# Images example

User makes a new script called “AITest” and inherits from the “UtilityAI” class.

A picture containing text, screenshot, font, software

Description automatically generated

In the inspector they are presented with the instance generator. They can add as many behaviours they’d like.

A screenshot of a computer

Description automatically generated with medium confidence

After adding the behaviours they’d like to generate, they click the Generate AI Instance button.

A screenshot of a computer

Description automatically generated with medium confidence

This will automatically create a new script with the name of their script (“AITest” in this case) prefixed with “UtilityAI\_”. Their script will now inherit from this new script and a “IUtilityAIMethods” interface which will prompt them to add the AIAwake, AIStart, and AIUpdate methods.

A screen shot of a computer program

Description automatically generated with medium confidence

Now if they look at the inspector again, they will be presented with all their chosen behaviours as serialised fields. (These can be updated, removed, or added to at anytime by regenerating)

A screenshot of a computer

Description automatically generated with medium confidence

Each behaviour has many fields which can be edited. These include:

* The evaluation cooldown: The time between each re-evaluation of the behaviour score
* Value range min and max: The range of values which are expected to be returned by the evaluator (E.g. the user expects their run evaluation to return a value from 25-80. This will be remapped behind the scenes to a value between 0-1.)
* The evaluator delegate method: A method which returns a float value of a behaviour’s score (E.g. a value from 25-80 on how much an enemy wants to run away from the player)
* A timer duration: A float which is used by a timer condition to deactivate a behaviour after a certain time (This condition will be an inbuilt method of every UtilityAI which can be added in the Interrupt Conditions – behaviours do not require a timer condition)
* Interrupt Conditions: A list of methods which return true or false to whether a certain behaviour should end. If any condition becomes true, the behaviour can end (e.g. an enemy has a run behaviour and a tiredness condition, and if the enemy becomes too tired it will look to do a new behaviour)
* When Active: A list of methods to be called every frame while the behaviour is active.
* On Start: A list of methods to be called when a behaviour first begins.
* On End: A list of methods to be called when a behaviour ends.

A screenshot of a computer

Description automatically generated with medium confidence

The AI system also has a behaviour selector which is a method that takes in an array of floats representing behaviour scores, and returns an index of any of those floats to be the new chosen behaviour. The UtilityAI class will have a few inbuilt methods or the user can make their own.



Create custom behaviour selector in code

A screenshot of a computer

Description automatically generated with low confidence

Find it in inspector

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

Or set it in the code

A screenshot of a computer program

Description automatically generated with low confidence

Or use the default ones supplied

A screenshot of a computer

Description automatically generated