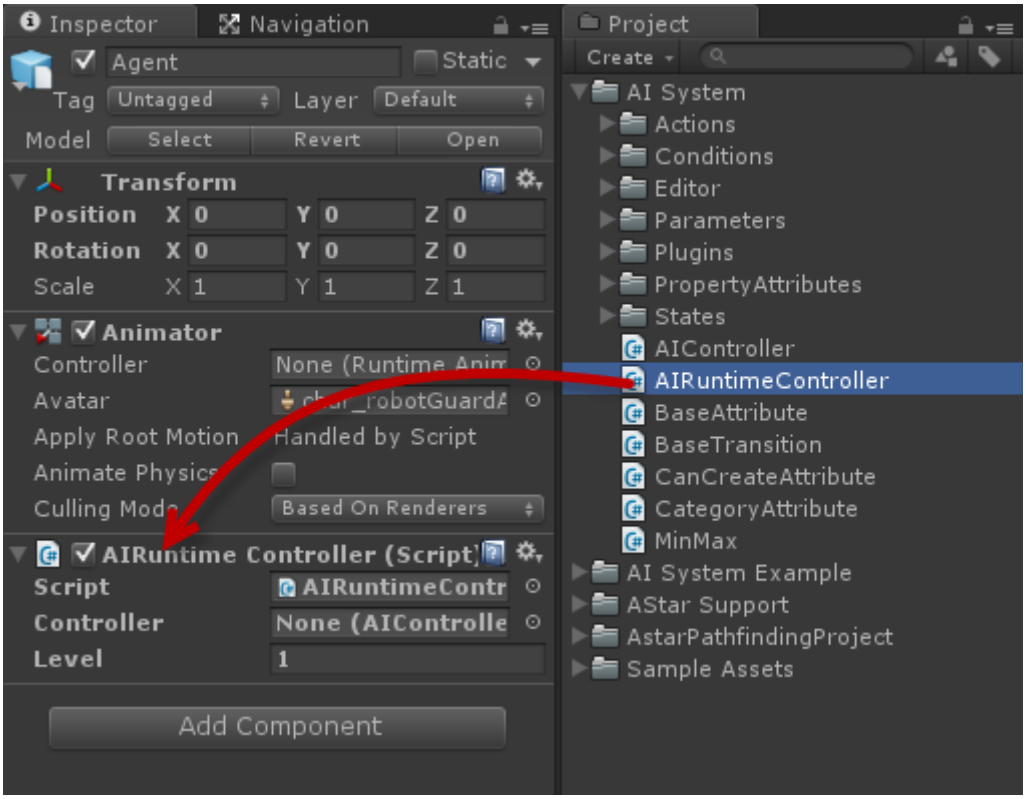
AI System for Mecanim Documentation

How to setup your custom agent?

To setup the agent you need to drag and drop your prefab or model into the scene and add the AIRuntimeController.cs script on it, or you can use the Add Component button from the inspector.



You can create and add any „Animator Controller“ to the animator component. Please follow the unity tutorials to get started on this.

Now it depends which pathfinding solution you would like to use.

Using the default unity NavMesh pathfinding:

The navmesh requires the NavMeshAgent component on your agent. To add this component, select your agent in the Hierarchy, go into the Component menu at the top of unity, navigate to the Navigation submenu and click on the Nav Mesh Agent button to.

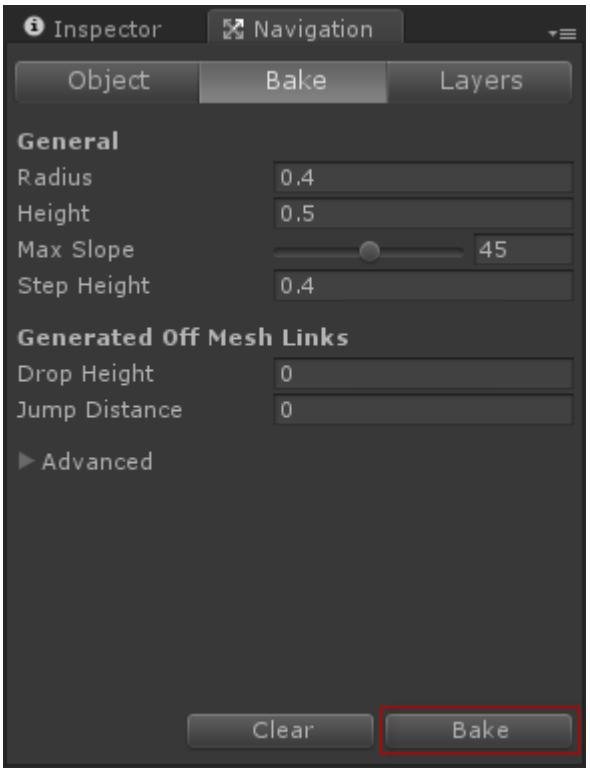
Component ➡ Navigation ➡ Nav Mesh Agent

Adjust the radius and height of the NavMeshAgent so it fits your prefab. You do not need to setup the speed nor the angular speed, because this be changed at runtime anyway inside the AIRuntimeController.

The last step is to bake the navigation mesh of your scene. Select for this your enviroment (Floor, obstacles and other things you want the agent to avoid) and mark them as static at the top of the inspector. Now go into the window menu at the top and select „Navigation“ from the dropdown menu.

Window ➡ Navigation

In the inspector you will see the navigation baking tools of unity. You can adjust the settings and click afterwards on bake, don’t worry about the settings yet, you can change them afterwards.



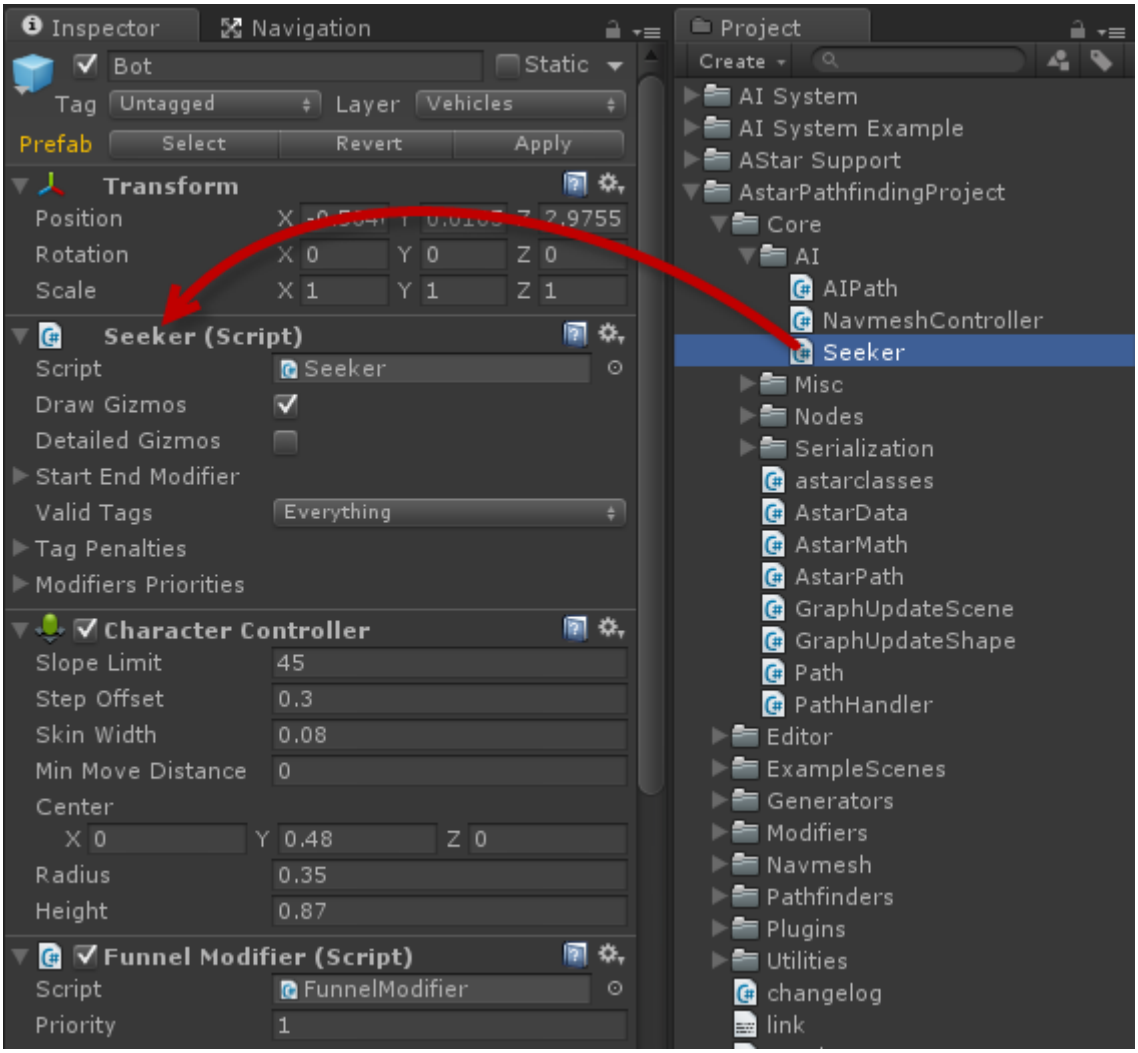
Using the AStar supported pathfinding solution by Arongranberg:

The example states require the CharacterController component. To add this component, select your agent in the Hierarchy view,

go into the Component unity menu then into Physics and click on the Character Controller button.

Component ➡ Physics ➡ Character Controller

You need also to attach the Seeker.cs and FunnelModifier.cs script from the A* Pathfinding solution package.

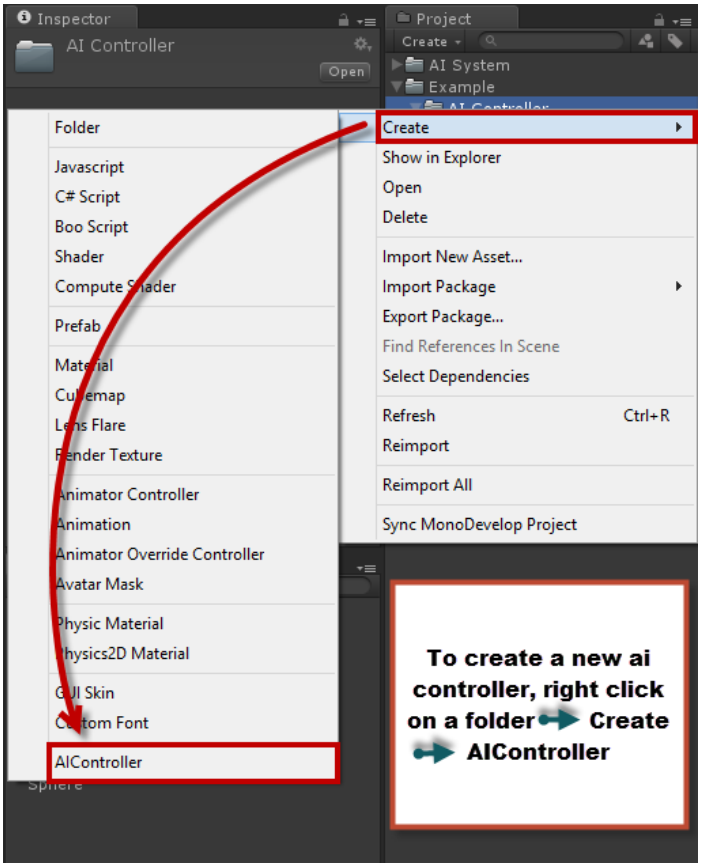


Please follow the instructions on how to setup those components from the original writer of those scripts and how to create the grid for the pathfinding. You can find more information on his website.

You may have noticed that the AIRuntimeController component has two public fields. The next section will describe, how you can

create the AIController. The AIController is the core holder of your states and settings.

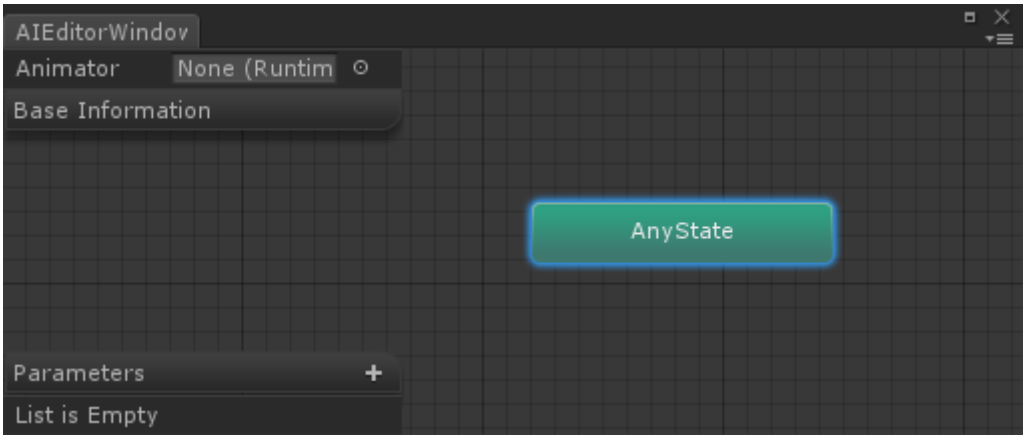
How to create a new AIController?



The creation of a new AIController works the same way as you create an Animator Controller.

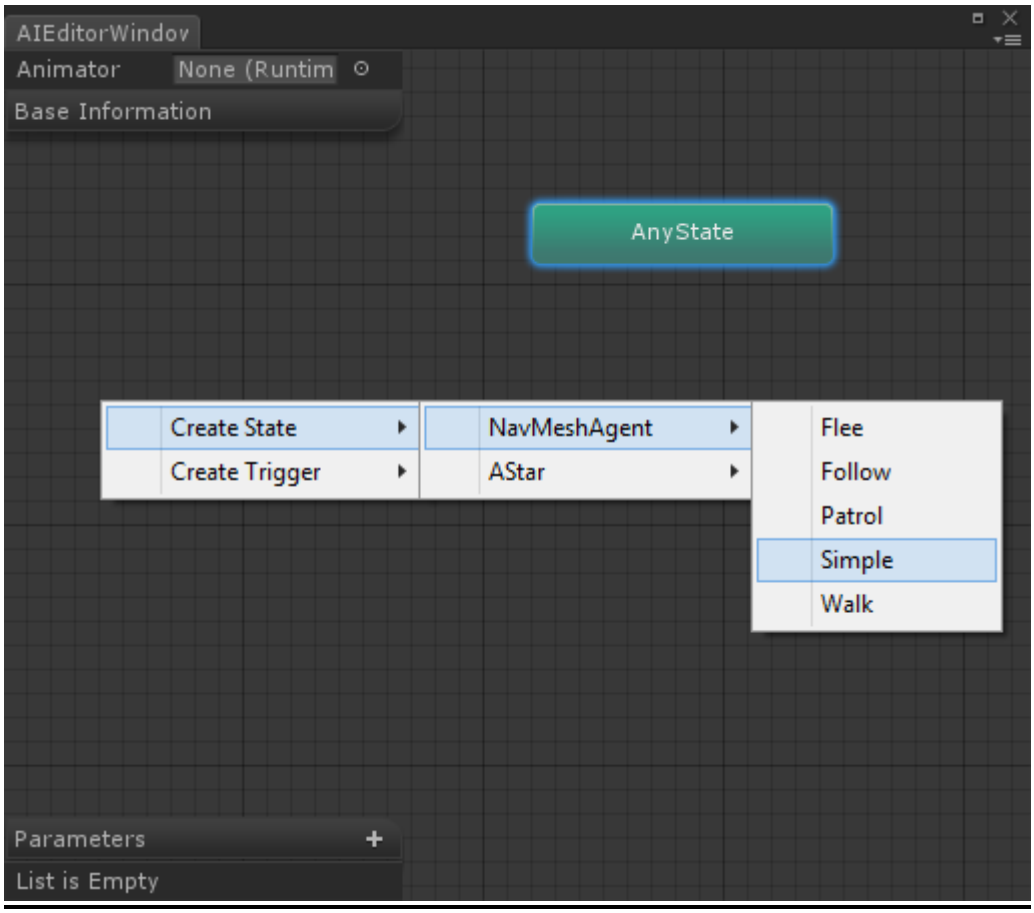
Just right click in the Project window, go to Create AIController.

Now that your AIController is created you can double click it and a new window will open, where you can create the whole ai for your agent using states, actions and conditions. The just opened window should look like this:



Now that your AIController is created drag and drop the created asset into the controller field of you agent in the AIRuntimeController public field.

How to create new states?



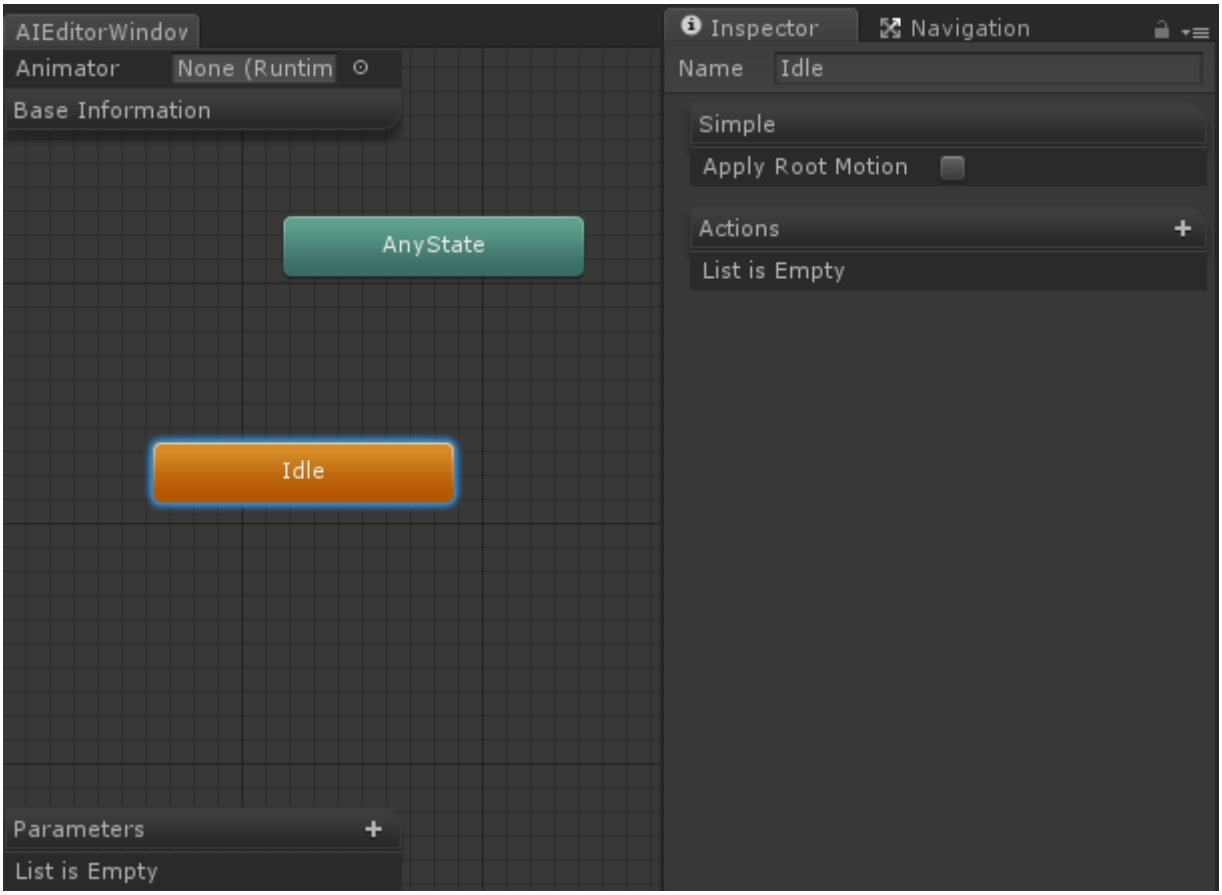
To create new states just click on an empty area in the graph, a dropdown menu will popup, where you can select and create states based on your pathfinding solution. Please don't worry as more states will come over time. I am planning to include states for 2.5D games and also 2D sprite based games. In later updates you will also notice some states based on rigidbody for a spaceship or fly game.

Now that you know how to create states lets take a look at what settings you can do on each of this states.

We will start with the „Simple" state.

Simple State:

When you create the state and select the state, your ai window and the inspector should look like this:

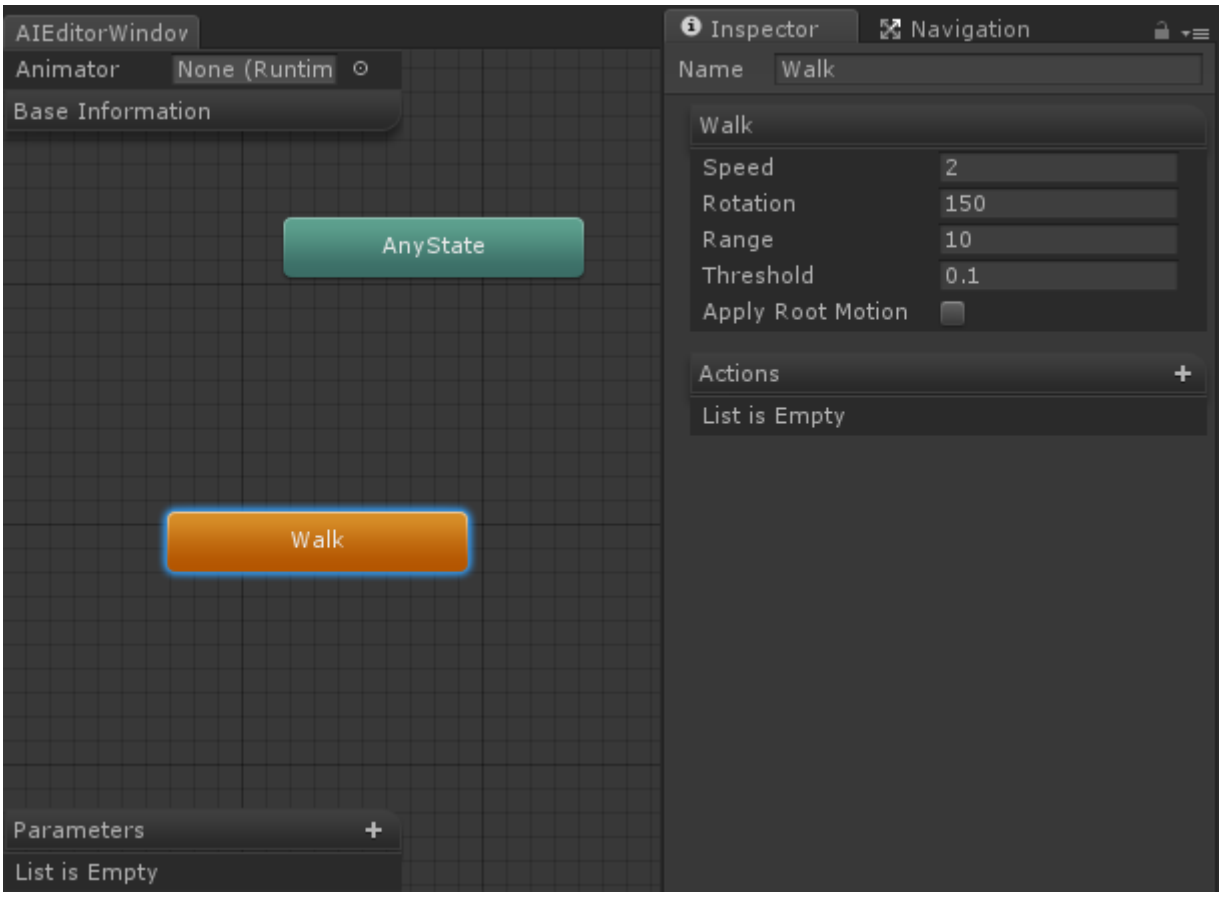


If it does not look like this, but you still see the titles of the lists „Simple“ and „Actions“, just click on the title area and they will foldout.

The simple state is the state that has mostly no settings it can be used as „Idle“ or „Attack“, this means no movement is done in this state, but you can still add all actions to play for example an animation or to send a message…

Apply Root Motion means that the root motion data that is on the animation will be applied to your agent and if you are using the navmesh, this root motion data will be applied tot he velocity oft he navmesh.

Walk State:



The walk state is a state for random walking in an area with the following properties you can do.

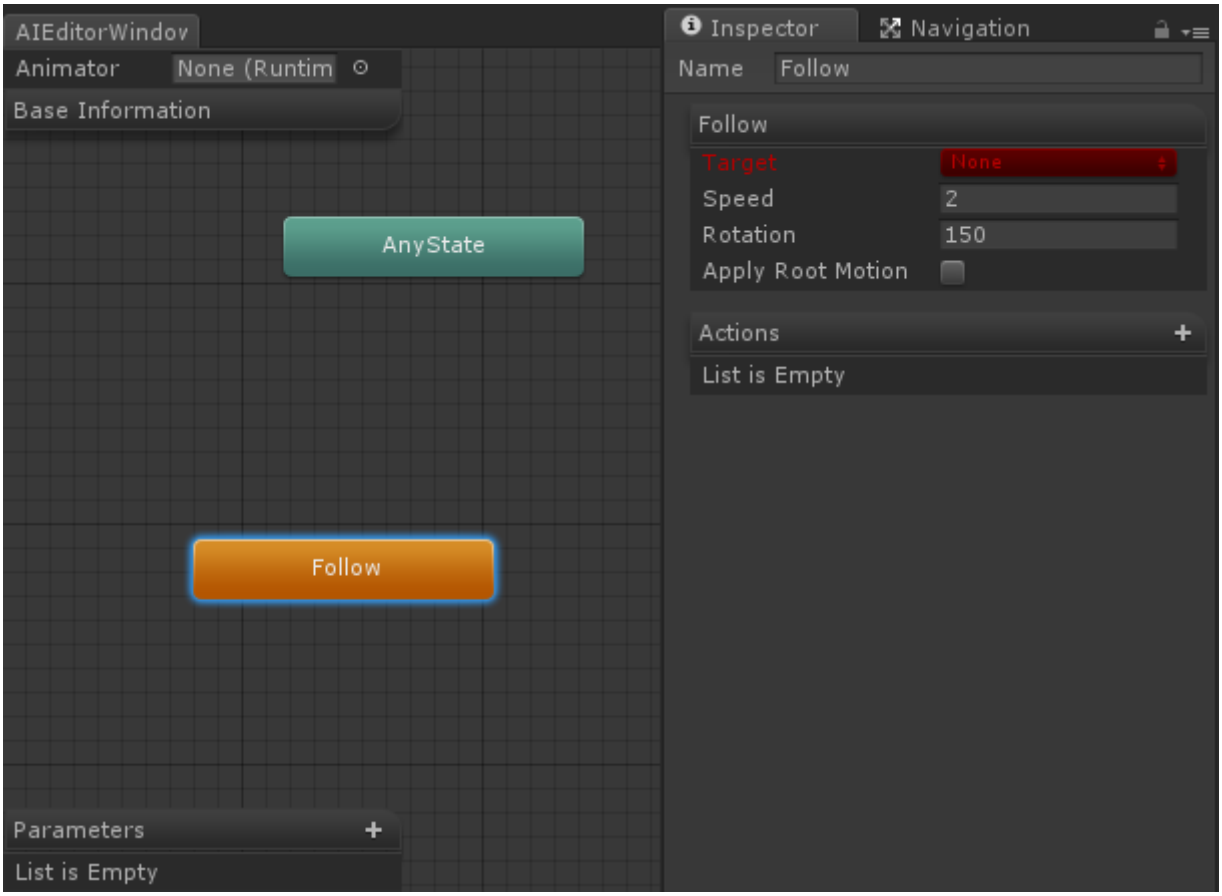
Speed: *This ist he property where you can setup how fast you agent will walk arround.*

Rotation: *How fast shoud the agent rotate to his destination.*

Range: *The agent will randomly pick up a vector point in this range and walk towards it. The destination is calculated based on the initial position of the agent.*

Threshold: *This is the distance at which the agent will pick up a new destination. It is unlikely that floats will be exacctly equal, unless they are explicitly set that way, because of this behaviour we need to setup the threshold.*

Follow State:



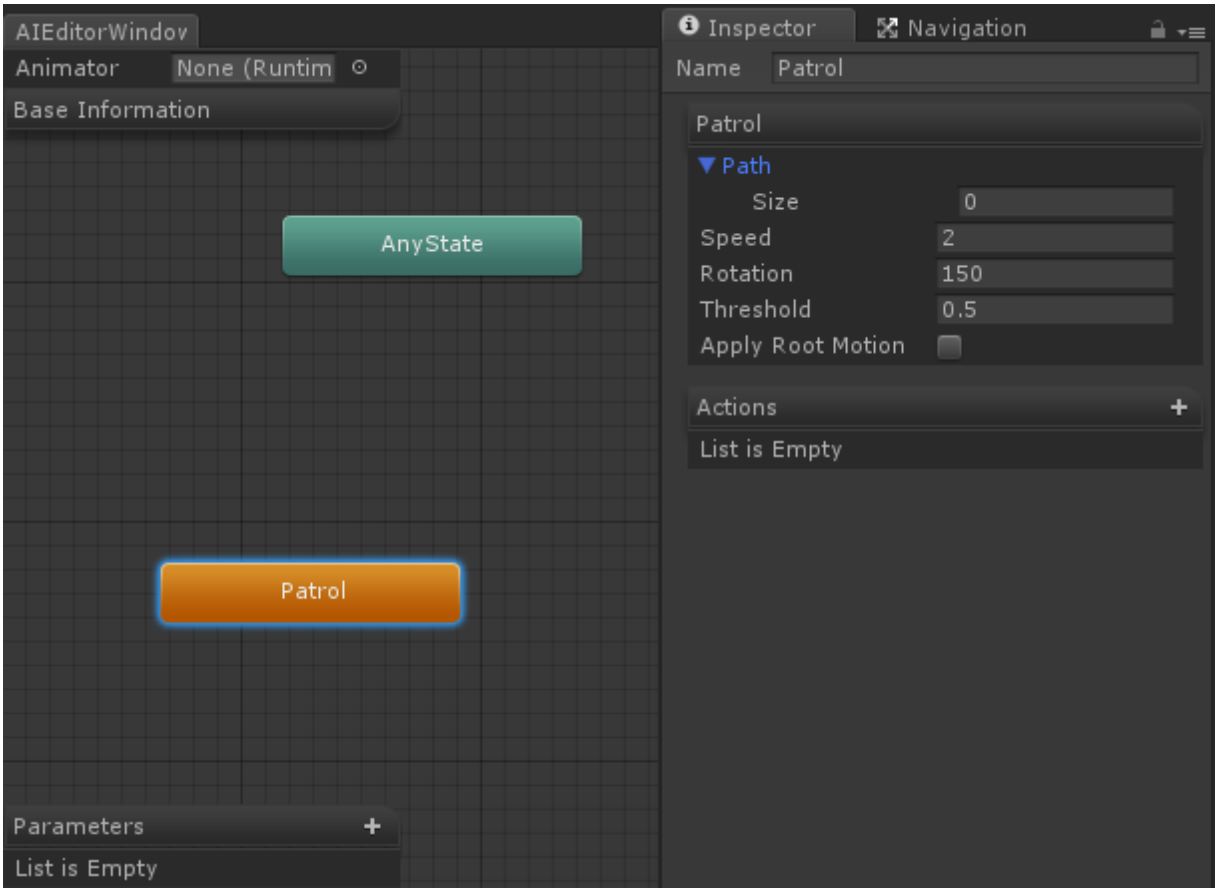
Target: *The target is marked as red because there are no global parameters created. The target requires a global game object parameter, which the agent should follow. How to create global parameters and set them at runtime will be exmplained later.*

Other settings are the same as in the walk state, so take a look at the description there.

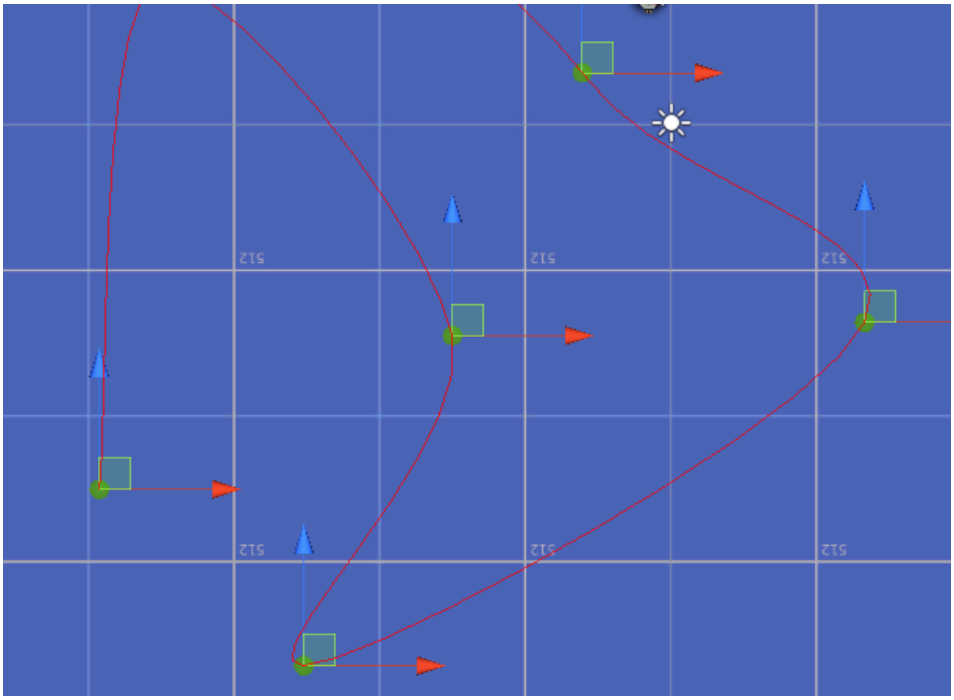
The „Flee“ state has also the same parameters and you should look at previous state descriptions if you are unsure, how to setup those.

The next new state is the „Patrol“ state.

Patrol State:

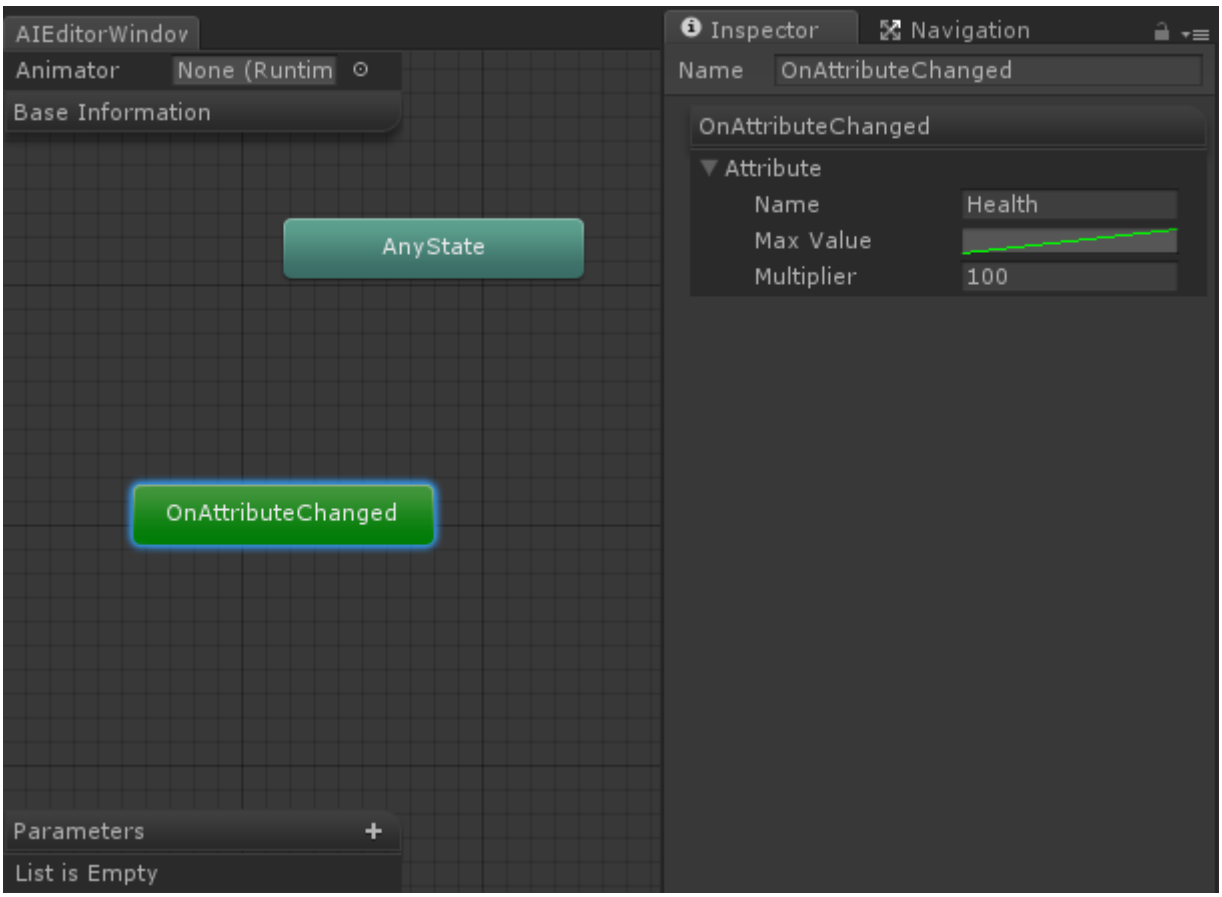


Path: *The path contains points, which the agent will follow. You can create those points by going into the scene tab and click on your enviroment floor. Deselect the path state to disable the creation of those points.*



Beside states there are triggers that you can create the same way as state. What are triggers? Triggers are states from which you can transition based on global interaction. For example if the player hits your agent a tigger is called and you can maybe create a „Get Hit“ effect.

OnAttributeChanged:



In this trigger you have to create an attribute and if this atttribute is changed it will be called and transition to another state. How to transition to other states, we will see in the next section.

But before we go to the next section lets take a look at which settings you can do in the Attribute.

Name: The name of the attribute, which should be unique for each attribute in one AIController, because the attribute is identified by his name. To get the attribute from code you can call it from the reference of the AIRuntimeController component

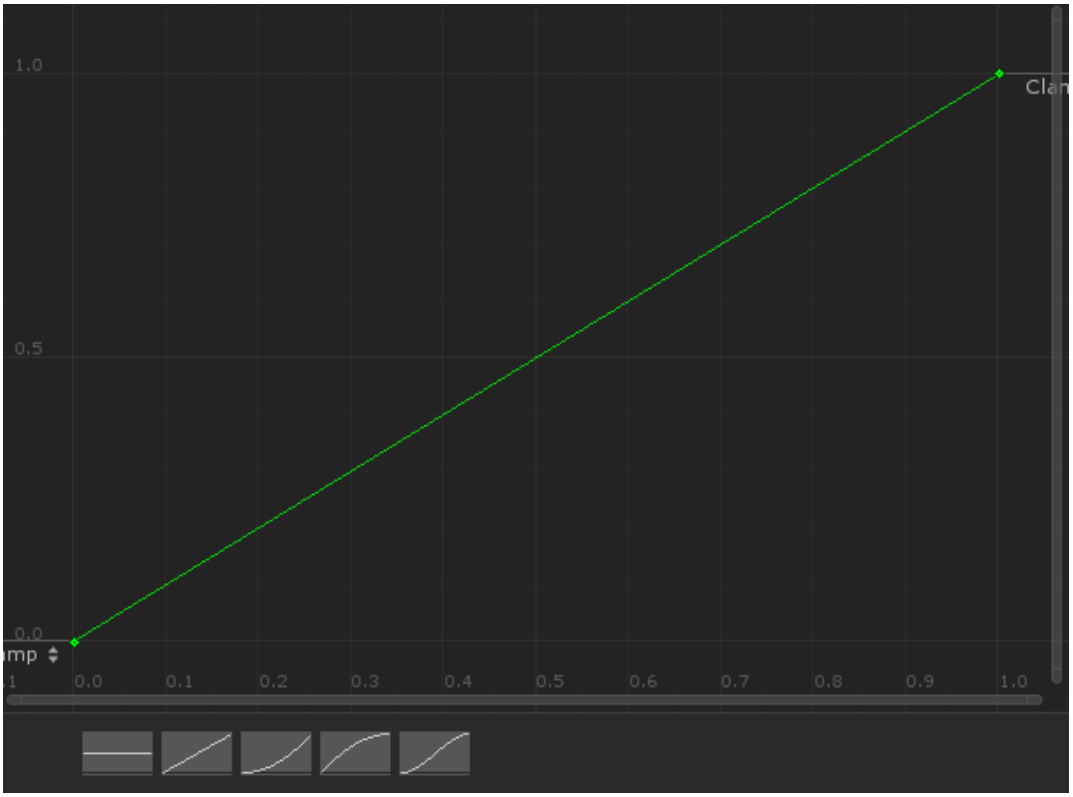
controller.GetAttribute(string name);

```
//This method is received from the animation event on the player
private void SendDamage(int damage){
    RaycastHit hit;
    //Send a ray from players transform with his forward direction
    if (Physics.Raycast (transform.position + Vector3.up, transform.forward, out hit, 2,sendDamageMask)) {
        //We hit something so lets check if it is an enemy with an AIController
        AIRuntimeController behaviour = hit.transform.GetComponent<AIRuntimeController> ();
        //Check if we hit the enemy
        if (behaviour) {
            //Yes we hit the enemy, so lets get the health attribute
            BaseAttribute defenderAttribute = behaviour.GetAttribute ("Health");
            //Is the health attribute defined in the AI Editor?
            if (defenderAttribute != null) {
                //Yes the attribute is defined in the AI Editor,
                //so lets consume the damage from it.
                defenderAttribute.Consume (damage);
            }
        }
    }
}
```

If you just wish to reduce it, you can send a message to the AIRuntimeController holder(agent).

```
//This method is received from the animation event on the player
private void SendDamage(int damage){
    RaycastHit hit;
    //Send a ray from players transform with his forward direction
    if (Physics.Raycast (transform.position + Vector3.up, transform.forward, out hit, 2,sendDamageMask)) {
        hit.transform.SendMessage("ApplyAttributeDamage",new object[]{"Health",damage});
    }
}
```

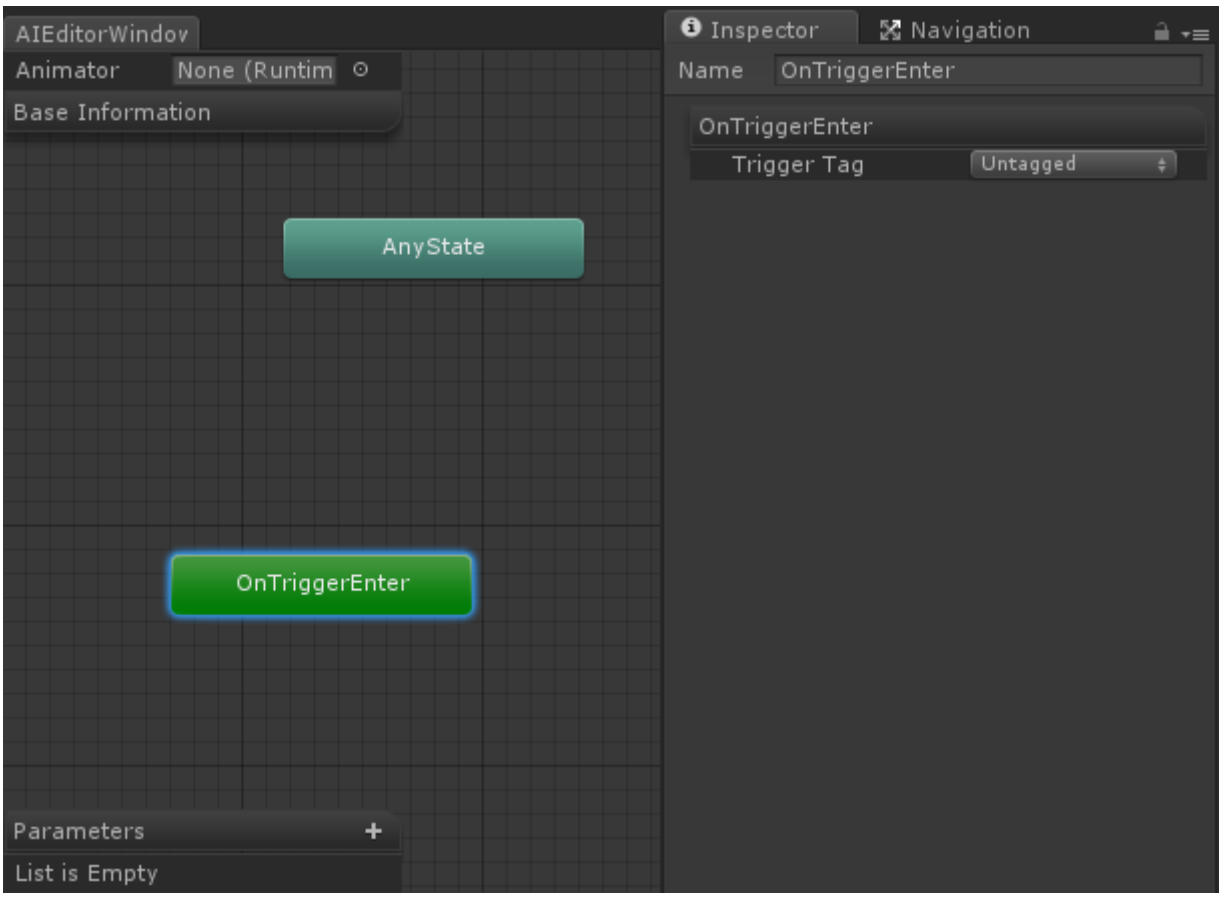
Max Value: *The maximum value of the attribute is calculated based on the level of your agent. Maybe you can remember that the AIRuntimeController has a second field „Level“.*



The time axis is the level x 0.01, the attribute value is the y axis value multiplied by the „Multiplier“ value.

Example: If we set the level to 20 and the „Multiplier“ value to 100 the maximum value will be 20.

OnTriggerEnter:

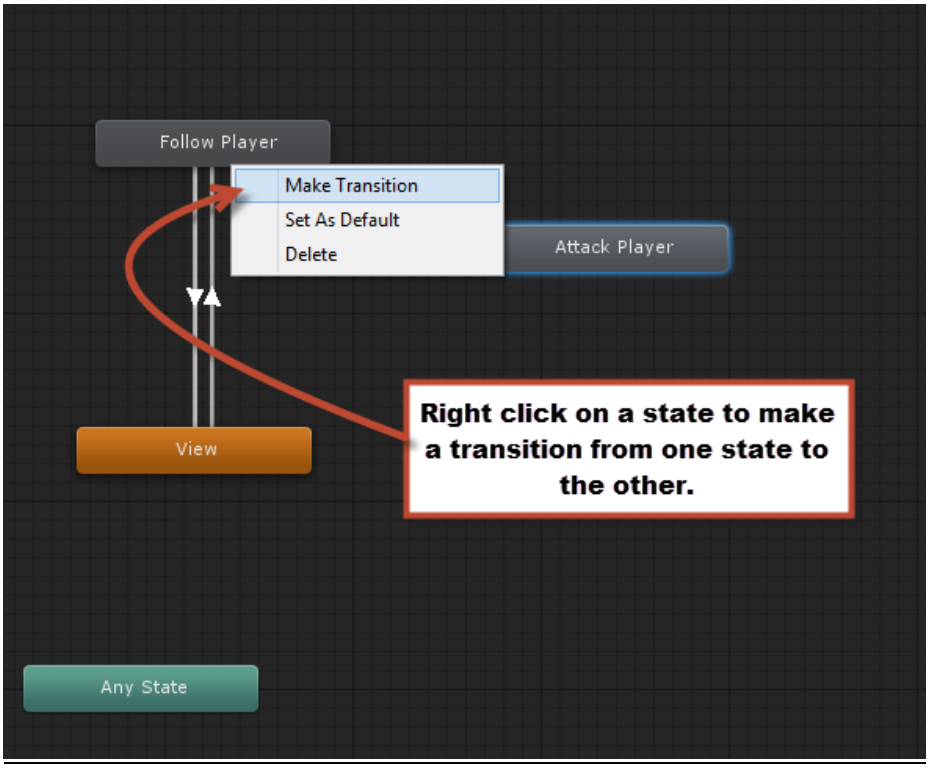


OnTriggerEnter is a simple trigger that is called when you agent walks inside a trigger. Please note that your agent requires a collider and a rigidbody component and that the trigger objects collider is set to isTrigger true.

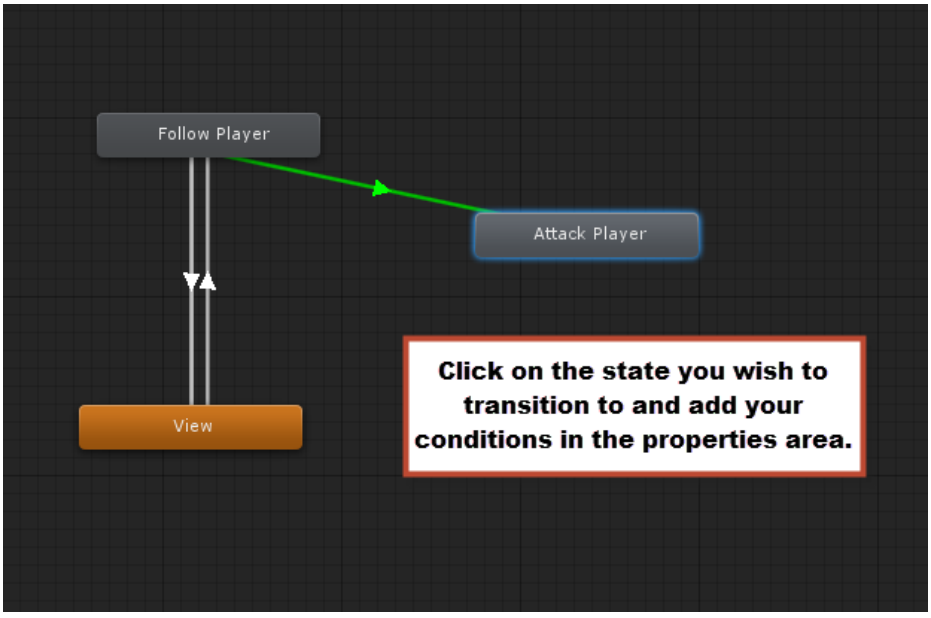
Trigger Tag: *You can specify on which tag(other game objecs tag) this state should be called. For example, if you set the tag to „Untagged“, but the other game objects tag is set to “Obstacles“, this trigger will not be called.*

How to link states and make transitions?

To make a transition from one state to another, you can right click on a state where you want to transition from, choose „Make Transition“ from the menu.

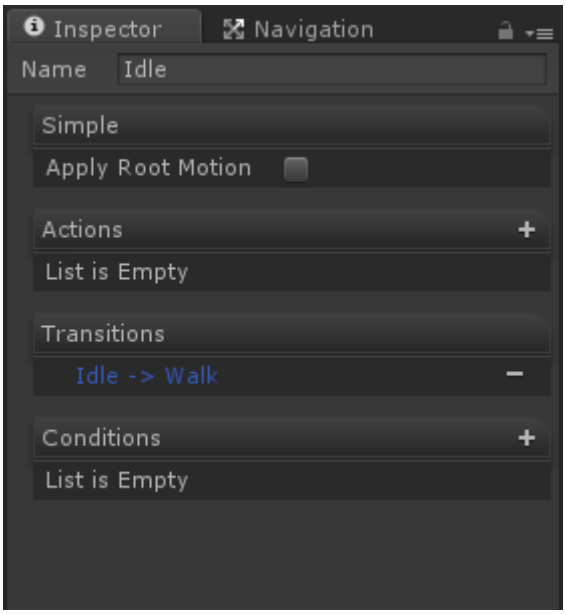


Afterwards click on a state you want to transition to.



Conditions

After you have created the transition select the state you made a transition from. You may have noticed that your inspector changed and you can create conditions now clicking on the plus icon.



How to add „and“ / „or“ conditions?

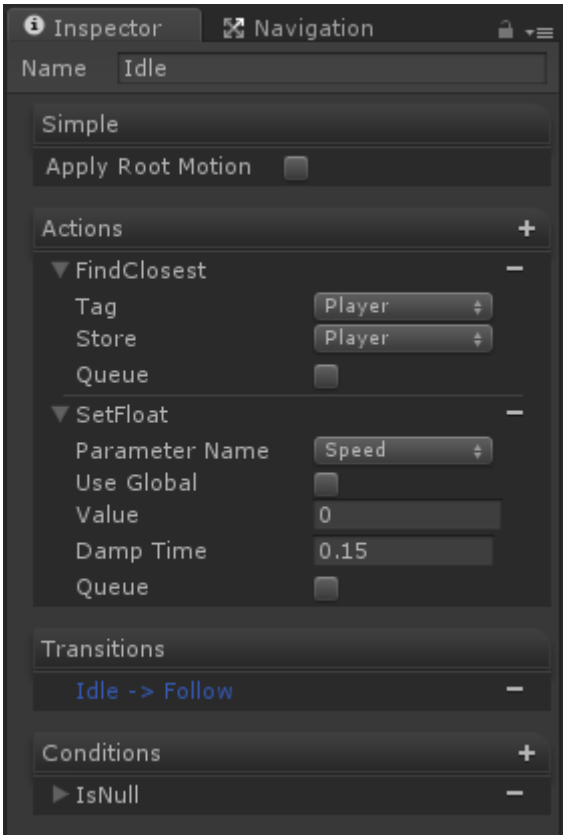
To add an „and“ condition just click on the plus button in the inspector. For an „or“ conditions you need to make a new transition to the same state.

Actions

To create new actions just click on the plus icon in the inspector in the actions area. From there you can select the action you would like to use.

I will pick up two actions and describe them. The first one ist he FindClosest, which is located at the GameObjects sub menu.

The second one is the SetFloat animator action.



FindClosest: *This action will search for the closest game object that is tagged as „Player“ in this example. Furthermore it will store the result into the global parameter with the same name. You can run this action in a queue by setting this field to true. This means that if you use a WaitForSeconds action, which is automatically set to queue the actions and will run one after another from the queue. This gives you an incredible control over your actions.*

SetFloat: *This action sets the float value of your animator controller. In this example i am setting the „Speed“ to zero with a damp time of 0.15. You can also choose the value from a global variable by enabling „Use Global“.*

Contact Information

Konstantin Janson

Email: konstantin.janson@freenet.de

Skype: tharon211

Unity Forum: Zerano

Credits

The models included in this package are from Mr. Necturus. If you like them check out his site.

Furthermore i want to thank Kalamona who lets me use some of her particle systems. So if you like them search for the key word „Kalamona“ in the asset store and get more particle system for a very great price.

Video Tutorials

- Basic Overview
- Setup your agent
- Create a follower controller
- Extend the follower controller to attack controller
- How to add LineOfSight
- 2D Sprite Setup