Documentation

1. Requirements

- Unity version 2021.3.21f1, downloaded via the Unity hub (https://unity.com/download)
- python (Download Python | Python.org)
- Pedpy (PedPy: Pedestrian Trajectory Analyzer — PedPy 1.1.3 documentation)

2. Setup new environment

To setup a new environment the first thing to do is open the project in Unity version 2021.3.21f1 (other versions may still work but were not tested). Inside the project open the folder Assets/Scenes, then create a copy of the BluePrint scene, rename the copy to any name you prefer and open it.

In this scene it is possible to create and import the environment in which the experiment will take place, all the static objects like buildings, furnitures, walls, etc. need to be put inside the object Environment, like in image 1. Assets can be found in the Unity Asset Store (https://assetstore.unity.com/)

1 Environment example

[1] Environment example

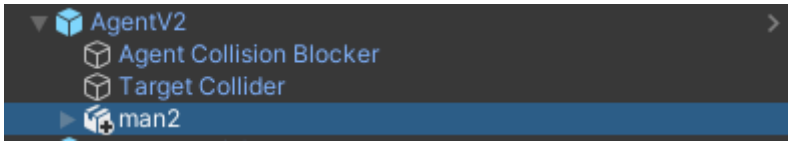
3. Agents setup

After creating the environment with all the needed assets it is time to create all the agents and set them up.

There are two kind of agents: a navmesh agent and a ML-agent, the setup process is similar for both, but a navmesh agent uses a navmesh to navigate the environment, while a ML-agent uses machine learning.

ML-Agents setup

To create a ML-Agent drag the prefab AgentV2 located in Assets/ML_Agents/Utils/AgentV2 into the scene, after creating the object pick an avatar in the folder Assets/ People and drag it inside the created agent, like in figure 2.

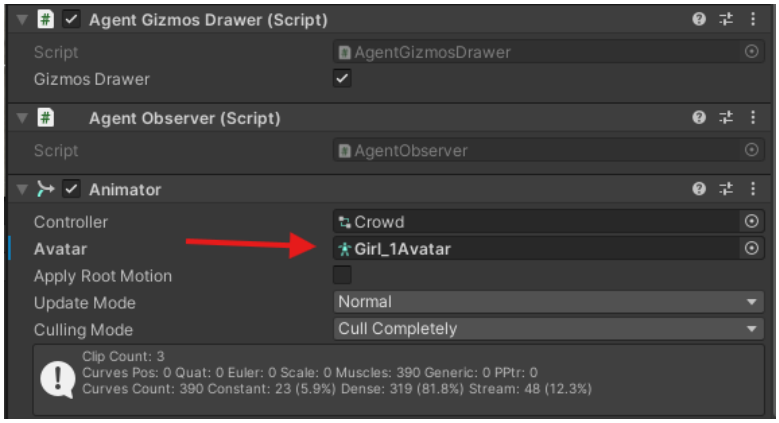


2 Agent person example

After that take the avatar from the same person inside the People folder and put it in the "Avatar" field in the Animator of the newly created agents, like shown in figure 3 and 4

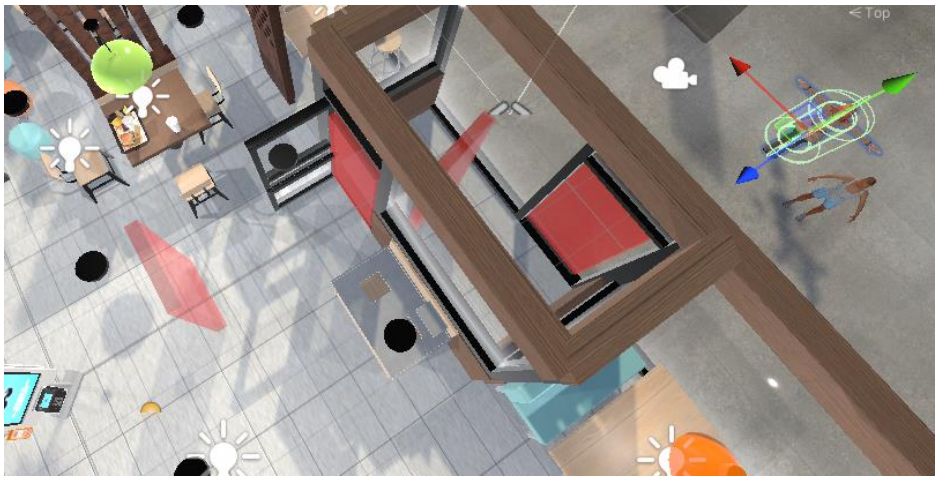
3 Avatar example

[2] Agent avatar example
[3] Agent avatar example



4 Insert avatar into Animator

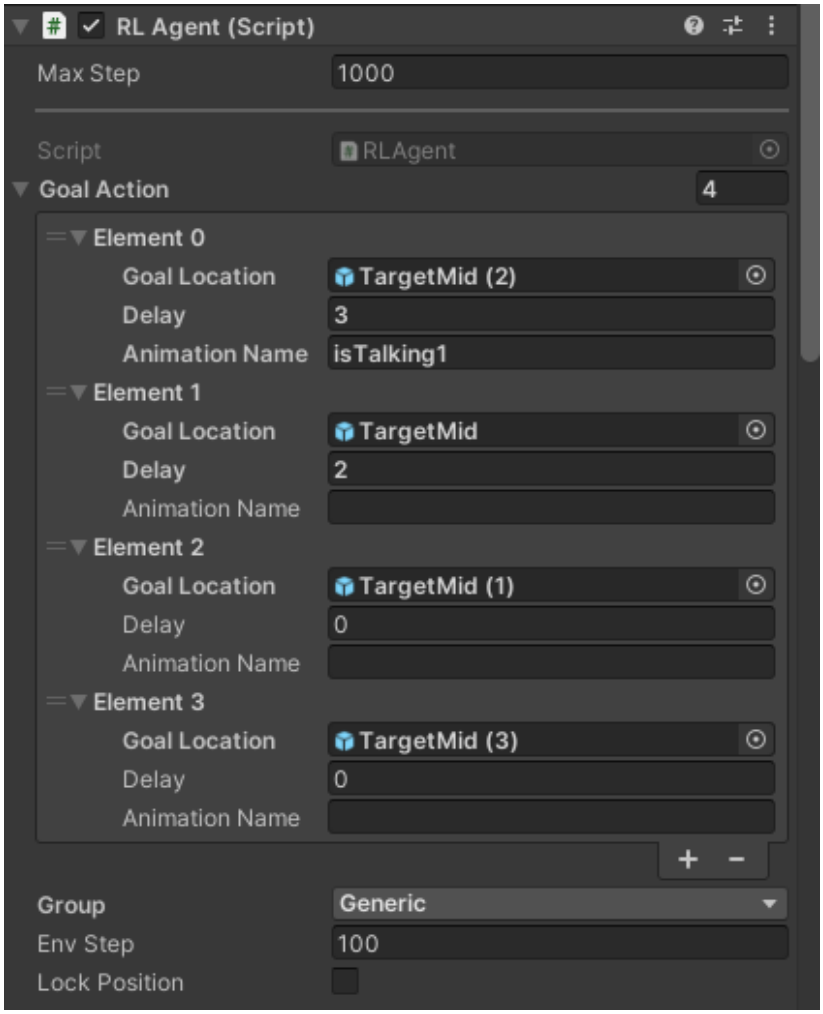
The agent is now created, the next step is to insert into the scene targets for it to follow. To insert targets into the scene take TargetMid from the folder Assets/ML_Agents/Utils/TargetMid and drag it into the scene. To create a path insert as many targets as needed so that at each target the agent can see the next one, like in figure 5 (the agent can 'see' in a similar way humans see: in a 180 degree area in front of it but can't see behind walls or even transparent materials).
To see an example of a set up scene with targets already positioned open the "FFK Sample Scene".



5 Path example

After positioning every target in the path, select the agent and in the RL Agent (Script) section add elements to the Goal Action array and insert the targets created in order from the first to the last in the path inside the Goal Location of each element, like in figure 6.

[4] Agent avatar example
[5] Agent avatar example



6 Targets array example

Optionally it is possible to set a delay at each Target that dictates how many seconds the agent has to wait in the Target's location. To do this simply change the Delay value in the element of the corresponding Target.

Finally, it is also possible to set the animation that the agent needs to perform while waiting for the delay. For this type the animation name inside the Animation Name field in the corresponding Target's element. For a list of the available animations look into the "Crowd" animator located in Assets/People/Crowd, on the left of the screen there is a list of all the available animation (exept "Speed"which is a variable used to define the speed of the animation). Be sure to check for typing errors if the animation doesn't start.
The default animation names are:

- "isIdle"
- "isWalking"
- "isRunning"

[6] Agent avatar example

- “isTalking1”
- “isTalking2”
- “isPointing”

NavMesh Agents setup

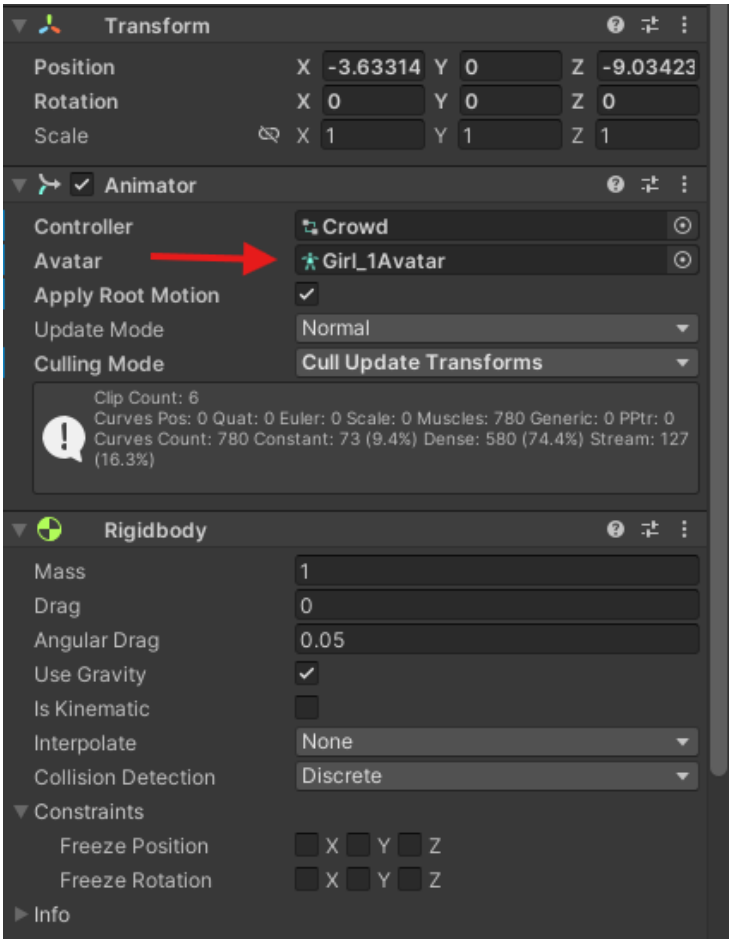
To create a NavMesh Agent drag the prefab NavMeshAgent located in Assets/NavMesh_Agents/ NavMeshAgent into the scene. After creating the object pick an avatar in the folder Assets/ People and drag it inside the created agent, like in figure 7.

7 Agent person example

After that take the avatar from the same person inside the People folder and put it in the “Avatar” field in the Animator of the newly created agents, like shown in figure 3 and 4

8 Avatar example

[7] Agent avatar example
[8] Agent avatar example

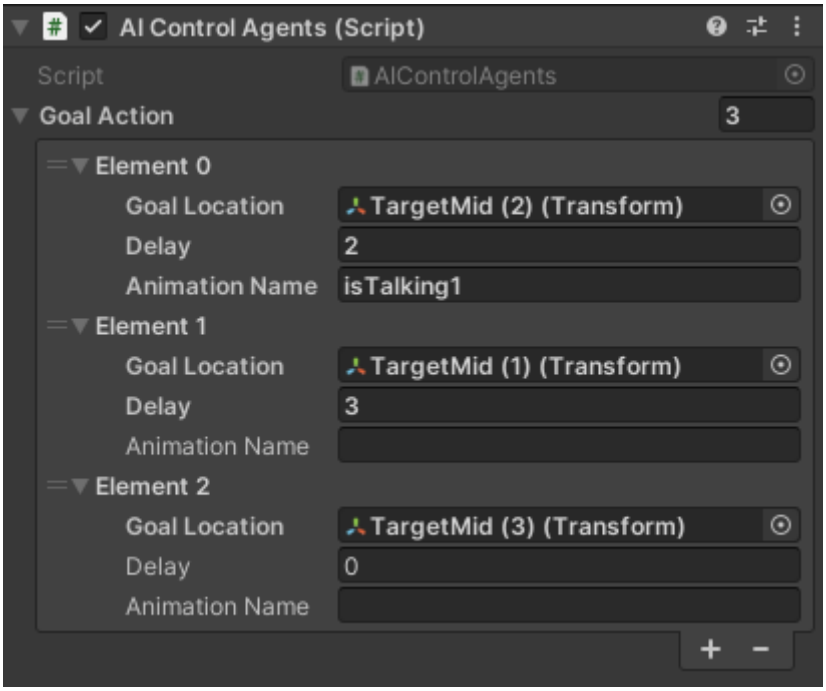


9 Insert avatar into Animator

The agent is now created, the next step is to insert into the scene waypoints for it to follow, the suggested way is to either use the waypoint prefab, located in Assets/NavMesh_Agents/ Waypoint or use a Target object if the NavMesh Agent need to follow the same path as other ML-Agents. To create a path for the agent to follow; insert as many of these objects as needed along the path, this time it is not required for the agent to see the waypoints.

After positioning every target in the path select the agent and in the AI Control Agents (Script) section add elements to the Goal Action array and insert the targets created in order from the first to the last in the path in the Goal Locatio of each point, like in figure 10.

[9] Agent avatar example



10 Goal Action array example

Optionally it is possible to set a delay at each waypoint that dictates how many seconds the agent has to wait in the waypoint's location. To do this simply change the Delay value in the element of the corresponding waypoint.

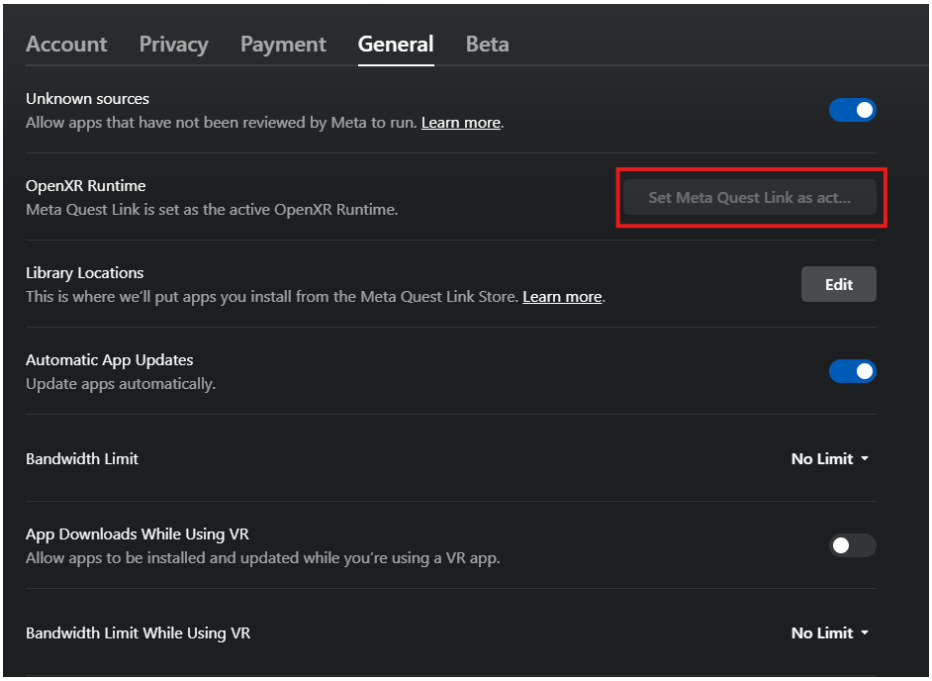
Finally, it is also possible to set the animation that the agent needs to perform while waiting for the delay. For this change the Animation Name string inside the corresponding waypoint's element to the name of the desired animation. For a list of the available animations look into the "Crowd" animator located in Assets/People/Crowd, on the left of the screen there is a list of all the available animation (exept "Speed"which is a variable used to define the speed of the animation).

4. Experiment execution

To execute the experiment the first thing needed is to setup the physical environment so there is enough space for the user to move.

After setting up the physical environment the only thing needed is to connect the VR headset to Unity. To achieve this for the Meta Quest headset open the Meta Quest Link software on the PC running Unity, enable OpenXR Runtime inside General settings, as shown in figure 11.

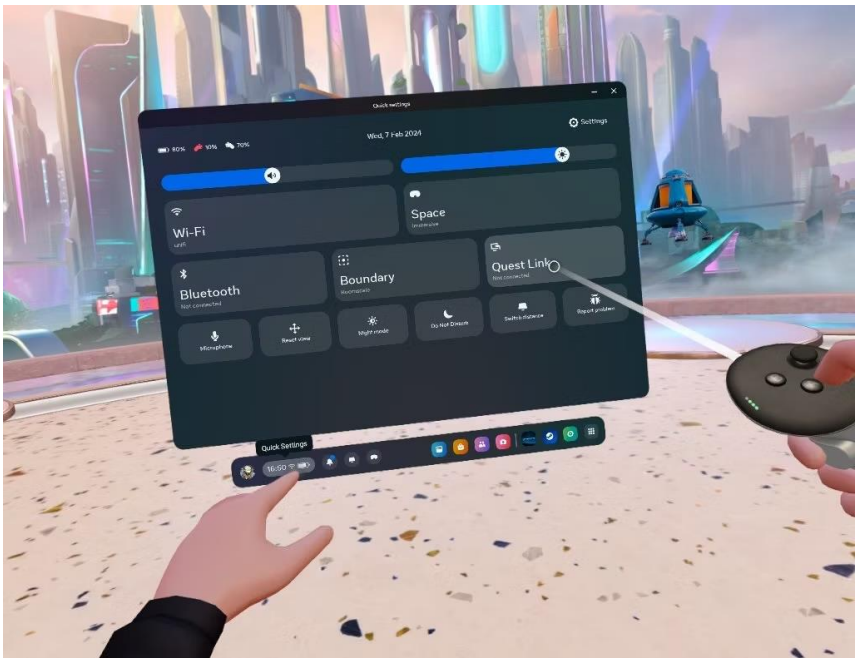
[10] Agent avatar example



11 OpenXR Runtime setting

Then if not already done connect the headset with a USB cable to the PC; then click on the Devices menu on the left and add the headset.

After connecting the headset to the PC put on the headset and, inside the headset, in the quick settings (accessible by clicking on the time on the bottom left of the menu) click on the Quest Link button and the connect to the PC

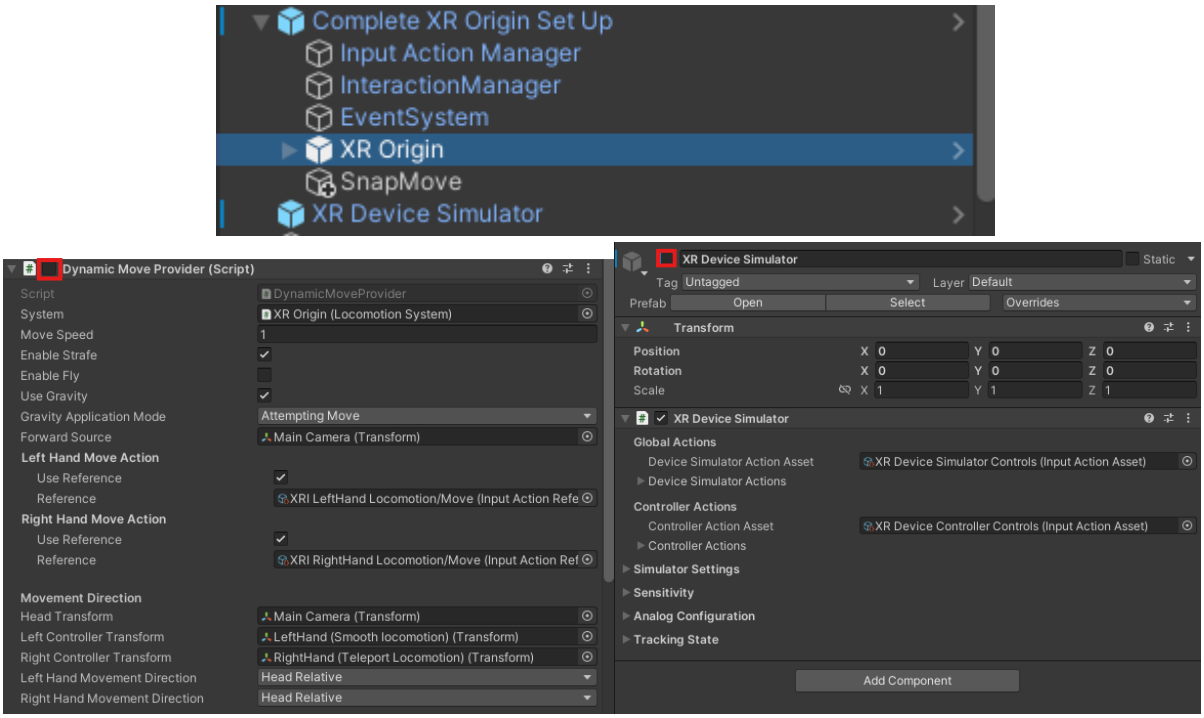


[11] Environment example

Once inside the quest link environment start the experiment by starting the scene, which should start the environment inside the headset.

To move inside the environment, use the joystick on the left controller and point it forward to move by approximately one meter towards where the user is looking. To turn around it is possible both to physically turn around in real life or to use the joystick of the right controller to turn left and right by pointing it left and right and to turn 180° by pointing it backwards.

For moving inside the environment without the head mounted display, when testing the environment inside Unity, just enable the " XR Device Simulator " object and the "Dynamic Move Provider" component inside the "XR Origin" object located inside the "Complete XR Origin Set Up" object. To enable them click the checkmark beside its name in the inspector.



After enabling both, start the environment, press tab once to select the left controller and the keys w a s d to move forwards, left, backwards and right respectively. To look around press tab another two times to switch back to the head control and move the mouse to rotate the head.

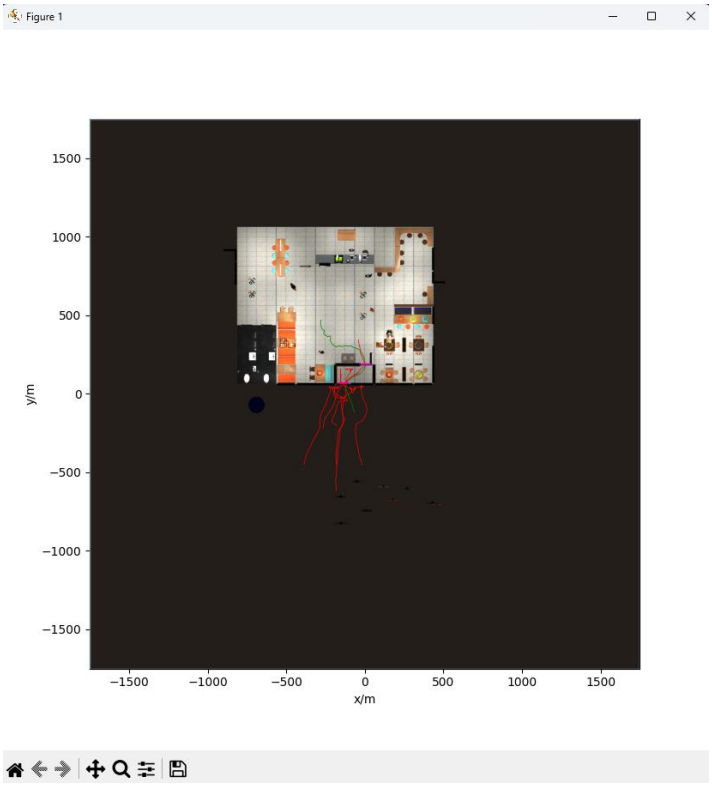
Remember to disable both components before switching back to using the VR headset.

[12] Environment example

5. PedPy analysis

After executing the experiment, it is possible to use PedPy to plot and analyse the trajectory data generated from the agents and user during the experiment. For this purpose, it is necessary to drag the "BackgroundMap", "PedPyStats" and "UserStats" files to the PedPy folder (the one provided with the unity project). After this open PowerShell and travel to the PedPy folder and execute the command "python .\pedpy.py". After a couple of seconds, a small window will open with the plot resulting from the analysis.

The default script present in pedpy.py creates a plot that shows the agents trajectory in the environment in red and the user's in green; other types of analysis can be done by editing the pedpy.py script following the PedPy User Guide — PedPy 1.1.3 documentation.



13 Example of the default plot

6. Troubleshooting

- If ML-Agents instead of following the path stand still and keep turning around and going in random direction, try moving the next target closer to the agent or create an additional target and check if it is visible from the agent's position.

[13] Environment example

- If ML-Agents keep getting stuck on objects or walls when turning, try moving targets around to create a better path further away from corners if possible.