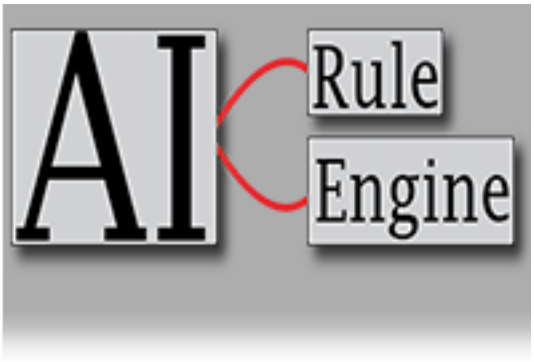


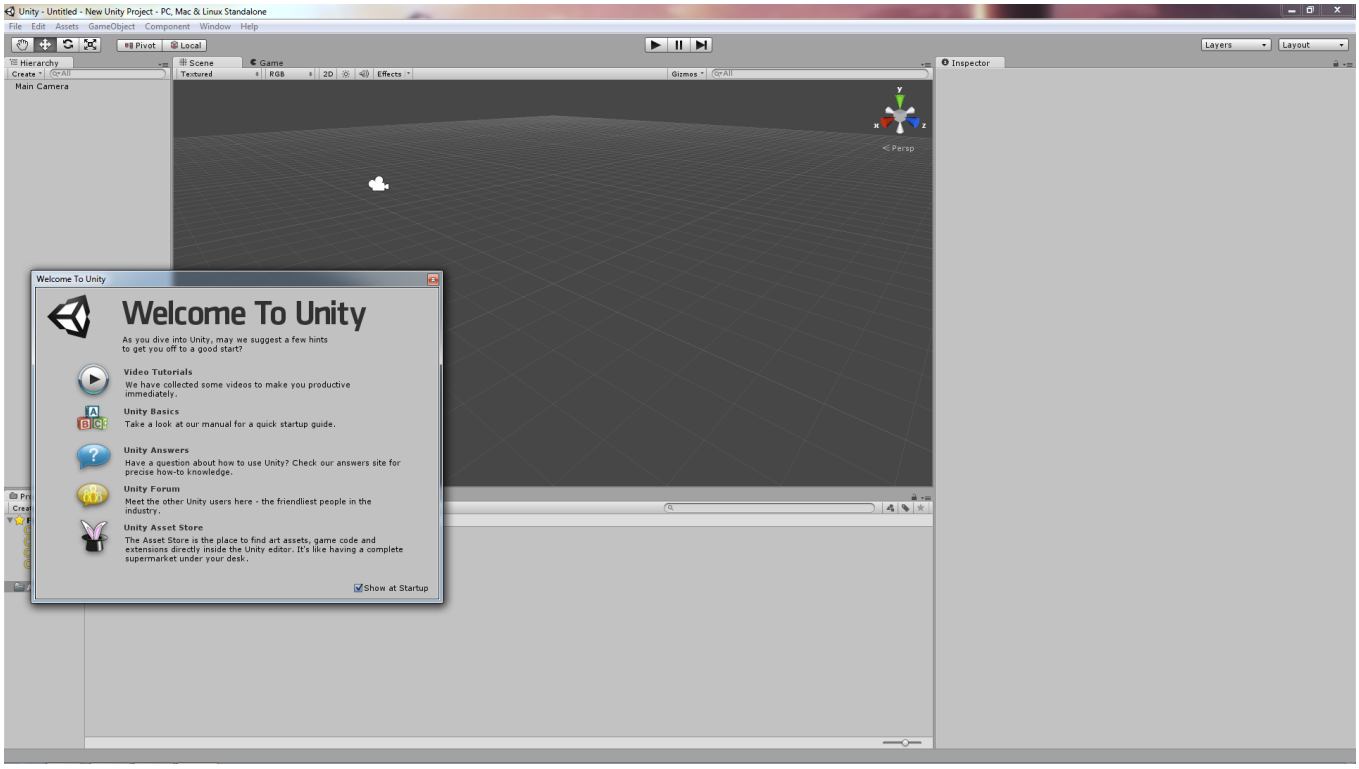
AiRuleEngine Quick Start Guide

http://www.airuleengine.com
VgpUnityRulebase@gmail.com

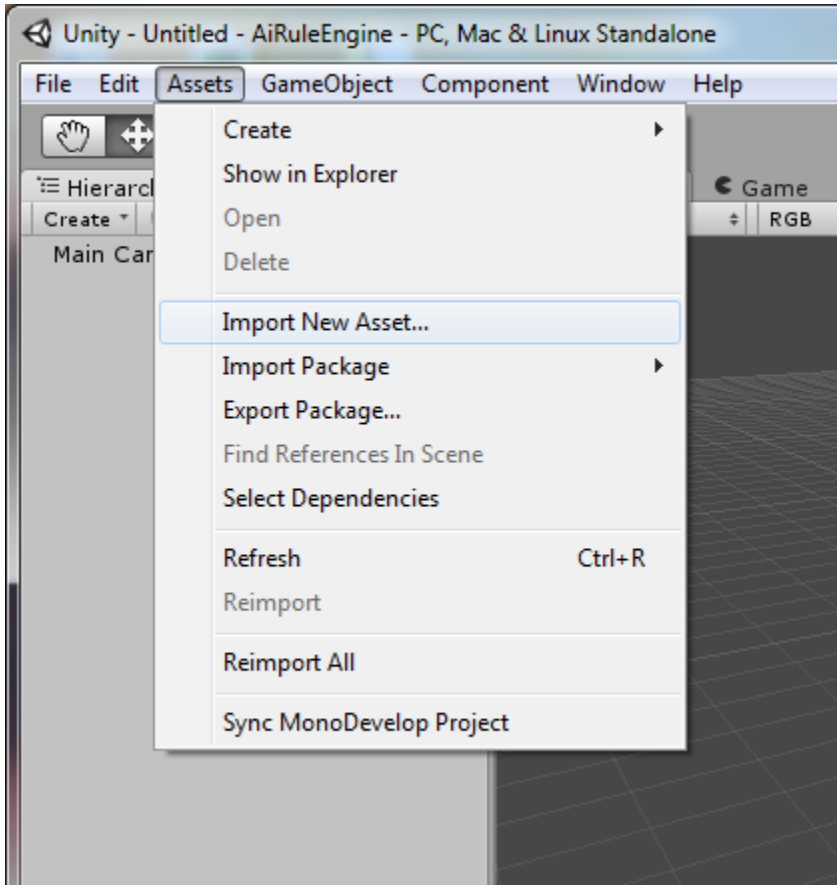
Welcome to the AiRuleEngine Quickstart Guide!

Our goal is to get you familiar with the software so that you can create a basic rule, and have a cube move in the scene based on that rule. Let's get started!

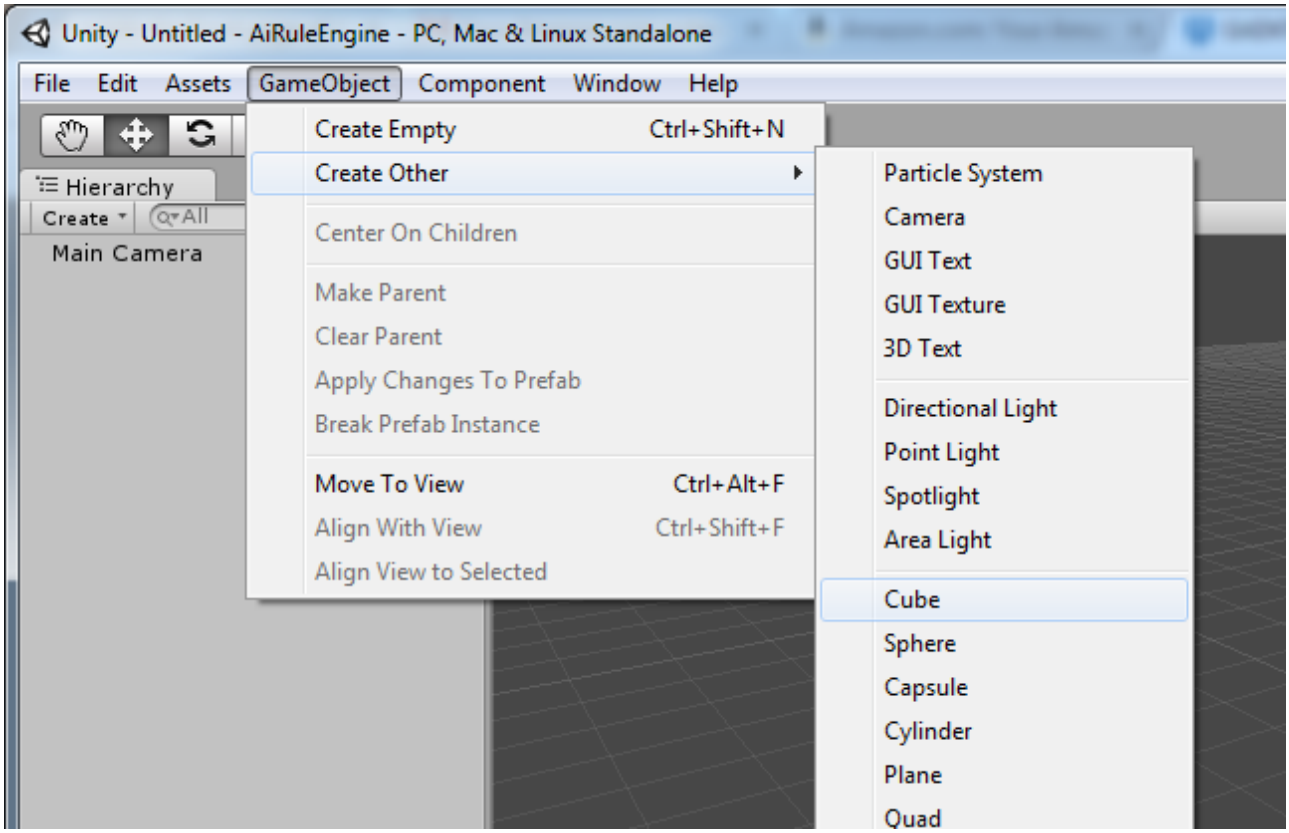
1) First, create a new, empty project in Unity. From the File menu, select New Project…



2) Next, we're going to import the AiRuleEngine. To do this, click on Assets, then Import New Asset and direct it to the AIRuleEngine asset.

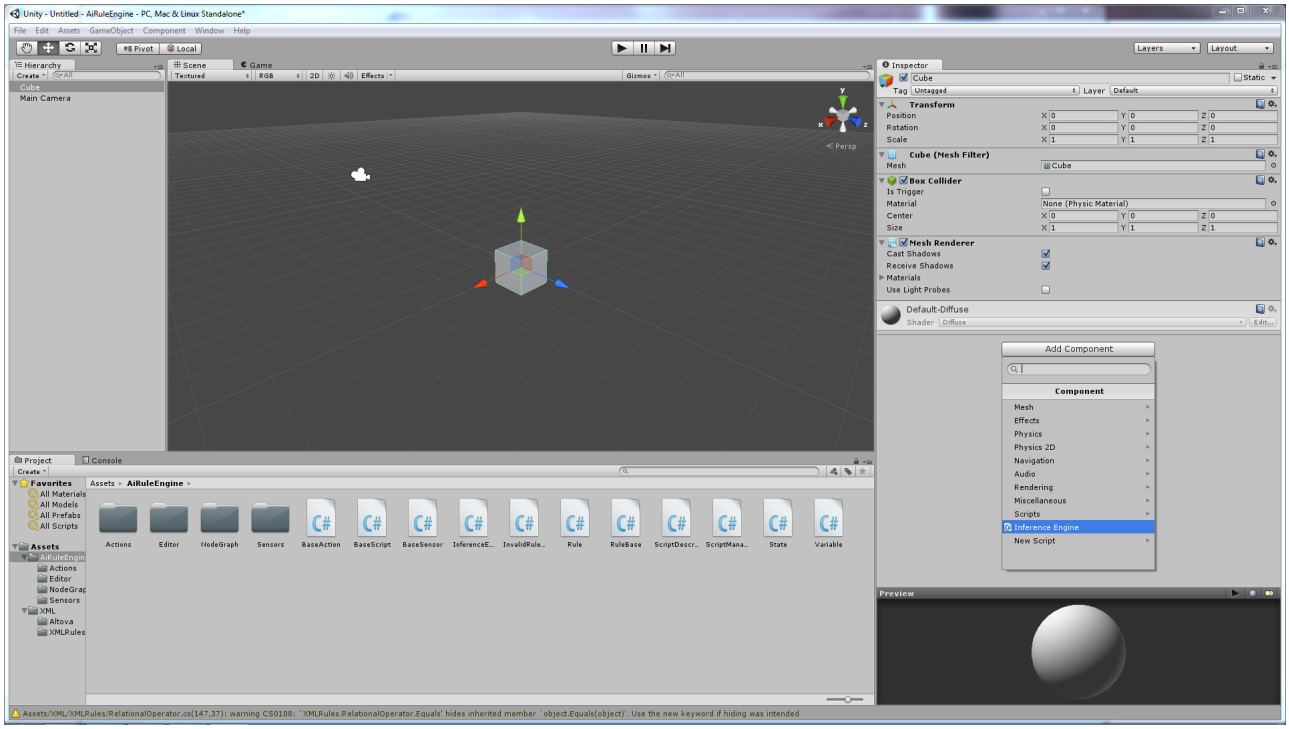


3) Now, we need to create a game object. From the GameObject menu time, select Create Other, then Cube.

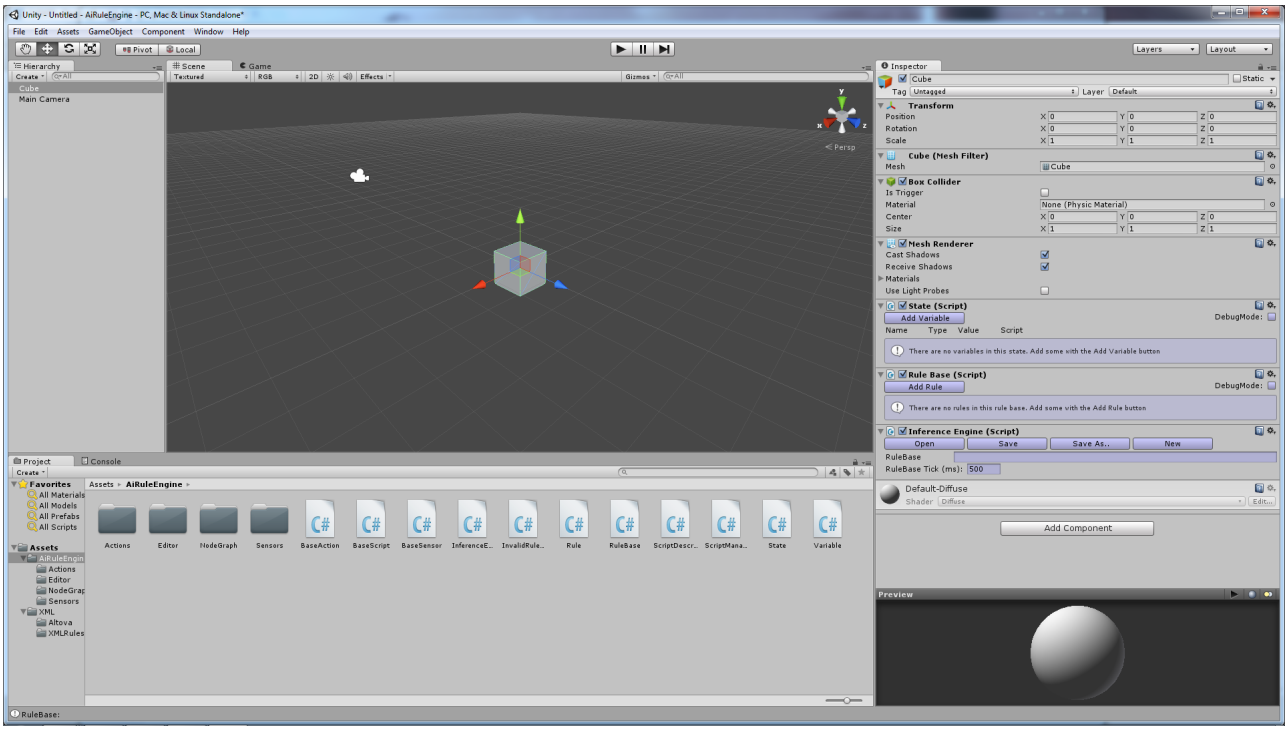


This will create a Cube game object in your scene. You can select any type of game object you want.

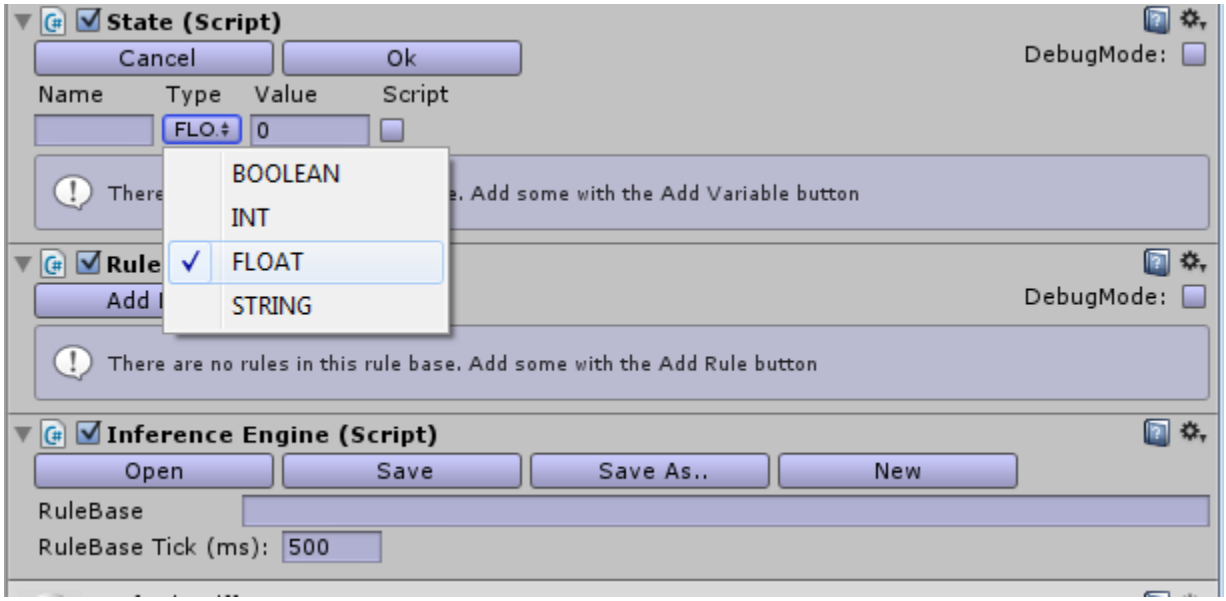
4) Next we're going to add the AiRuleEngine component to the Cube you just created. Go to the Inspector for the Cube object; click Add Component, then select Inference Engine.



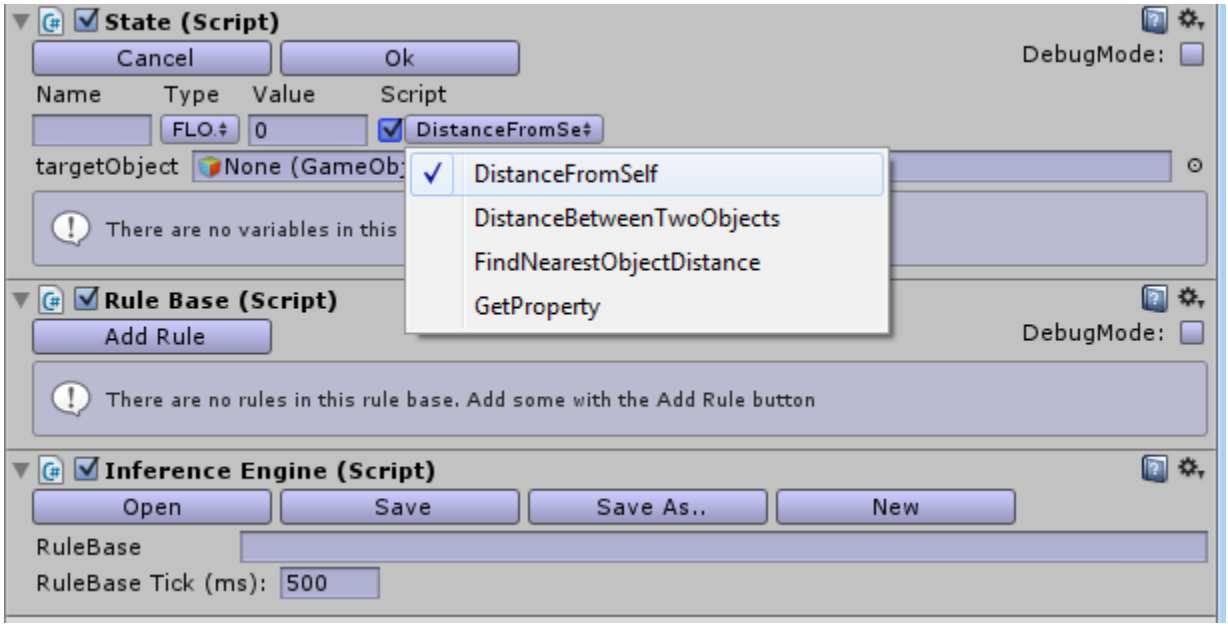
5) This will add three components to your game object. The State component holds a set of variables that represent the state of the world for the Inference Engine. The Rule Base component holds all of the production rules. Finally, the Inference Engine component does the forward chaining inferencing using the production rules and the current state.



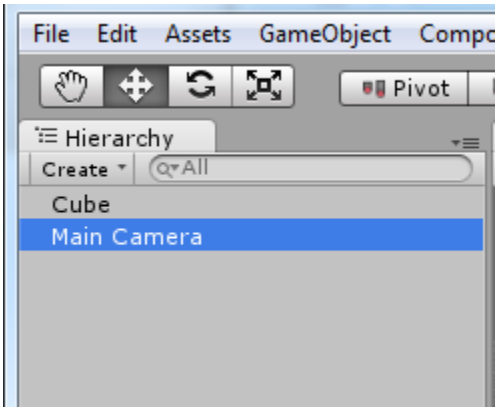
6) We'll go to the State component first, because this is where we will create our variables. To create a variable click on Add Variable button. This will open the variable inspector. Here we'll be able to name the variable, give it a type, and choose whether or not we want to add a sensor script to populate it at runtime, or to just keep it at a constant value. In this example, we're going to give it a sensor script. We do this by clicking on the script checkbox. This will cause a drop down containing all the available sensor scripts to appear. The dropdown will only show sensor scripts that return the same type of value as the variable you are editing. Select the FLOAT type from the Type dropdown.

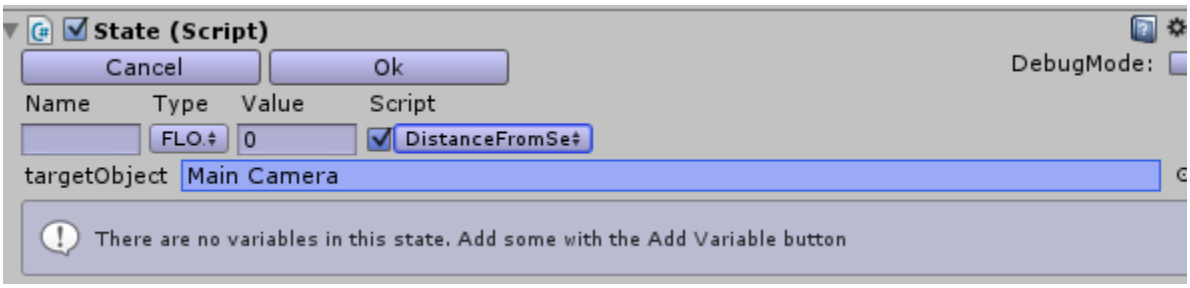


Now choose the DistanceFromSelf sensor script. This will populate our new variable with the distance between our cube and any other scene object we select.

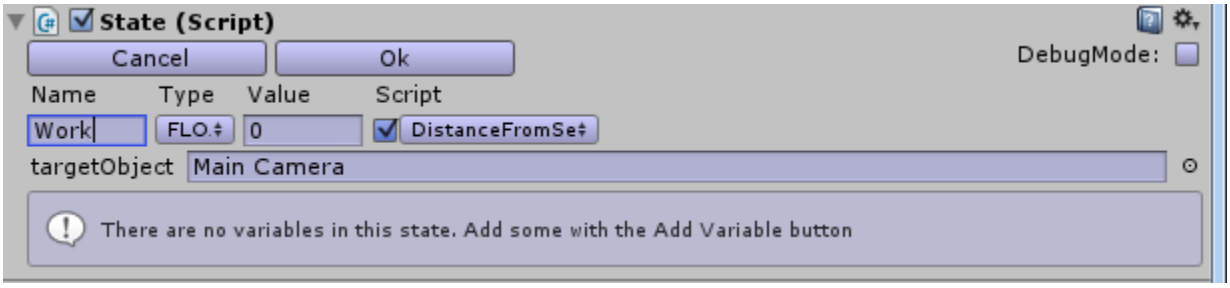


7) Now we need to populate the script's required object. We can use the Main Camera, so that the script will populate our variable with the distance from our cube to the camera. Just drag and drop the Main Camera from the Hierarchy browser to the targetObject field in the variable Inspector.



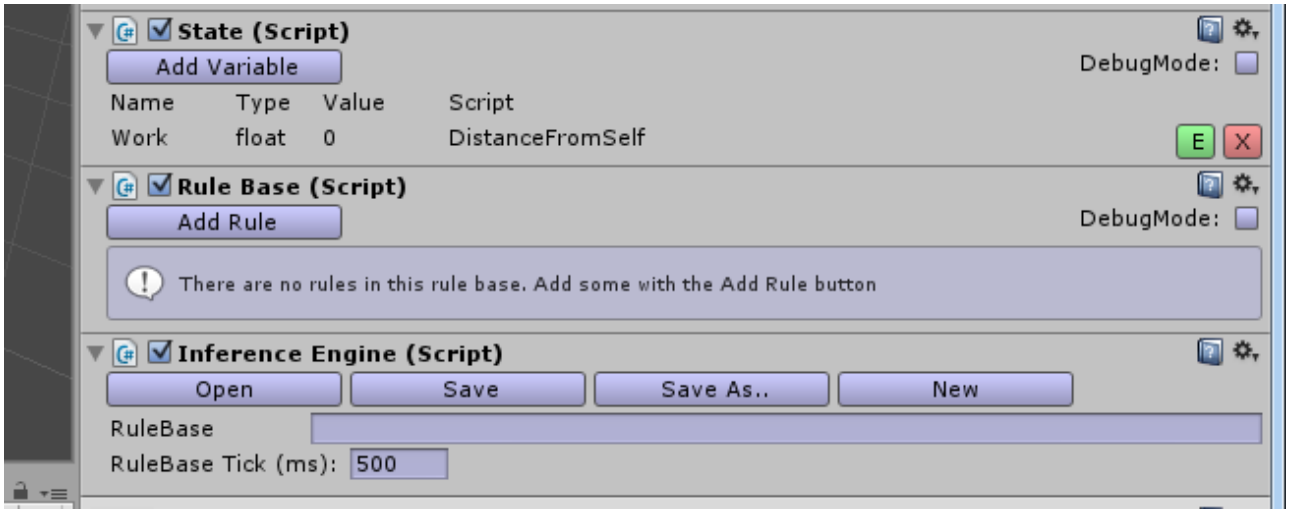


8) The last step is to name the variable, and then press the Ok button.

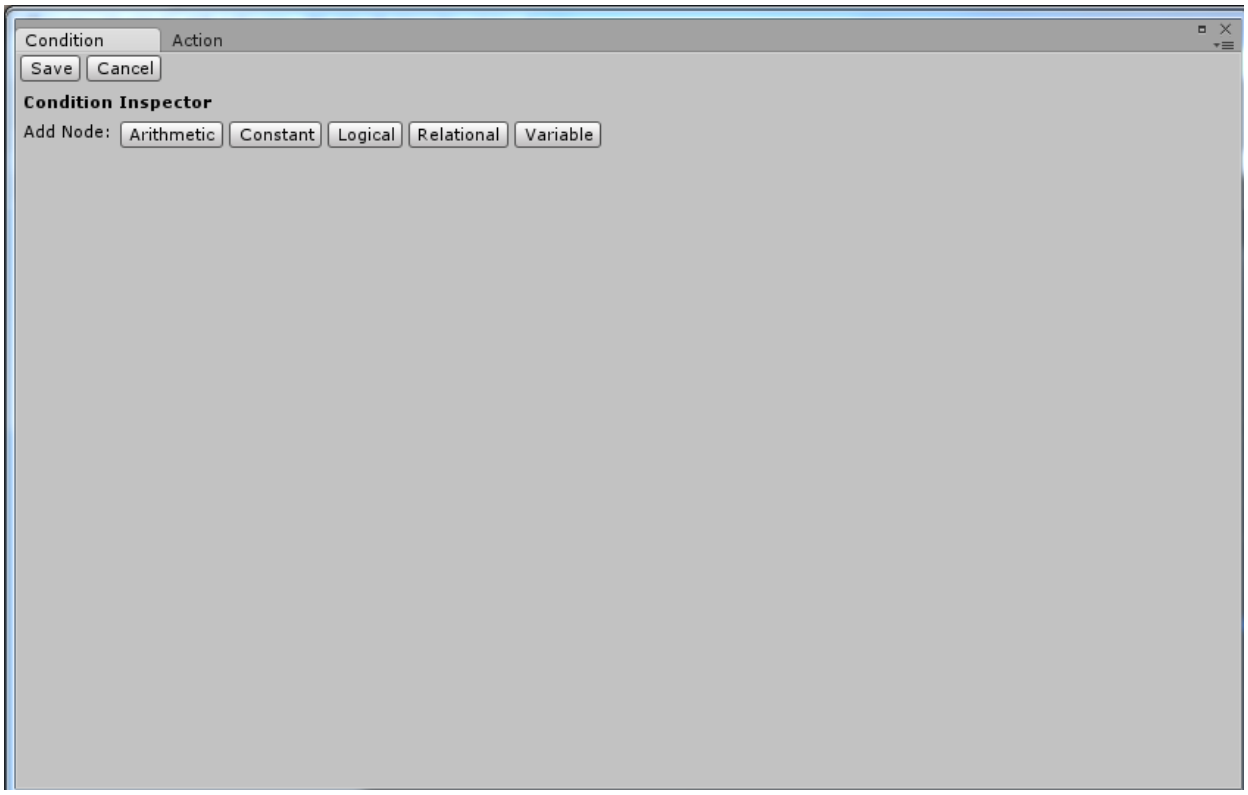


9) You should now see that we have a variable that will have its value populated with a sensor script. You will notice there are two buttons to the right of the variable entry in the state component. The green button labeled E, allows you to Edit the variable by opening the inspector. The red button labeled X deletes the variable.

10) Next, we need to add a production rule that does something interesting. Go to the Rule Base component and click on the Add Rule button. This will open a Rule Editor window.



11) The Rule Editor window has two main tabs: Condition and Action. In the Condition tab, you'll specify the condition for this rule. A rule condition is just a Boolean expression that evaluates to true or false. When the condition evaluates to true, the rule fires by executing the actions that you specify in Action tab.
The Rule Editor lets you create rule conditions visually by adding and connecting nodes to create an expression tree that represents the rule condition.

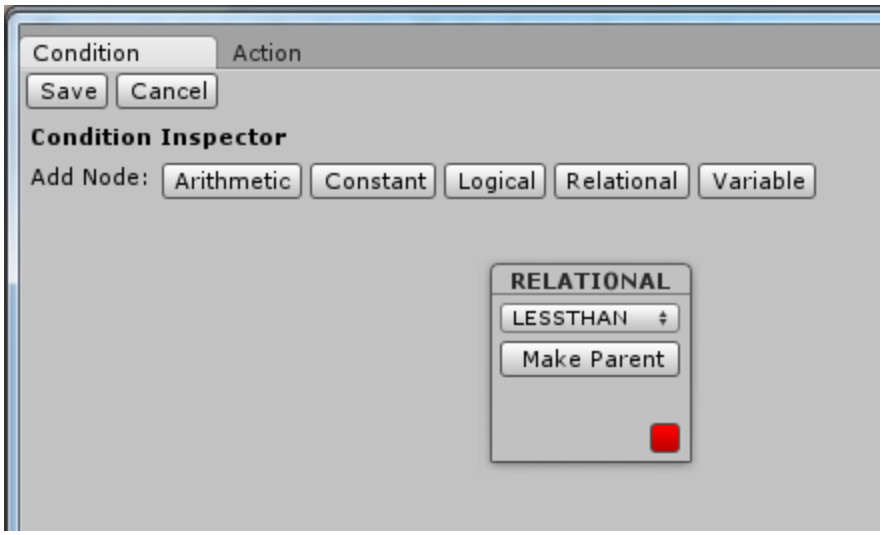


You can use five node types to create rule conditions:

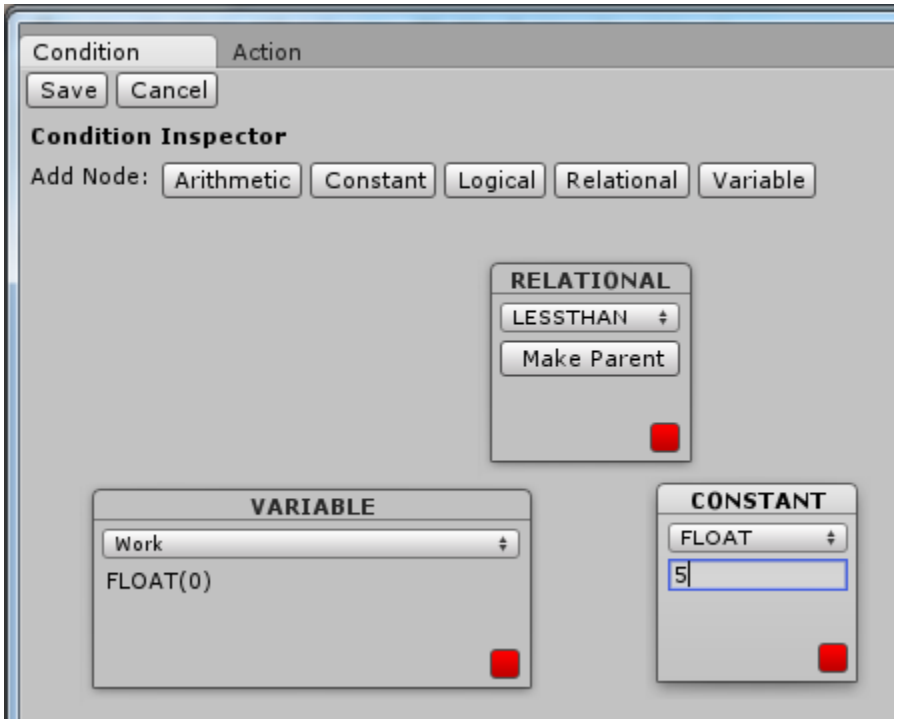
- Arithmetic nodes let you create math expressions using the typical math operators.
- Constant nodes hold a fixed value.
- Logical nodes let you create logical expressions using the And, Or, and Not logical operators.
- Relational nodes let you create relational expressions that compare the values of two sub expressions.
- Variable nodes reference the value of a state variable.

At runtime, the root node of the expression tree recursively evaluates its children and returns a Boolean value of true or false based on the values returned by its children. For rule conditions to work properly, the root node of a condition tree must either be a Logical node or a Relational node as both those types return Boolean values.

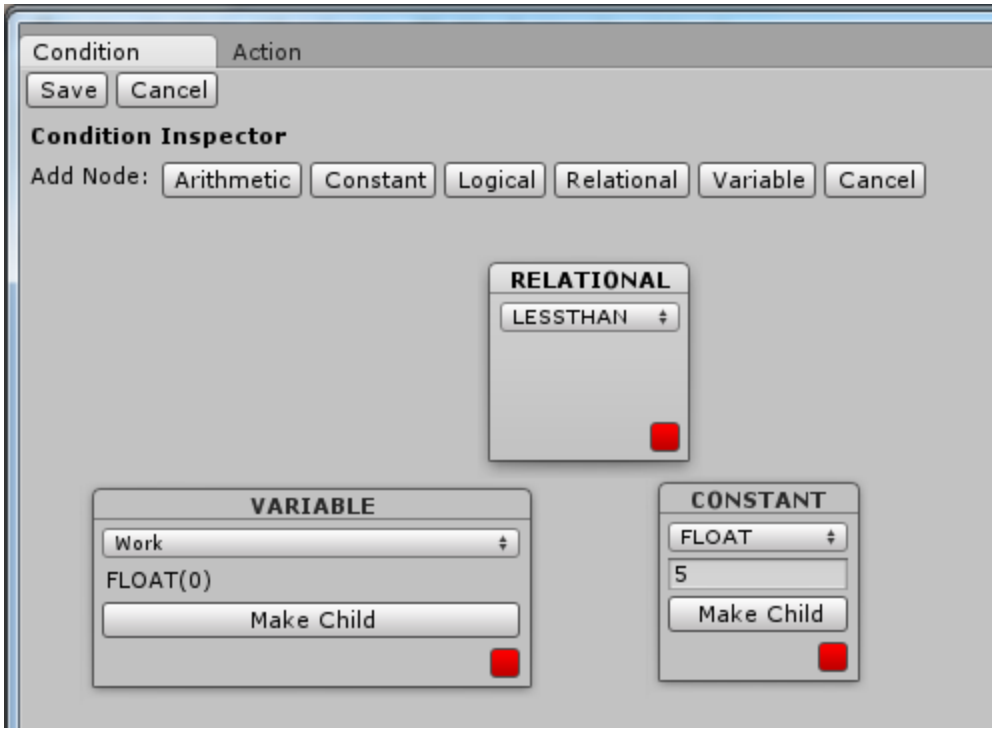
We'll use a Relational node as our root node for this rule. In the Condition tab, press the Relational button to add a new relational node. The relational node lets you select the type of relation using a dropdown menu. Click the drop down in that node and select LESSTHAN. The LESSTHAN relation will return true if the value of its left child is less than the value of its right child; otherwise it returns false.



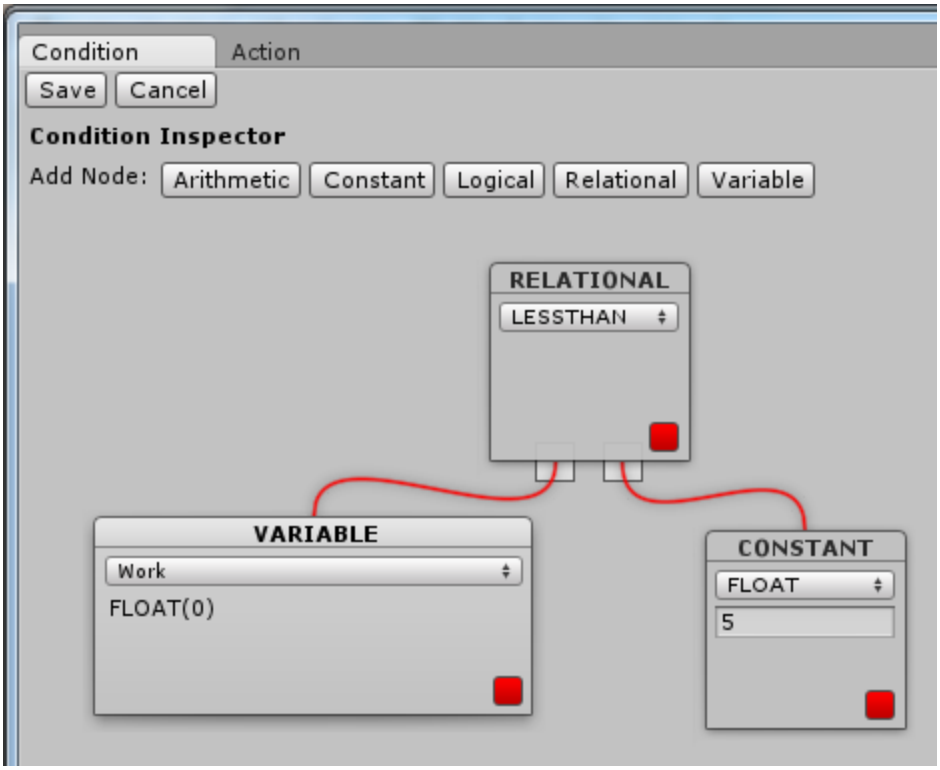
12) Now let's create a Variable node using the variable we created earlier. Click on the Variable button to add a Variable node. Our variable should be automatically selected, if not, select it in the drop down for that node. Also, let's create a Constant node by clicking on the Constant button. Use the drop down for this node to make its type FLOAT, and give it a value of 5.



13) Now we need to connect these nodes to make an expression tree. To do this click on the Make Parent button in the Relational node. This will cause a Make Child button to appear on both the Variable node and the Constant node.

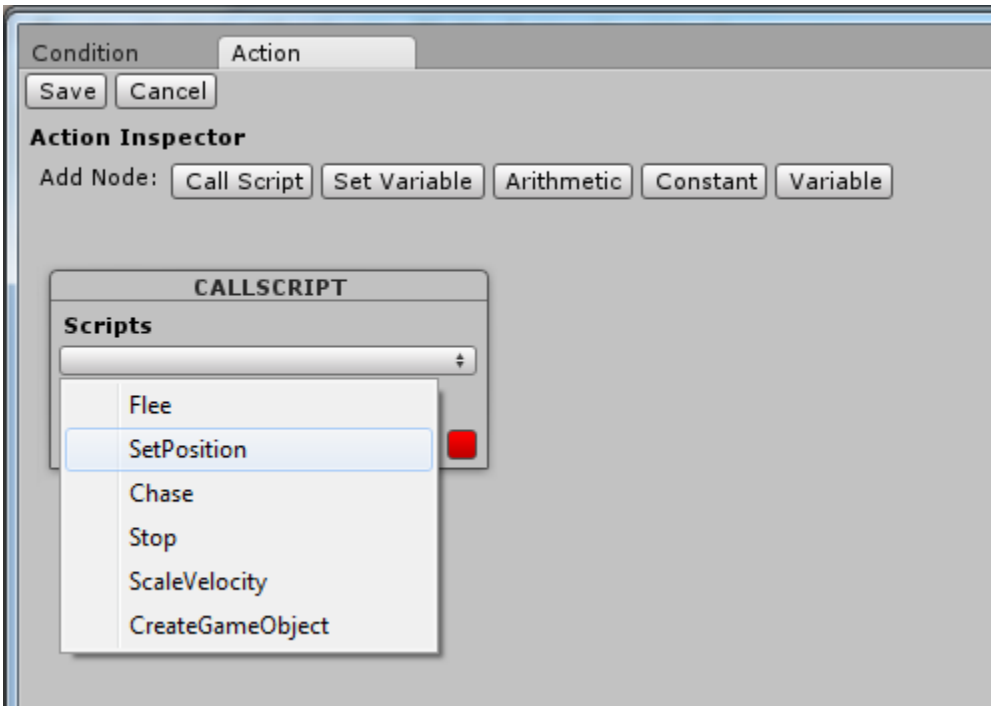


14) First choose the Make Child button on the Variable node, then hit the Make Parent button again on the Relational node, then the Make child button on the Constant node. Our condition should look like this now.

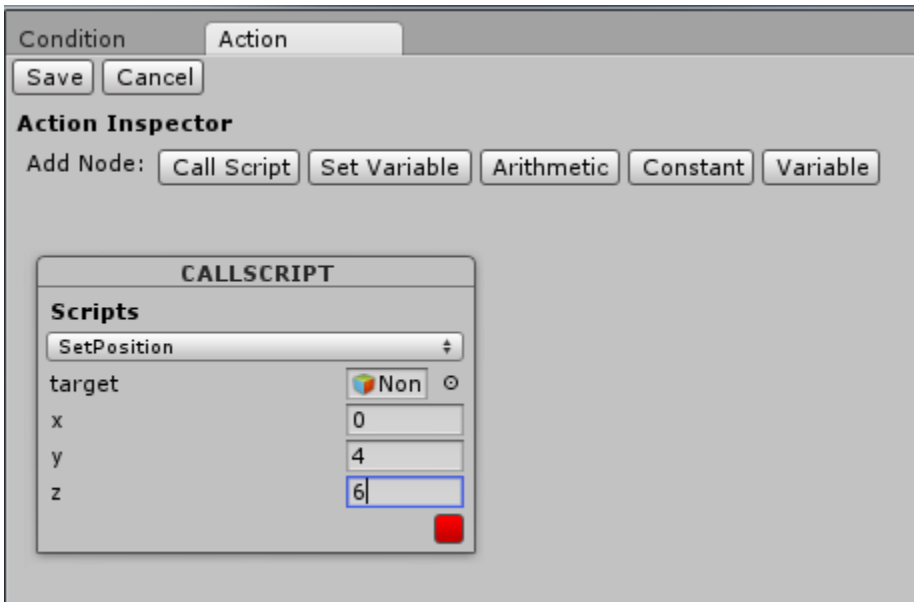


What this means is that our rule will fire if the distance between the Cube object and the Main Camera becomes less than 5.

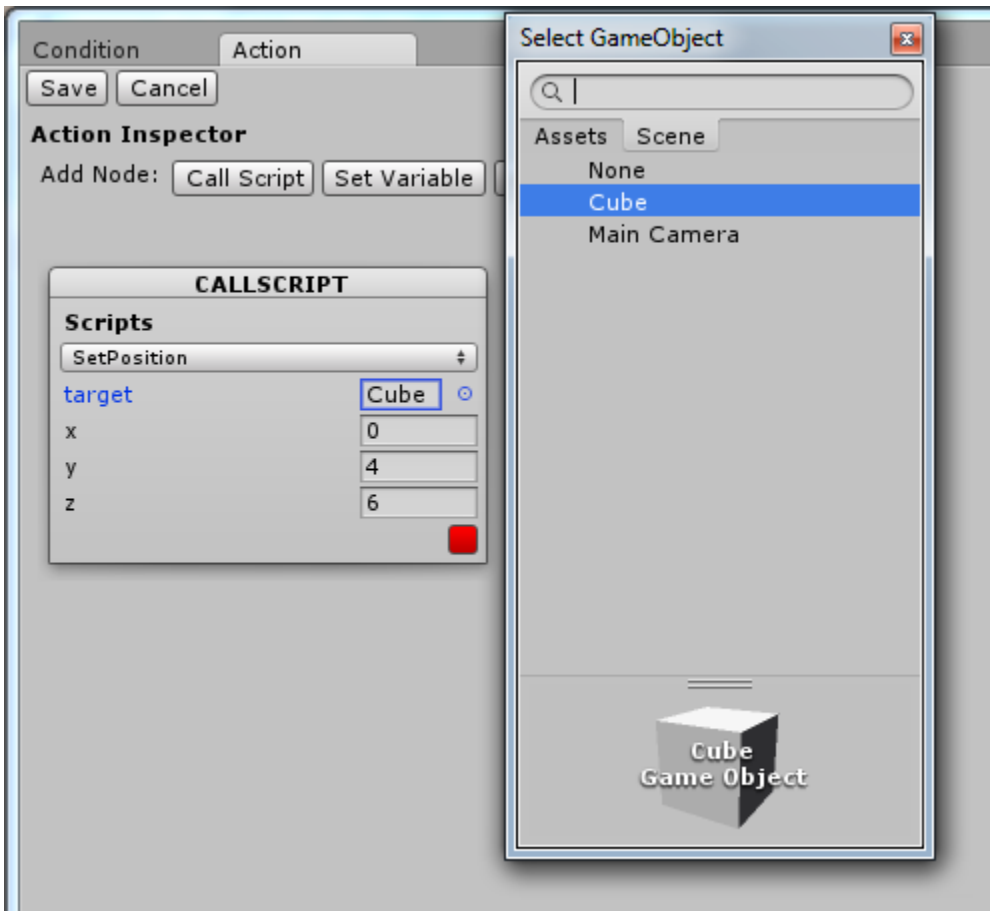
15) Next, we need to add some actions that we want to happen when the rule fires so let's tab over to the Action tab. Click on the Call Script button, then in the drop down of the new node, choose SetPosition. As you'd expect the SetPosition action script sets the target object's position.



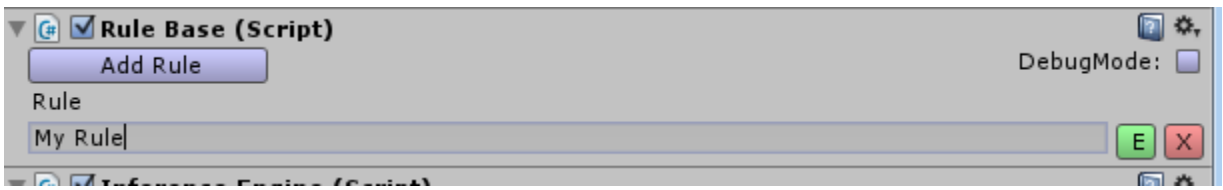
16) Next, let's tell it where to move when the script is called. Enter a value of 4 for Y, and 6 for Z so that we can still see the cube when it moves, and that it's not moving within 5 units of the camera so it's not constantly firing.



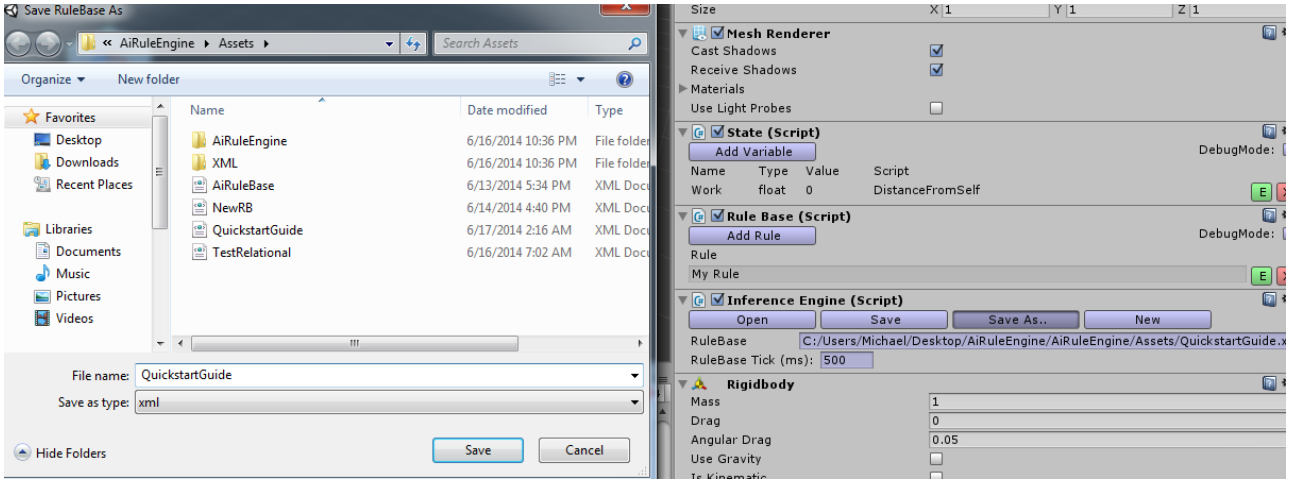
17) Let's not forget to tell the script it will be that cube that is moving. You can drag the Cube into the target field or click on the small circular target button to the right of the field to open an object selection popup.



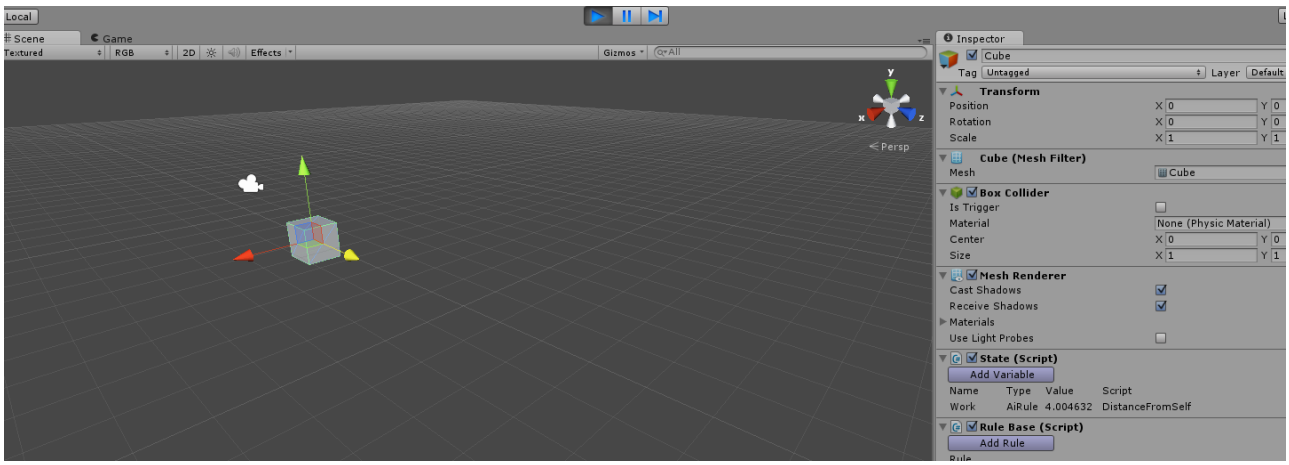
18) We're done editing the rule so click Save, then click the Cancel button in the top left corner of the editor window to close it. Now let's give that rule a name. Just enter a name in the text field to name the rule.



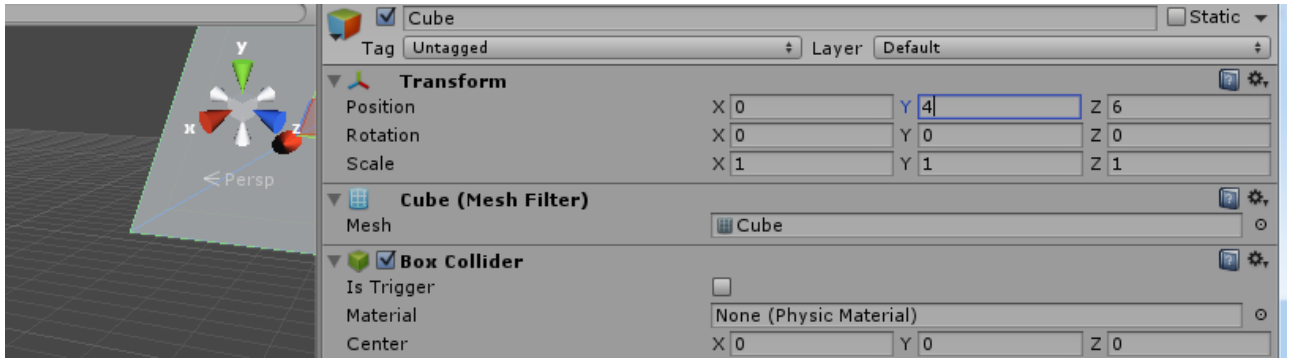
19) Next click on Save or Save As.. in the Inference Engine component. A Window will pop up where you can name your rule base as an xml file. Saving rule bases in a separate file lets you share your rule bases with other people and reuse them across different games.
One caveat though, it's important that you save your rule base each time you make any changes because when you start a scene everything in the editors is erased and the rule base is reloaded from the xml file.



20) Now that everything is set up, let's go ahead and run our Scene. We'll just manually move the cube in the scene while it's running so that it's close to the camera.



21) And BOOM! Our cube has moved where we told it to!



We hope this guide has been useful in getting you started, now you're only limited by your imagination. We have included with this package a few example rule bases that you can experiment with.
If you have any questions, comments, suggestions, bugs, or bug fixes, please feel free to contact us via email or from the contact page on our site. Also please share with us the awesome games that you make using our engine!