

Making Your World with Triggers – triggers obviously

The goal of this tutorial is to build on the last tutorial (Making Your World Leverly) by teaching you how to use two things, conversation filters and triggers. This tutorial assumes that you've finished the previous four Make Your World tutorials.

Getting Started:

- 1) Open The Aurora Toolset.
- 2) Open LeverMagic4 and save it as "LeverMagic5".

PART 1 Creating a conversation filter:

A conversation filter determines whether a conversation choice should be displayed or not. For a player response, all available choices will be displayed on screen, and the player can choose one by clicking it. For an NPC response, though, there can only be one choice displayed on screen (an NPC cannot say two things at the same time). So, if there are multiple choices for the NPC, usually the first choice in the conversation file is chosen automatically.

Let's look at the following example.

- 3) Double-click the guardianwelcome Conversation, and add the following conversations under Root.

"You're back, <FirstName>. Why are you back?"

 "I just wanted to thank you for your help."

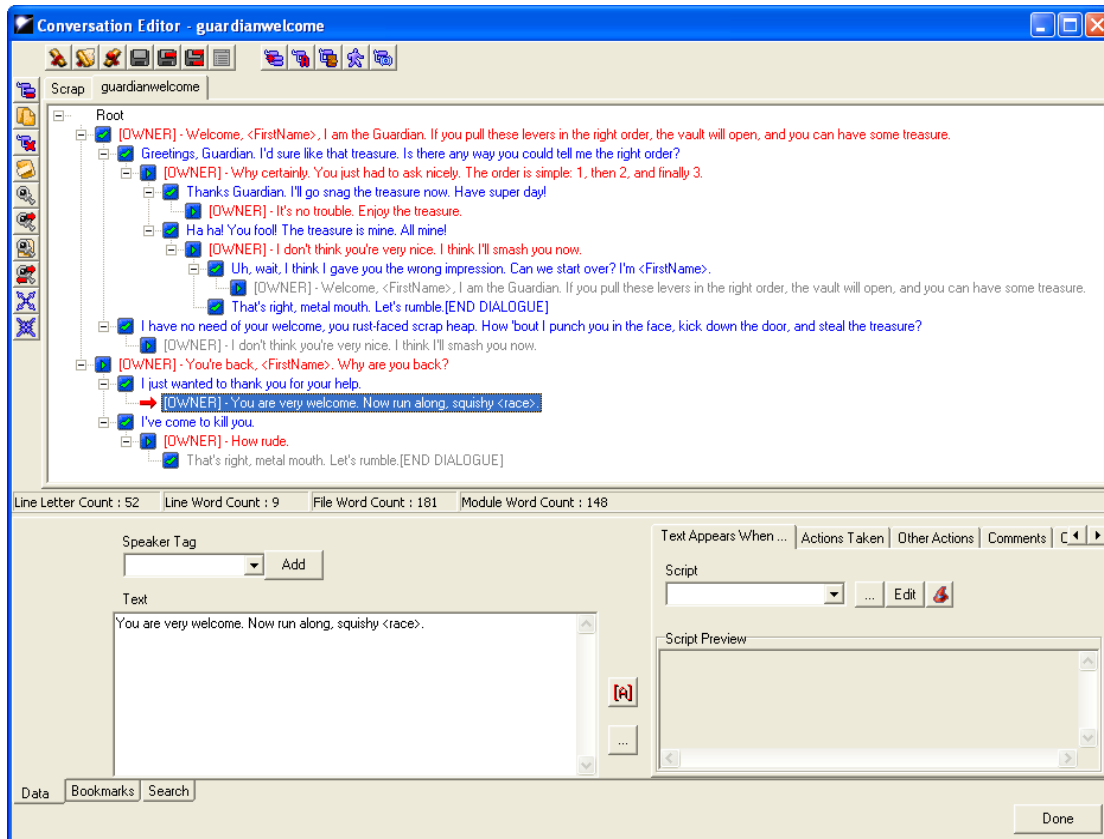
 "You are very welcome. Now run along, squishy <race>."

 "I've come to kill you."

 "How rude."

Create a link to "That's right, metal mouth. Let's rumble."

Check the following screenshot to make sure you have done everything right.



Now, from the Root, you can see two NPC responses: “Welcome, <FirstName>...” and “You’re back, <FirstName>...” Without a filter on the first response, the NPC will always give you the first response.

What we really want is this: the NPC will give you the first response when you talk to him for the first time. Afterwards, if you talk to him again, he should give you the second response. This requires a **conversation filter** on the first response.

- 4) Open the module in ScriptEase. Create a new Plot Token. Call it metGuardian.
- 5) Right-click the Guardian Conversation folder, New Specific Encounter → Encounters → Base CodePak → Conversations → Display **Conversation Node** only when all conditions are satisfied
- 6) In the left pane, select guardianwelcome.dlg. In the centre pane, select the line “[OWNER] - Welcome, <FirstName>, ...” Click OK.
- 7) Right-click the encounter, Add a Definition → Binary → Plot → Define **Owms** as whether **Owner** owns plot token **The Token**.
- 8) In the definition you just created, in the “Owner” tab, dropdown menu → **PC Speaker**.

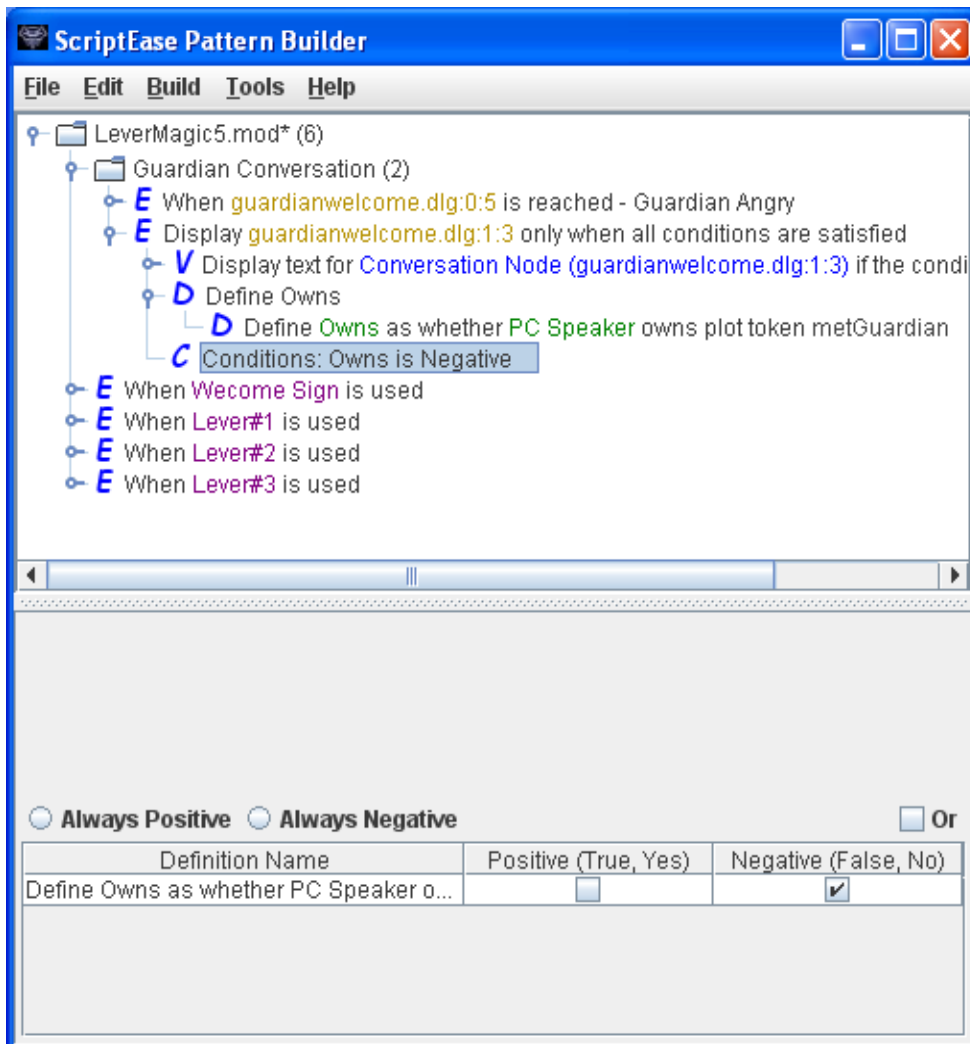
In the “The Token” tab, select Constant → metGuardian.

9) Select the MultiCondition (the line starting with a blue C). On the line “Define Owns as whether ...”, check the checkbox under “Negative (False, No)”.

A *MultiCondition* is just a list of conditions (binary definitions with their respective values). Recall that a binary definition can only have two values, yes or no.

Check the screenshot below. Now, intuitively, what you have here is a conversation filter that states:

If *Owns* is **negative** (which means the PC does **not** own the plot token “metGuardian”, which in turn means you have **not** met the Guardian), then the dialogue line “Welcome, <FirstName>, ...” will be displayed. Otherwise, the second choice, “You’re back, <FirstName>...” will be displayed automatically.



10) Lastly, we need to assign the plot token “metGuardian” to the PC after we met the guardian, so the above conversation filter actually works.

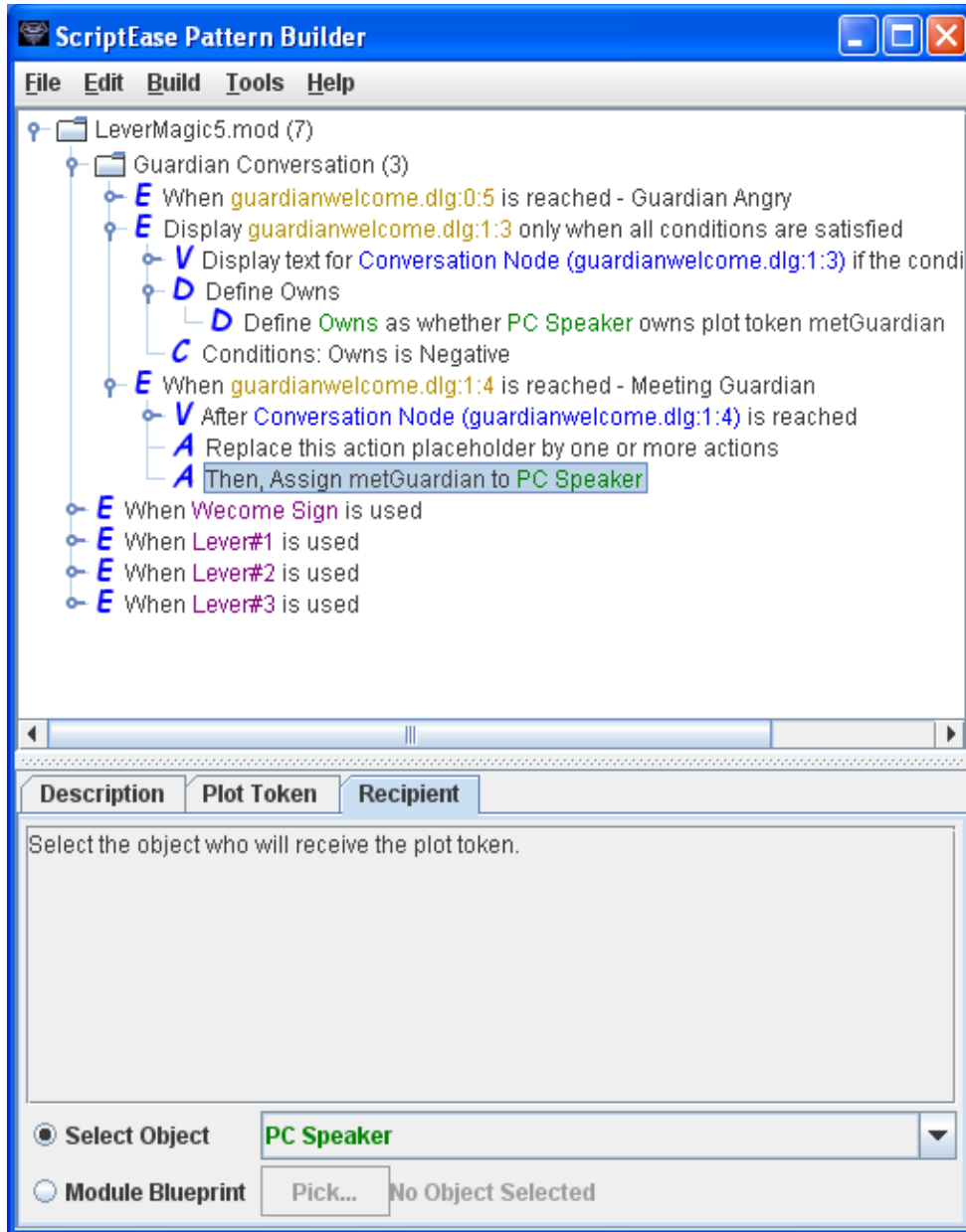
Right-click the Guardian Conversation folder, New Specific Encounter → Encounters → Base CodePak → Conversations → When [Conversation Node](#) is reached

And choose the line “[OWNER] - Why certainly. You just had to ask nicely...”

11) Select the new encounter, change the Encounter Name under the “Description” tab to “When <p1> is reached – Meeting Guardian”

12) Right-click the encounter, Add an Action → Action Atom → Plot → Assign [Plot Token](#) to [Recipient](#)

13) In the Action you've just created, under the “Plot Token” tab → Constant → metGuardian, and under the “Recipient” tab → dropdown menu → [PC Speaker](#). Your module should now look like the screenshot below. You can try your module now. When you talk to the Guardian a second time, the Guardian should give you a different response.

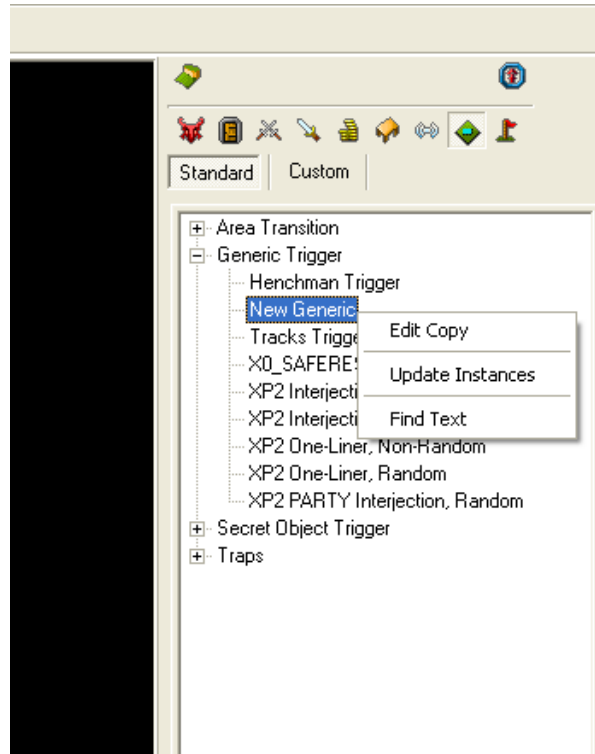


PART 2 Creating a trigger:

14) Open LeverMagic5 in Aurora if it is not already opened.

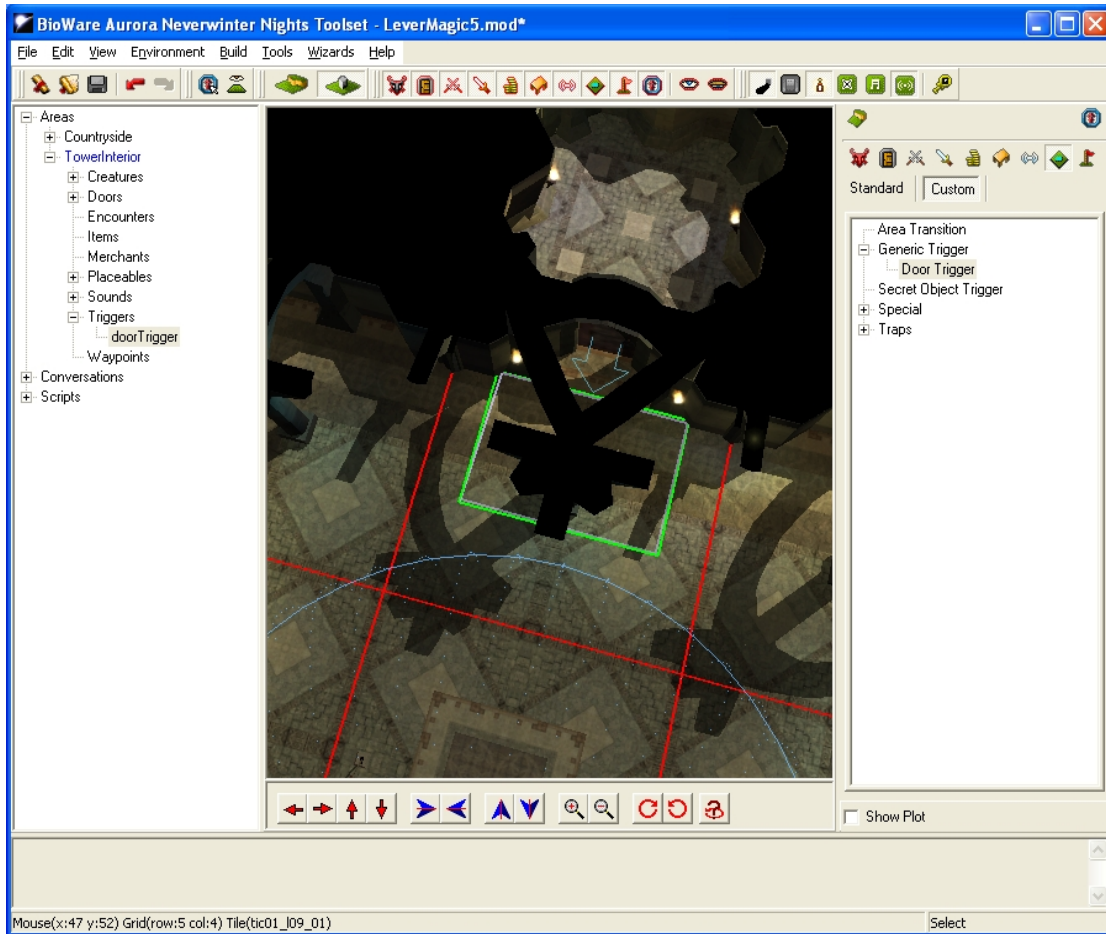
15) Double-click and open the TowerInterior area.

16) Under the Paint Triggers Tool (see below), right-click New Generic and Edit Copy. Change the name to Door Trigger and the tag to doorTrigger. Click OK.



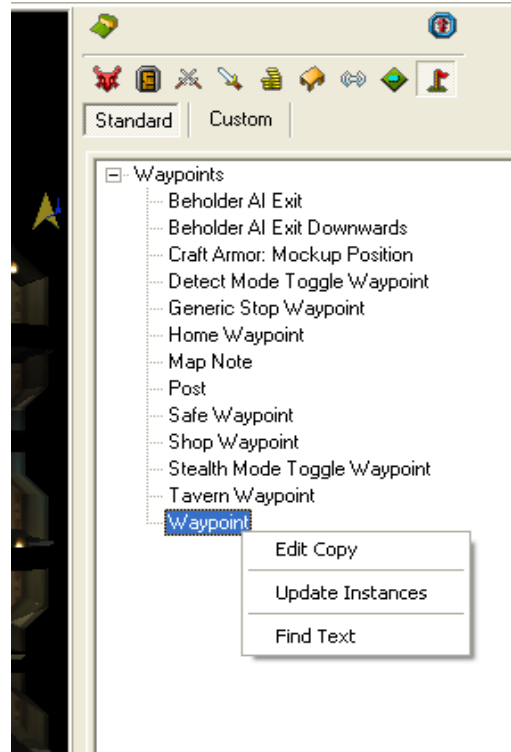
17) Now click Door Trigger to make sure it is selected.

18) Draw a rectangular trigger around the Vault Door. All you need to do is click on the place where you want the first corner, then click on the place where you want the second corner, and the third, etc. When you finish, double click the starting corner to complete the trigger.



19) All right, now we have a trigger. It doesn't do anything until we put some scripts on it. Let's do that. Before we close the module, we need to add one more thing, a waypoint.

20) Under the Paint Waypoints Tool, right-click Waypoint and Edit Copy. Change the name to Trigger Centre and the tag to triggerCentre. Click OK.



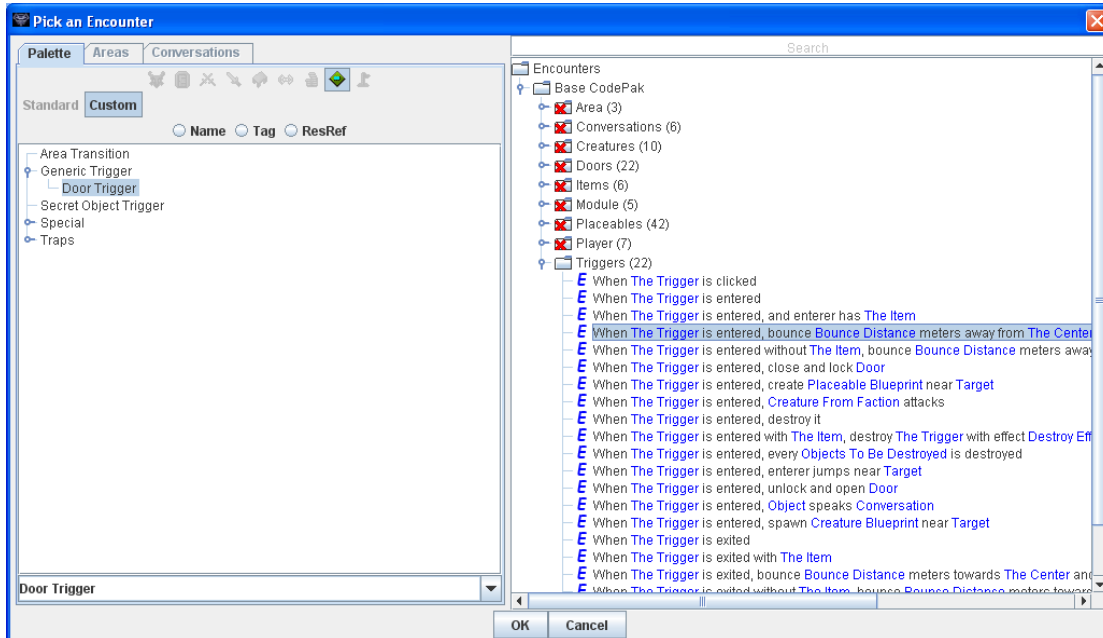
21) Create a waypoint “Trigger Centre” at the centre of the trigger you just created. This waypoint centre is needed in ScriptEase.

Scripting the Trigger:

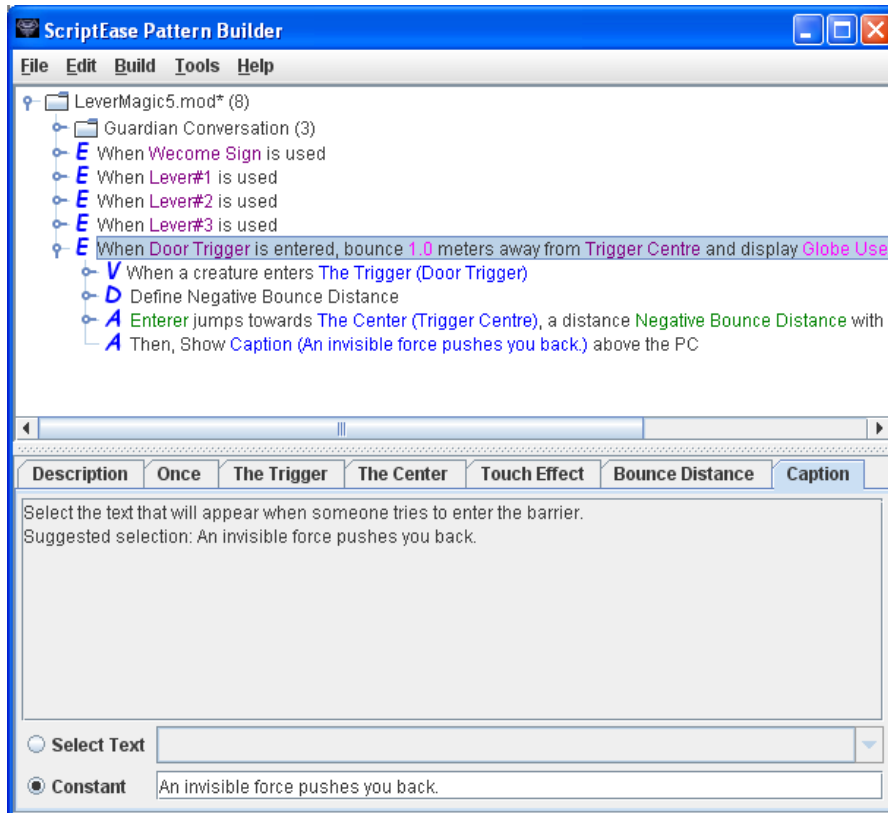
22) Save and close your module in Aurora and open it in ScriptEase.

23) Right-click LeverMagic5.mod and select New Specific Encounter.

24) Select Encounters → Base CodePak → Triggers → When **The Trigger** is entered, bounce **Bounce Distance** meters away from **The Center** and display **Touch Effect** and **Caption**. (I know, this is a very long name... but it will look nice after you finish.) On the left side, select Door Trigger. Confirm with the following screenshot and click OK.



25) Click the new encounter. In the “The Center” tab, select Module Blueprint → Pick..., and choose Trigger Centre. In the “Bounce Distance” tab, Constant → enter 1.0. In the “Touch Effect” tab, choose Globe Use (of course you can try other visual effects too, but Globe looks cool). In the Caption tab, enter “An invisible force pushes you back.” or something similar. Check with the following screenshot.



26) You can save and compile your module now. Test it to see the effect of the barrier. You shouldn't be able to get close to the vault door.

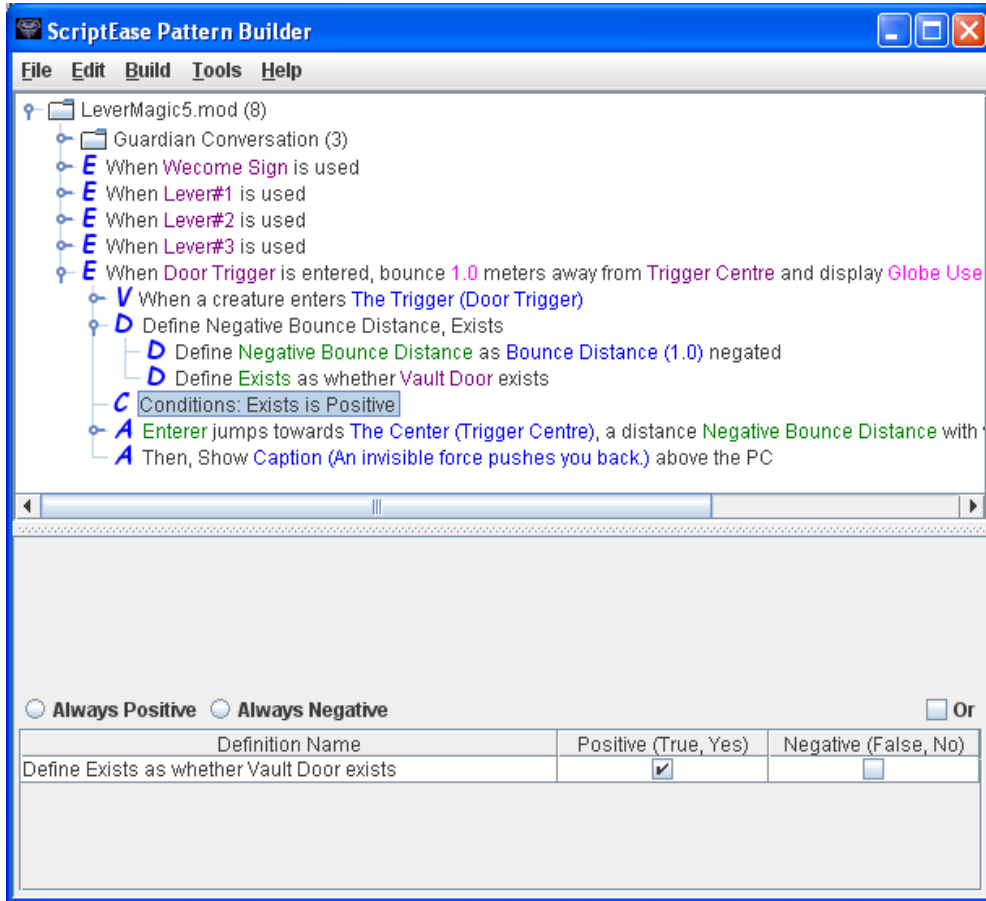
Disabling the trigger:

Now, how can you disable the trigger when certain conditions are satisfied?

27) To disable an encounter (in this case, the trigger-enter encounter, but it applies to any encounters), a *condition* on the encounter is needed. A condition requires a binary definition.

28) Expand the “When **Door Trigger** is entered...” encounter, right-click and Add a Definition → Binary → Testing Object Properties → Define **Exists** as whether **Object** exists. Under the “Object” tab → Module Blueprint → Pick..., select the Doors icon from the top row, and select Vault Door.

29) Now, we have our binary definition, we can create the condition. Right-click the encounter and select “Create MultiCondition”. Select the MultiCondition, and check the checkbox under “Positive” for “Define Exists as ...”



30) We are done. The trigger will only work if the Vault Door exists (the Vault Door is destroyed when you pull the levers). Save and Compile. Test your module. Have fun.