Visionaire Basics for beginners

Video 7.1 – Conditions #1

Introduction

In this section we will start with a discussion on what conditions are and how conditions work.

What are conditions?

Conditions can be seen as a sort of switch that we use in the game world to signify a state or a condition. Let's look at a practical example.

Let's say we have a key in our scene that our player needs to pickup and add to our inventory. Visionaire needs to have a way to know whether the key has been picked up or not picked up. We can keep track of this via what we call a condition.

So we have our key in our scene.

We will pick up our key and add it to our inventory. When we do this we need to have a condition to keep track. I will create a condition called **'Key_picked_up'** which we initially set to FALSE.

When we pick up the key our condition '**Key_picked_up'** needs to be set to TRUE which allows Visionaire to remove it from the scene.

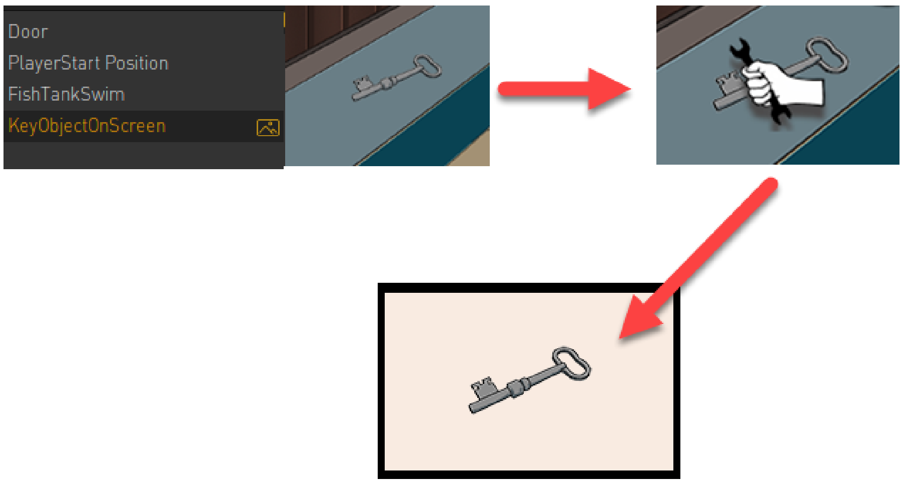
Important things to know

- All conditions are global which means it doesn't matter where your condition was created as you can access them from anywhere
- There are several places in Visionaire where you can create conditions, but this simply defines how items are grouped when displayed. All conditions are accessible from anywhere but if I create my condition in the SCENE area I need to navigate to that section to access it.
- Take time to decide where best to create your conditions as it does matter later in a project's lifetime for organizational purposes. In our example with the key it would make sense to create the condition on the object itself.

Create our first condition

In this section we will use the above example to create our condition.

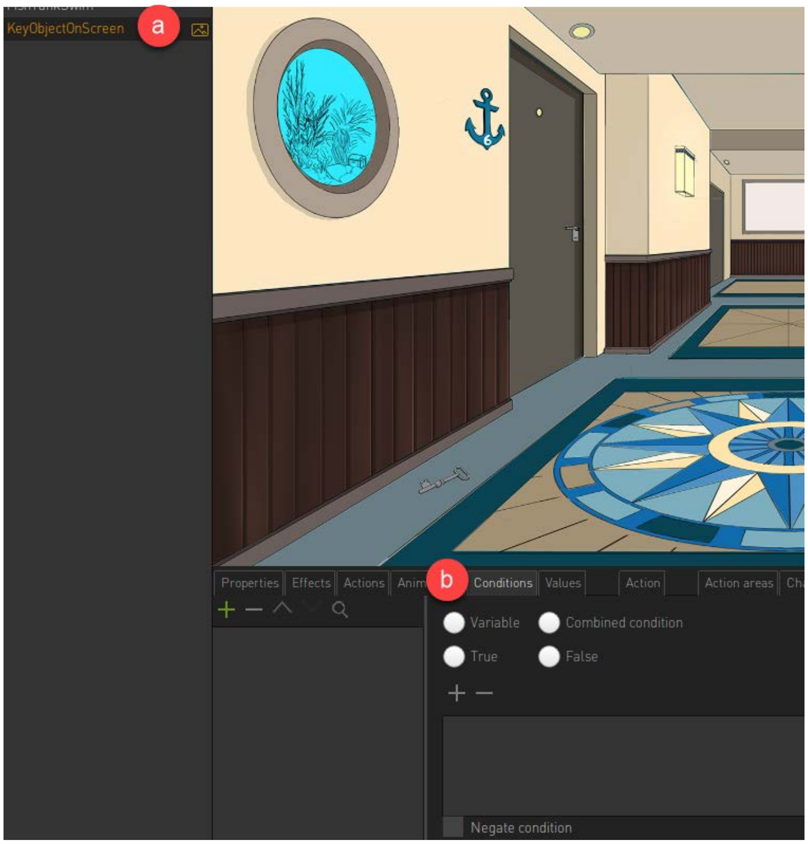
1. In our scene we have a **KeyObjectOnScreen** object. We have already setup interaction in that if the player 'USES' this object it will be added to your inventory. The problem is that the key remains on screen and this is what we will fix.



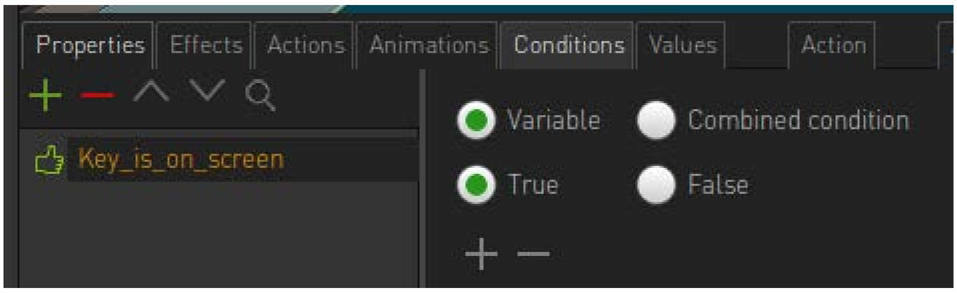
2. First is to access the Condition area on the object itself.

   a. Ensure your object is selected.

   b. Click on the Conditions tab

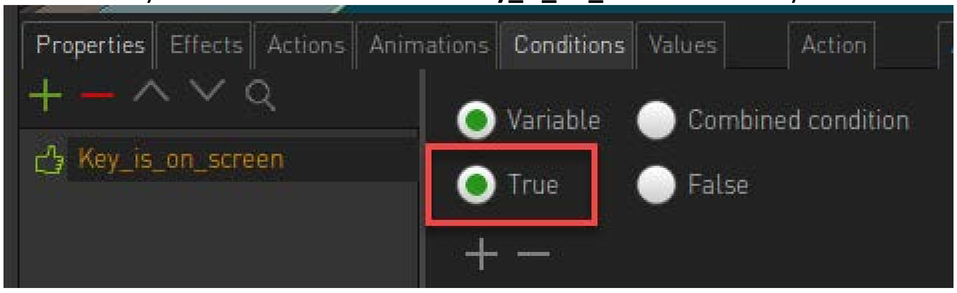


3. We will now create our condition. To do this click on the green plus icon and name it something appropriate.



NOTE!! Remember that the name of the condition matters. If we call it **Key_is_on_screen** then the default entry should be TRUE but if we call it **Key_picked_up** then the initial entry is FALSE.

4. Set the entry to TRUE as the condition **Key_is_on_screen** is initially TRUE.

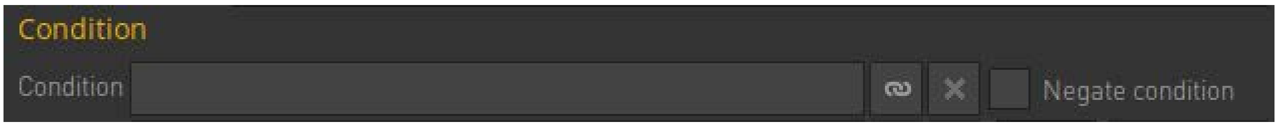


5. Now that we have our condition we need to use it.

Using our created condition

Now that we have created the condition we will now use it to evaluate whether the key is picked up. If you think about it we need to state whether the key is 'active' in the scene (does it appear onscreen). We do this in a different area under the Properties page.

Here we have a field called Condition and this is where we add our condition to evaluate whether the object is active or not.

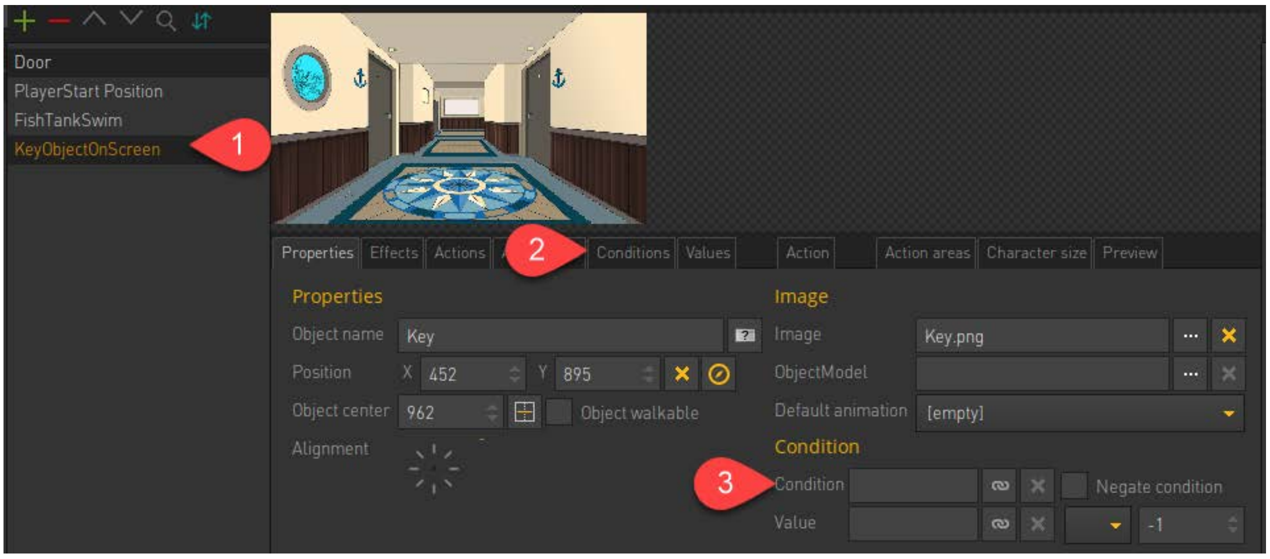


To set this do the following;

1. Ensure that your **KeyObjectOnScreen** is selected.

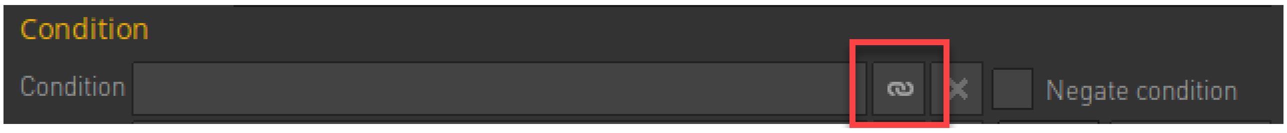
2. Remember that we created a condition under the Conditions page. Keep this in mind as we continue.

3. On the Properties page focus on the CONDITION field area

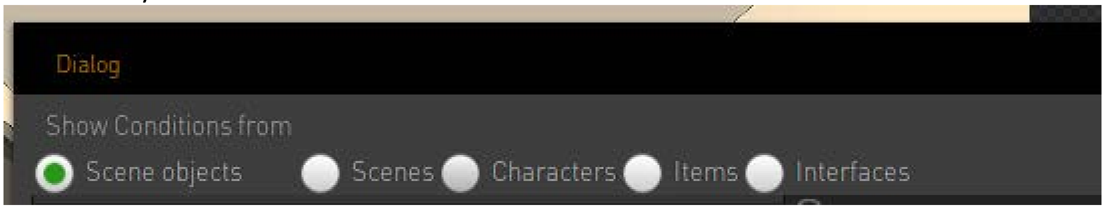


4. The CONDITION field evaluates whether an object is active or not. In our case it will define whether the key will be displayed on screen or not. So let's add our condition.

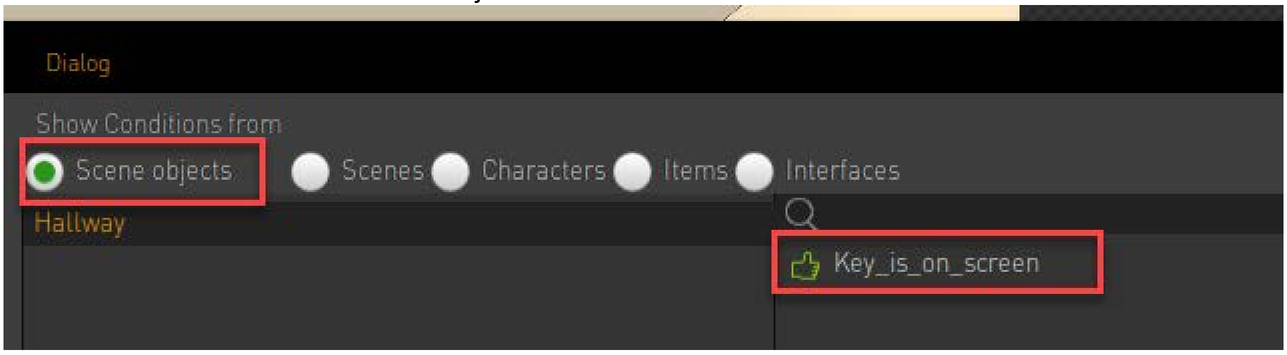
5. Click on the small link chain icon shown below to choose and set our condition



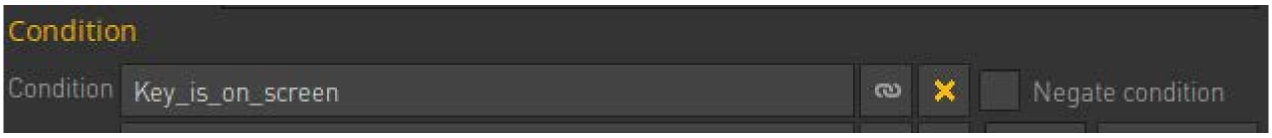
6. We now have a list of conditions that we can choose from. We have several areas we can access and the conditions we see here are dependant on where they were created. For more information see the introductory section.



7. We created our condition under our Object so that's where we find it. Select our condition and click OK.



8. Our condition is then added.

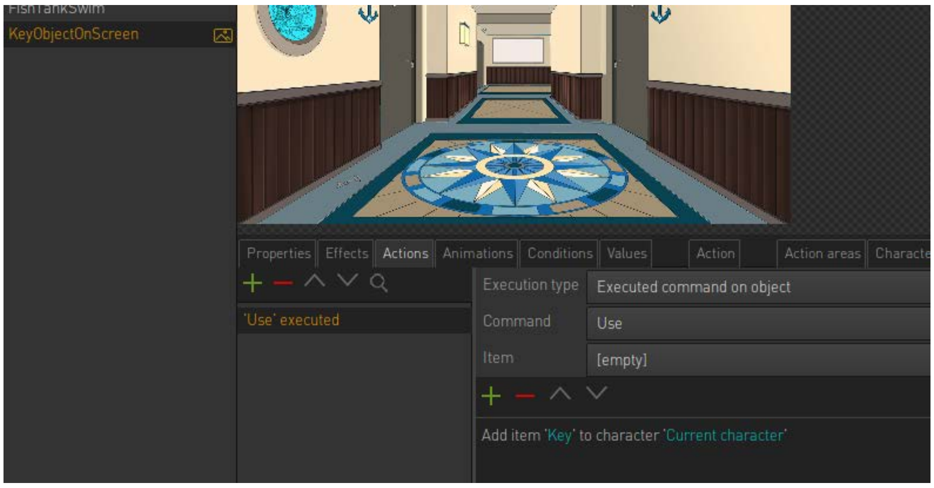


9. Now play the scene and pickup the Key Object. Notice that it did not disappear from the game world! Why? Well, the answer is simple. We aren't changing the condition to false when picking it up. It remains TRUE (so key is active an onscreen) throughout our play session.

   Let's fix this.

Changing a condition

So we need to set the condition **KeyObjectOnScreen** to FALSE when the player picks up the key object. First where is this done? We do this on the Actions area of the object and we already have some action parts placed here. We have a USE command that adds the key object to our inventory.



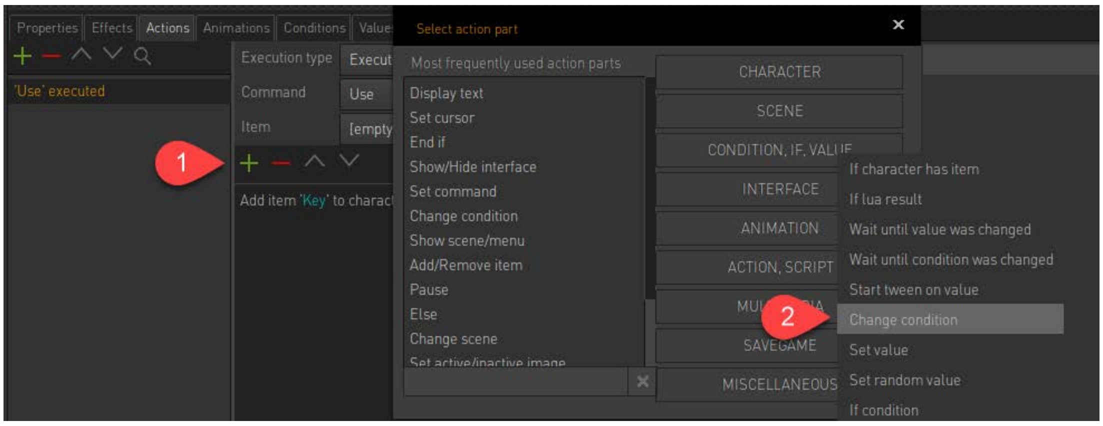
'Use' executed

Execution type: Executed command on object
Command: Use
Item: [empty]

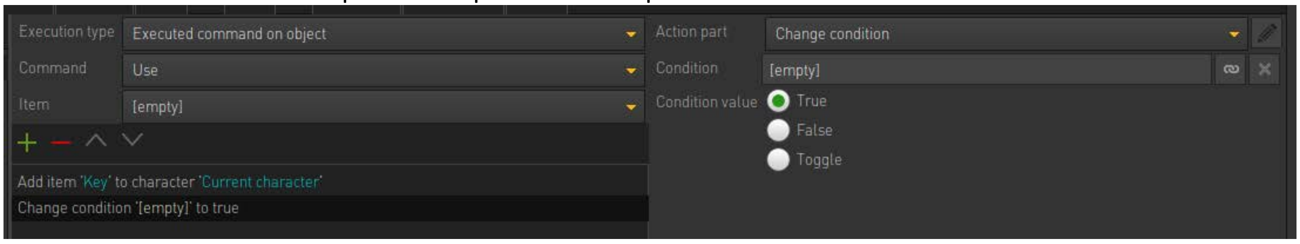
Add item 'Key' to character 'Current character'

So the logical thing to do would be to change our condition at this point. To do this do the following;

1. Add a new Action part by using the small green plus icon.
2. Add action part CHANGE CONDITION found under CONDITION, IF, VALUE menu



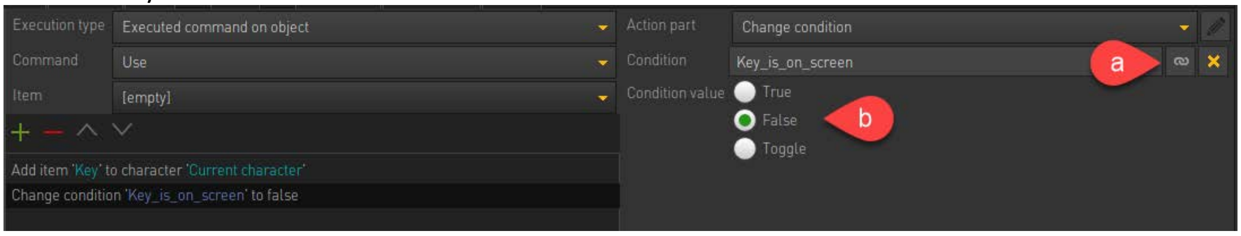
3. This will add the new action part with options we can provide.


Execution type: Executed command on object
Command: Use
Item: [empty]

Add item 'Key' to character 'Current character'
Change condition '[empty]' to true

Action part: Change condition
Condition: [empty]
Condition value: [✓ True] False Toggle

4. We need to choose our condition under the CONDITION field and then choose what to change it to.
   a. Choose your created condition
   b. Set the entry to FALSE



Execution type: Executed command on object
Command: Use
Item: [empty]

Add item 'Key' to character 'Current character'
Change condition 'Key_is_on_screen' to false

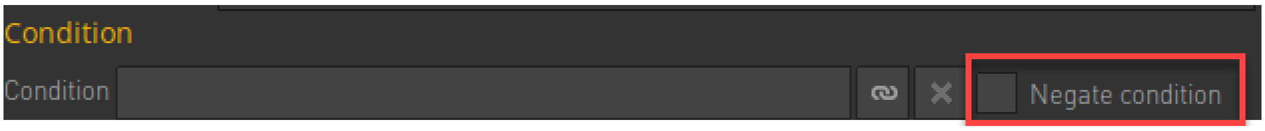
Action part: Change condition
Condition: Key_is_on_screen
Condition value: True [✓ False] Toggle

So our condition **Key_is_on_screen** will be set to false as soon as the player picks up the object. This is the entry we use to define whether our **KeyObjectOnScreen** is active (check the Properties -> Condition field). If we change the condition to false the KeyObjectOnScreen will not be active anymore.

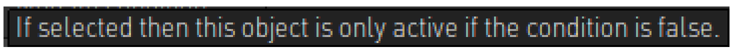
Now play your scene and note if the key disappears or not.

Negate Condition

You'll note that you have a small tick box next to your condition field called NEGATE CONDITION.

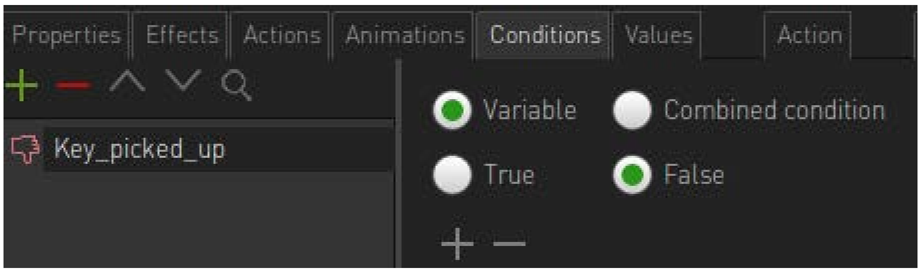


What does this do? Let's hover our mouse of the icon and read the tooltip


If selected then this object is only active if the condition is false.

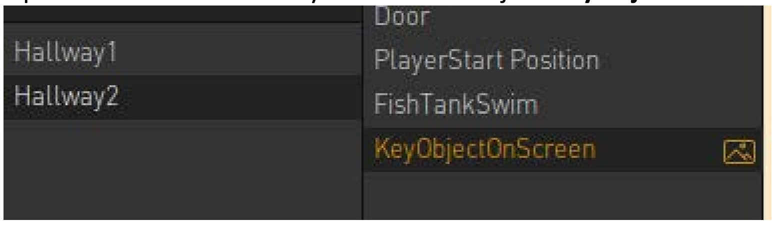
So what does this mean? This has to do with what we choose to call our conditions and how we evaluate those conditions to determine if an object is active or not.

With the key example used above we will use a different condition name. Initially we chose to call it Key_is_on_screen which obviously defaults to TRUE but what if we instead called it **Key_picked_up**? The entry would then need to be set to FALSE.

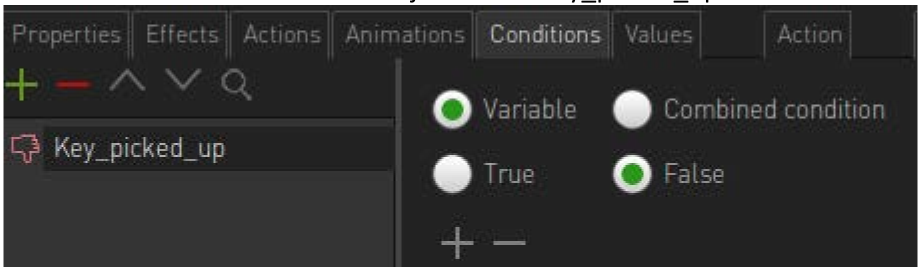


So let's do this;

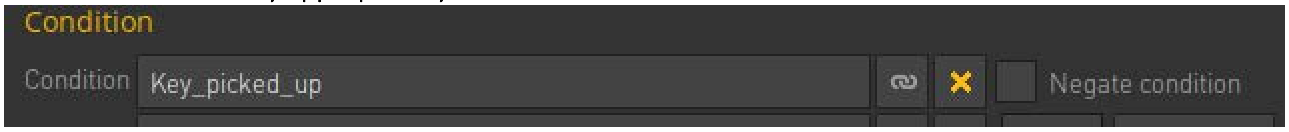
1. Open scene called Hallway2 and select object **KeyObjectOnScreen**



2. Create a new condition on that object called Key_picked_up and set this to FALSE



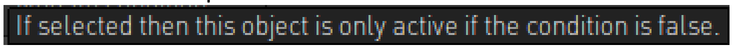
3. Set the Condition entry appropriately



4. Play your scene and note our problem! Our key does not appear! Why?

5. It's because we marked our Condition to FALSE due to how we named it. This in turn means when evaluated it sees it is false and thus inactive. It is here where the NEGATE CONDITION comes in handy.

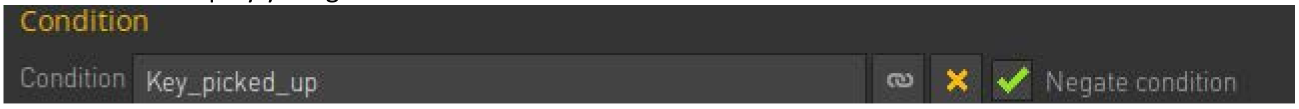
   Reread the tooltip



If selected then this object is only active if the condition is false.

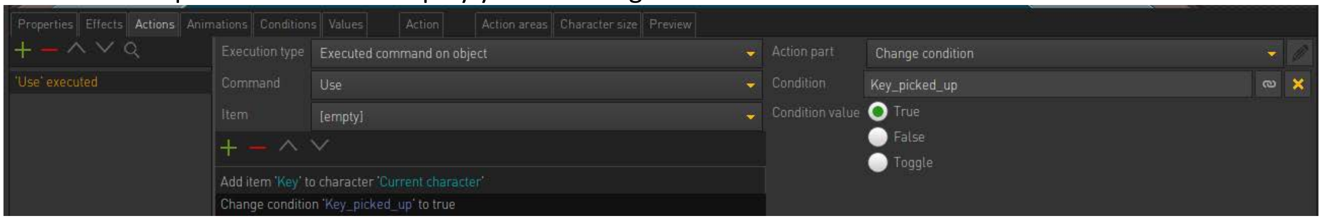
6. So in effect if we tick this then the object will be seen as active / displayed if the entry is false. We are in effect NEGATING the condition.

   Tick this box and play your game



7. The object now appears again.

8. Set the action parts as needed and play your scene again.



'Use' executed

Execution type: Executed command on object
Command: Use
Item: [empty]

Add item 'Key' to character 'Current character'
Change condition 'Key_picked_up' to true

Action part: Change condition
Condition: Key_picked_up
Condition value: [✓ True] False Toggle

9. All behaves as expected.