
  - Heatmaps . . . . . 9
  - Viewing Agent Interactions . . . . . 9
- Complete System Reference . . . . . 10
  - AI Agents & Configuration . . . . . 10
  - Data Visualization . . . . . 12

# Quickstart Guide

## First thing's first

Set up the Unity Navmesh if you haven't already from Window > AI > Navigation. Make sure you have PathOSManager and PathOSAgent objects in the scene - prefabs can be found in PathOS/Prefabs.



## Level Markup

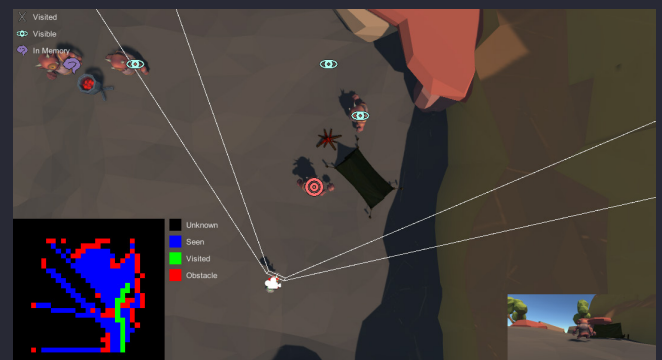
Use the **Level Markup** tab of the Manager Inspector (see [Level Markup](#)) to label important or interactive objects in the scene (e.g., enemies, collectibles). Tag any objects that would be indicated on a player's compass or minimap with the "Always Known" flag in the Manager's Entity List.

## Agent Set-Up

Adjust the motive sliders on the Agent object to reflect the desired profile, or select a profile from the available presets and apply it to the agent (see [Agent Customization](#) and [Custom Profiles](#)).

## Running the Simulation

Hit play to start the simulation and you can watch the agent navigate in realtime. Select it in the hierarchy to view an onscreen overlay showing its targeting logic, mental map, and player view (see [Runtime Interface](#)).



## Extras

You can run multiple agents automatically as part of a testing batch (see [Batch Simulation](#)) and record data for later review and visualization (see [Data Recording & Visualization](#)). For a complete explanation of the system's components and Unity Inspector properties, check out the [system reference pages](#)!

# Project Set-Up

## Demo Project

The included demo project and prefabs are set up to work "out of the box" - all you'll have to do is hit the Bake button in Unity's Navigation panel to re-bake the Navmesh after changing the level layout.

## Will my project work with PathOS?

Hopefully! If your game involves players moving around a 3D world, the answer is probably yes. As long as you can bake a Unity Navmesh for your scene, you can use PathOS to test your level designs. However, the tool has its limitations - be sure to read "A Few Caveats" below to see if the framework is a good fit for you.



## Navmesh

PathOS Agents work with Unity's Navmesh system for pathfinding. For the tool to work, you'll need to bake your Navmesh from Window > AI > Navigation. If you're starting from scratch, make sure that the baked agent settings (height, radius, etc.) match the settings on the Unity NavMeshAgent component of your PathOS Agent prefabs and GameObjects.

## A Few Caveats

**Tool Usage.** PathOS is a tool for simulating navigation, not complete gameplay. Agents' navigation is dependent on the contextual information you provide - which GameObjects are collectibles, enemies, goals, and so on - but agents don't interact with your game's mechanics. Agents can't fight enemies to tell you if your combat system is too difficult. At least, not yet.

**Level Layout.** Agents navigate in 3D, but their spatial logic is planar. This means PathOS works best when your level is mostly laid out at a uniform altitude with a defined ground plane - or when you can test your level in separate, mostly flat sections. PathOS will not work properly for levels with vertical layering.

**Visibility.** Whether or not agents can "see" game objects is based on Unity's physics system, so anything you want to occlude visibility should have colliders attached. Visibility is calculated approximately, so don't be surprised if what you see through the player's POV camera doesn't match up exactly with the agent's logic.

# Level Markup

For agents to navigate in the context of your game, they need to understand which objects in the level can be interacted with, and what purpose they serve. To tag game objects, use the **Level Markup** tab of the PathOS Manager Inspector.

## Markup Brush

In the **Level Markup** tab of the Inspector, you can click on one of the entity types to activate tagging for that type (your cursor will change). In the scene, click on objects to tag them with the selected entity type. You can also use this mode to change the tag on already labelled objects, or clear tags from labelled objects.

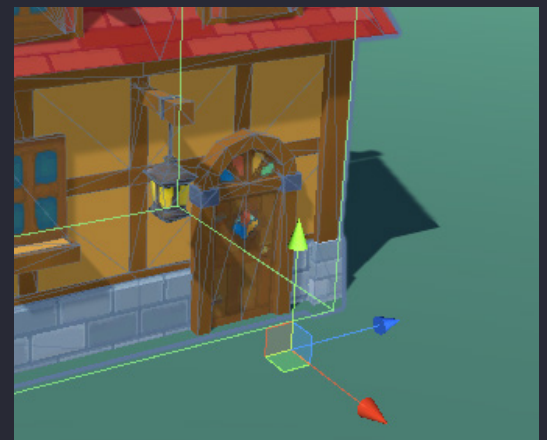


## Entity List

You can also edit tags via the **Level Entity List** tab. You can add and remove tags using the '+/-' buttons at the bottom of the list. Here, you can also change the GameObjects referenced by each tag, or set an object as "always known" (mimicking the effect of entities which would be indicated on a player's minimap or compass).

## A Note on Entity Locations

When agents target and visit game entities, they use the position of the Transform on the tagged GameObject. For large objects with colliders attached (e.g., buildings), a parent or proxy GameObject with the desired "visit location" should be used as the tag reference. The prefabs included in the demo project have already been set up with parent GameObjects with suitable pivot points chosen.



# Game Entity Types

There are nine different tags available for level objects during the markup process. Here is a summary of their meaning. Type tags are used by agents to help drive their navigation through the game world.



**Optional Goal.** In-game missions or objectives that are optional. (e.g., sidequest marker)



**Mandatory Goal.** Objective that must be completed to finish the level. (e.g., main mission marker)



**Final Goal.** Objective that would allow the player to complete/exit the level, if applicable.



**Collectible.** Item that can be collected in game for achievement value. (e.g., treasure)



**Self-Preservation Item.** Item that can be collected to boost player survivability. (e.g., health/ammo)



**Enemy Hazard.** Hazard that could result in a combat encounter if engaged. (e.g., monster)



**Environment Hazard.** Interactive hazard that will not result in a combat encounter. (e.g., traps)



**Point-of-Interest.** Environment landmark intended to draw in players for exploration. (e.g., setpieces)



**NPC.** Non-hostile character that can be interacted with. (e.g., questgiver)





# Runtime Interface

During playmode, select the agent in the hierarchy expand the PathOS Renderer component in the Inspector to enable the PathOS Agent UI. Unchecking the "3D Gizmos" option in Unity is recommended.

## Controls

**Spacebar.** Toggle on-screen legend for mental map and Gizmos.

**Click and drag.** Pan the PathOS World Camera (if present).

**Mouse wheel.** Zoom in/zoom out with the PathOS World Camera (if present).



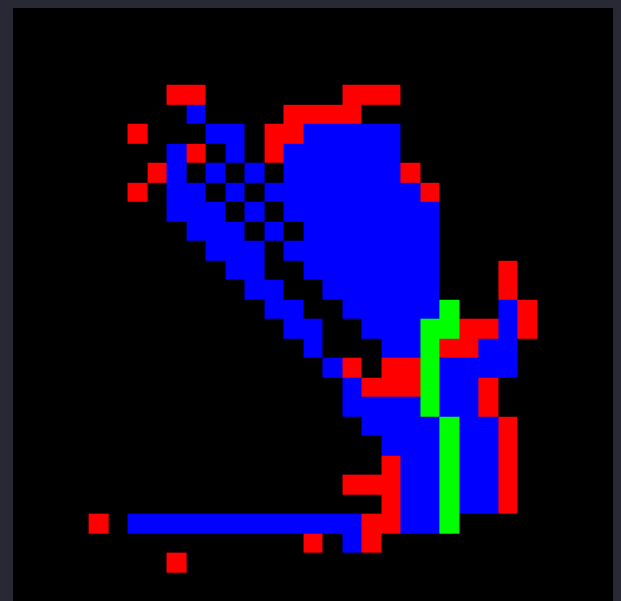
## UI Layout

In the lower left corner, the agent's mental map is displayed. In the lower right, the agent (player) POV camera view is rendered. Gizmos are displayed on level entities indicating their state in the agent's world model.

## Mental Map

The mental map shows the agent's internal, tile-based representation of the scene's spatial layout. Four colours are used to indicate the state of each tile as perceived by the agent:

- Black - Unknown
- Blue - Seen as unobstructed (e.g., flat ground)
- Red - Seen as obstructed (e.g., wall)
- Green - Visited/traversed by the agent







## Player View

The player view displays what the agent is currently "seeing" through its player POV camera. This can be used to double-check visibility of game objects outside the approximate system used by agents.

## Entity Gizmos

For all game entities tagged using the markup system, Gizmos are displayed indicating how they factor into the agent's logic at any given time. The meaning of these gizmos is as follows:



### [NO ICON]

Entity is not affecting agent logic - it is not visible, remembered, or previously visited.



Currently targeted by the agent. Can be applied to a game entity, or an empty point in the scene (while the agent is exploring).



Entity is contained in the agent's memory.



Entity has been previously visited.



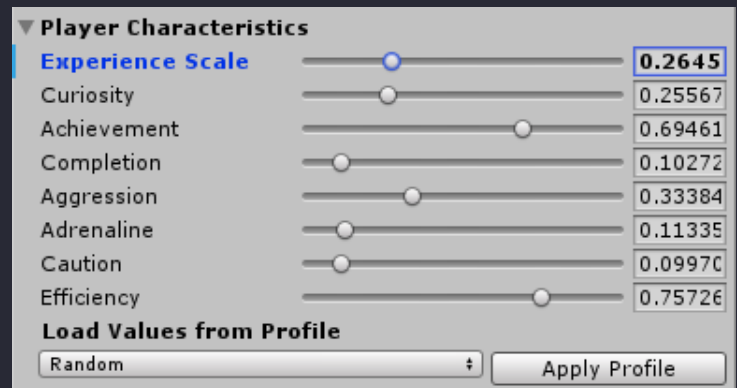
Entity is currently visible to the agent.



Entity has been determined to be unreachable (the agent cannot navigate to it using the Navmesh).

# Agent Customization

Agents are governed by their motives, which reflect player motivations and can be customized for each agent. To change an agent's behaviour, you can use the **Player Characteristics** tab of the PathOS Agent Inspector. Here you can tweak individual motives, or apply a custom profile preset (see below).



Player Characteristics	
Experience Scale	0.2645
Curiosity	0.25567
Achievement	0.69461
Completion	0.10272
Aggression	0.33384
Adrenaline	0.11335
Caution	0.09970
Efficiency	0.75726

Load Values from Profile

Random Apply Profile

## Motives

There are seven agent motives in addition to the experience scale (e.g., amount of prior game experience). These motives affect the agent's behaviour and affect the way it will evaluate tagged entities as potential destinations. These motives are as follows:

**Curiosity.** The motivation to explore for exploration's sake, and discover all a level has to offer.

**Achievement.** Wanting to earn achievements, complete game objectives, and rack up a high score.

**Completion.** The desire to complete every in-game log, find every collectible, and so on.

**Aggression.** A drive to seek out conflict and combat, dominating the game world.

**Adrenaline.** Thrill-seeking, not only in combat, but in besting challenges or environmental gauntlets.

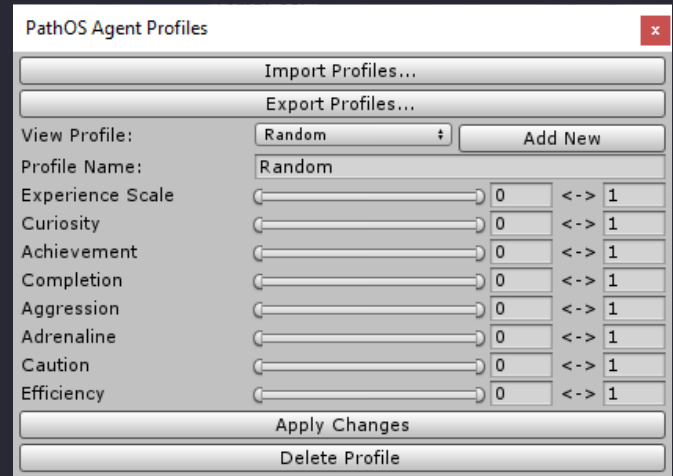
**Caution.** Taking care to maximize survivability, avoiding combat and hoarding resources.

**Efficiency.** Wanting to get through a level as quickly as possible, prioritizing necessary goals.

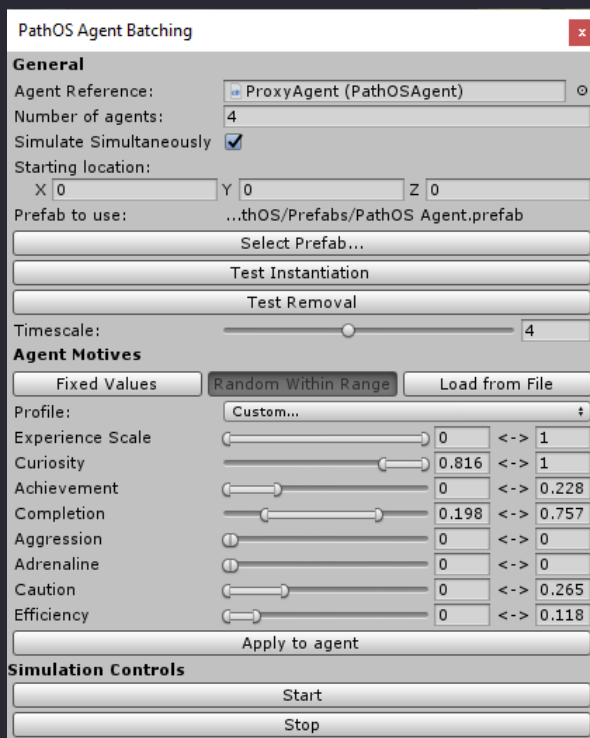
As a note for advanced users, you can view and edit the relationship between these motives and level markup tags in the **Motive Weights** tab of the PathOS Manager Inspector. Positive weights between a motive and entity tag indicate that an agent with a high value for that motive will be drawn to entities of that type. Conversely, negative weights will cause repulsion, and a zero weighting indicates no effect of the chosen motive on the agent's behaviour around entities of the chosen type.

## Custom Profiles

To manage agent profiles, go to Window > PathOS Profiles. From here, you can create and edit profiles, as well as loading or saving files to carry profile sets between projects. Each profile has a range defined for the seven agent motives, as well as experience. When a profile is applied to an agent, values are picked randomly from these ranges.



## Batch Simulation



To simulate multiple agents automatically, go to Window > PathOS Agent Batching. From here, you can choose whether to simulate agents consecutively or simultaneously. For consecutive simulation, be sure to drag in a reference to the agent in your scene. For simultaneous simulation, specify a prefab for the system to instantiate for each agent needed, as well as a starting position for agents in world space.

You can also adjust the timescale of the simulation to speed things up - note that any time limit set in the Manager inspector will proceed in real time, however.

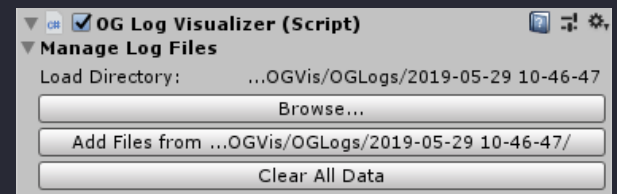
Agent motives will be initialized automatically based on the settings specified - either using fixed values, randomizing them within a range, or loading their values from a file. For range initialization, a custom profile (see above) can be selected to automatically define ranges. For file initialization, a .CSV file should be specified containing values for each of the desired agents (a sample file is included with the demo project).

# Data Recording & Visualization

To automatically record logs containing information on agents' navigation and visiting level entities, toggle the "Enable Logging" setting on the OGLog Manager component. This will record logs both if playmode is triggered manually, or if simulation is handled en masse through the agent batching window.

## Loading Data Logs

Load logs through the OGLog Visualizer component in the Inspector. Logs are stored as CSV files. To load them, select the directory where logs are located and hit "Add Files from...".



## Display Filters

In the **Filtering/Display Options** tab, you can control what data is visible. The "time range" (in seconds) will exclude data from outside the specified range. The "display height" controls at what altitude (in units) visualization elements will be rendered in the scene. You can also choose which agents should be included or excluded from the visualization. The profiles of the agents used to create the data can be viewed by clicking the ellipsis next to each agent name.

## Viewing Agent Paths

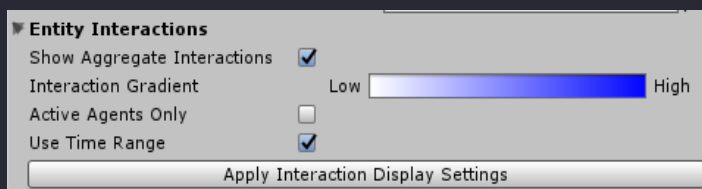
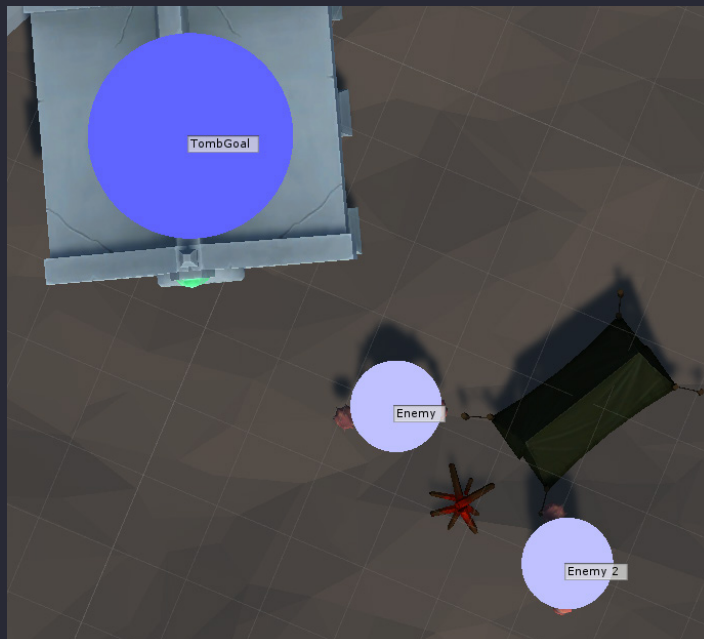
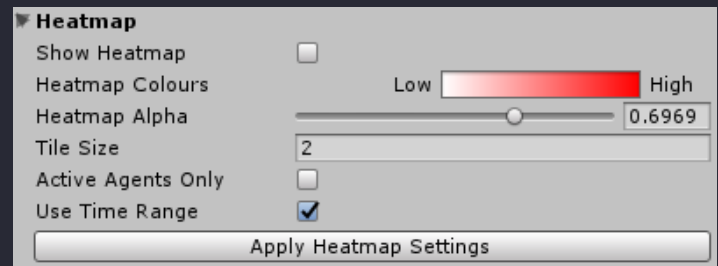
Individual agent paths through the world can be toggled from the **Individual Paths** tab. Enabling "individual interactions" will also show a record of individual agents visiting level entities along their journey through the level. From here, you can also specify colours for displaying each agent's trajectory.



## Heatmaps

To use the heatmap functionality, ensure that a child object of the OGLog Visualizer has the OGLog Heatmap Visualizer component attached (the provided prefab is set up with this configuration).

The **Heatmap** tab can be used for customization (e.g., colour scheme). The "tile size" attribute can be used to adjust the heatmap granularity. If "active agents only" is enabled, only data from agents enabled in the **Filtering** tab will be used. If "use time range" is enabled, only data from the time range specified in the **Filtering** tab will be used.



## Viewing Agent Interactions

In the **Entity Interactions** tab, you can visualize how many agents visited different level entities in the scene. Each entity is shown as a circle, with scale and colour adjusted to reflect the proportion of agents which visited each entity.

If "active agents only" is enabled, only data from agents enabled in the **Filtering** tab will be used. Circles will be scaled according to the number of total agents in the enabled group. If disabled, data from all agent logs loaded will be used and scaled according to the total number of logs loaded.

If "use time range" is enabled, only data from the range in the **Filtering** tab will be used (i.e., entities visited outside this time range are excluded from the visualization).

# Complete System Reference

This section explains the function of each of the included script components, as well as the variables exposed in the Inspector. Most Inspector properties have tooltips in Unity for your reference, and the included prefabs (PathOS > prefabs) are designed to work out-of-the-box.

## AI Agents & Configuration

### PathOS Manager

There should be one PathOS Manager in the scene. It is responsible for storing level markup data (see [Level Markup](#)) and providing agents with a lookup table for motive scoring (see [Agent Customization](#)).

**Limit Simulation Time.** If enabled, once testing begins, playmode will exit automatically after a time limit.

**Max Simulation Time.** The time limit for the above (in real time).

**Final Goal Triggers End.** If enabled, once all agents have reached the object tagged as the final goal (if one exists), playmode will exit automatically.

**Show Level Markup.** If enabled, Gizmos indicating level markup will be displayed when the Manager is selected and the PathOS Manager component is expanded in the Inspector.

### PathOS Agent

This component makes a GameObject function as a testing agent. When it is added, a Unity NavMeshAgent component will be added automatically, along with the PathOS Agent Memory, Eyes, and Renderer.

**Freeze Agent.** If enabled, the agent's logic and movement will be frozen (can be toggled during playmode).

**Player Characteristics.** Control the agent's motivation profile (see [Agent Customization](#)).

**Explore Degrees.** When exploring, the agent will cast out rays in its FOV every X degrees.

**Explore Degrees (Back).** When exploring, the agent will cast out rays outside of its FOV every X degrees.

**Look Degrees.** When looking around, the agent will rotate to either side by X degrees.

**Visit Threshold.** How close (in units) the agent needs to pass by a game entity to consider it visited. This value should be positive. If it is too large, the agent will consider entities visited when it is still very far away.

**Explore Threshold.** How close (in units) two exploration targets must be to be considered the same.

**Explore Target Margin.** When exploring, the radius the agent uses to find a target position on the Navmesh, in units (outside this radius, the agent will give up and deem the location unexplorable).

## PathOS Agent Memory

Controls agent memory. Added automatically with the PathOS Agent component.

**Grid Sample Size.** How large (in units) each tile of the agent's mental map will be.

## PathOS Agent Eyes

Controls agent perception. Added automatically with the PathOS Agent component.

**Player Camera.** The camera which should be used for the agent's "eyes", mimicking player view. The camera used should be a child of the agent GameObject and have its Camera component disabled.

**Raycast Distance.** The distance (in units) that the agent "looks" across the Navmesh for obstacles. This value should be positive.

**Raycast Height.** The height (in units) at which rays will be casted across the Navmesh.

## PathOS Agent Renderer

Controls the runtime agent UI. Added automatically with the PathOS Agent component. Renders only if the agent is selected and the Renderer component is expanded in the Inspector.

**Show Legend.** If enabled, a legend for the agent's mental map and displayed Gizmos will be shown.

**Show Memory Map.** If enabled, the agent's mental map will be rendered on screen.

**Map Screen Size.** The onscreen size of the mental map (in pixels).

**Show Player View.** If enabled, the view from the agent's player camera will be rendered on screen.

**View Screen Size.** The onscreen size of the player view (in pixels).



## PathOS World Camera

This component can be attached to an overhead camera for ease of viewing agents during playmode.

**Scroll Speed.** How quickly the camera will dolly in and out when the mouse wheel is scrolled.

**Pan Speed.** How quickly the camera will pan around when the mouse is clicked and dragged.

## Data Visualization

### OGLog Manager

Handles recording of data logs for agent behaviour and navigation.

**Enable Logging.** If enabled, data logs will be recorded for all agents in the scene.

**Log Directory.** The directory to which logs should be written.

**Log File Prefix.** How agent files should be named ("*prefix-#.csv*").

**Sample Rate.** How often (per second) agent position should be sampled for logging.

### OGLog Visualizer

Handles visualization of agent logs. See the [Data Recording & Visualization](#) section for an explanation of Inspector settings for visualization. Needs an OGLog Heatmap component attached to a child object in order to render heatmaps.

### OGLog Heatmap

Controls the rendering of heatmap visualizations.