

Tutorial Implementing Multiple Quests in UT3

Written by Matthijs Rekers

Edited by Joris Olde Bijvank

<http://www.ladybirdgames.com>

In this tutorial we are going to expand the possibilities from the previous tutorials about making Quests in UT3.

In those tutorials we made it possible to:

- Pick up items
- Talk to a NPC in a basic (previously set-up) conversation
- Accept (or decline) and deliver a quest

Within this tutorial we are going to make multiple quests, and let Kismet control the lot. There are many things we need to think of when letting Kismet control loads of data, therefore in this tutorial I'm not going to show the locations of all the Kismet objects we are going to use, also I'm not going to show how to make an UI-scene, or tell you what text to write in it. This can all be found in our previously made tutorials:

- Basic Conversations
- Making UI_scenes
- Picking up and dropping items
- Making Quests part 1
- Making Quests part 2
- Making Quests part 3

What we are going to create is a serie of quests (about 20 in total) in three different quest lines, given by approximately 10 different quest givers and persons where you can deliver the quests to.

Basic Setup

When I created quests for our game Son of Argan, I did it in a basic field, where all the NPC's are standing next to each other. It makes programming a bit easier. When copying the NPC's to your main game all you need to do is alter the object references when you switch your characters for the game version, or every other change in objects you make. No big deal though. So, what we need is the following:

- Subtracted box (about 2048 x 2048 x 512) textured (makes it more playable when testing)
- Enough light in the room
- About ten NPC's in a row, lining up along one of the walls (Generic Browser > Actor. Classes tab > PhysAnimTestActor)
- One player start, away from the line with NPC's

After this we can start our 'production' work.

Making text announcements when you get close to an NPC

When you get close to an NPC we want a textual announcement on our screen that tells us what NPC we are about to talk to. Therefore we need a trigger just in front of each of the NPC's (make it Dynamic Trigger's, you never know when it comes in handy.)

In Kismet make a new sequence and name it Announcements.

Make a Trigger_Touch event for each of the triggers, and create announcement objects (New Action > Voice / Announcements > Play Announcements) behind every single trigger.

Type the name of each of the NPC's in theAnnouncementText area.

Make sure the right triggers are connected to the right Announcement objects!

Creating UI scenes for all quests.

What I did in my game was that I made 1UIscene with a standard layout. Inserted a text-title, a text-area and two options (buttons) for accepting or declining below the text area. That one I duplicated for each of the UI scene's I needed.

I made 7 quests in series 1 (**Quest A1 till A7**)
I made 3 quests in series 2 (**Quest B1 till B3**)
I made 10 Quests in series 3 (**Quest C1 till C10**)
and many more, but that is of no relevance here.

I named each of my UIscenes as following:

- A1pickup
- A1deliver
- A1pickup2 (in case I had too much content text for just one UIscene, A1pickup would then have no buttons for accepting or declining, only a continue option that leads to A1pickup2)

I made 2 more UIscenes,

- **QuestCompleted** A quest-completed UIscene (can be completely blank, we use it only for object lists.)
- **HiThere**. (I used this one whenever a trigger is used when no Quest UIscene can be called. The text goes "Hi there, what can I do for you?".

Inside the UIscenes there need to be some Kismet. We'll come to that later on.

Creating Light's for the Questgivers.

The person that plays your game wants to see a small hint where he can get or deliver his quest. In this tutorial we do it by simple turning a colored light above the NPC on or off. You can also use other options, but the method stays the same.\

What you need to do is create a SpotlightToggleable directly over one of the NPC's, give it a nice color (I choose blue for accepting quests), copy the spotlight and place it on exactly the same location as the first. Give it a different color (i choose red for delivering quests).

Make sure that both lights have the option 'Enabled' turned off, so that the light won't be visible at start. Create those two for every NPC. Now we need to create some Kismet.

- Create a Toggle object, and assign the blue light for accepting quest to the Target block.
- Create a second Toggle object and assign the red light for delivering quests to the Target block.
- Create two Remote Event objects for each of the Toggle objects
- Call the one (in Eventname) that turns on the Accept light **<NPCname>StartOn** (In my case: MasterCybaraStartOn) and connect it with the Toggle ON block
- Call the one that turns off the the Accept light **<NPCname>StartOff** (In my case: MasterCybaraStartOff) and connect it with the Toggle OFF block
- Call the one that turns on the Deliver light **<NPCname>EndOn** (in my case: MasterCybaraEndOn) and connect it with the Toggle ON block
- Call the one that turns off the the Deliver light **<NPCname>EndOff** (In my case: MasterCybaraEndOff) and connect it with the Toggle OFF block

Now that we have created these Remote Events, you can simply call each of the events in your UI scene Kismet to either turn the appropriate light on or off.

Calling the right UIscene with object lists

When a player walks up to a quest giver we want the appropriate UIscene to pop up. It can be either:

1. A quest-start UIscene
2. A quest-deliver UIscene
3. A quest-Hithere UIscene (when none of the above can be shown)

What we will make is a Kismet sequence that does the following:

- Kismet checks if the first possible quest is already completed.
- If completed the first possible quest, Kismet will ask the same question, for every following quest, until all quests are completed (then the event Hithere will be called)
- If along the row a quest is not completed yet (it can be the first, the third, the twentieth, the last, whatever). Kismet will continue to the next question.
- If a quest is not completed yet, Kismet will check if the quest is already started.
- If the quest is not started yet, Kismet will check if all conditions to start the quest are met(all previous quests completed, or whatever you want.
- If all are met, Kismet will open up the appropriate UIscene.
- if not, the Kismet will show the Hithere UIscnene.
- If the Quest is allready started, Kismet will check if the completion conditions are met.
- If so, the Quest completion UIscene will be called.
- if not, the the Hithere UIscene will be called (or another quest related UIscene if you wish)

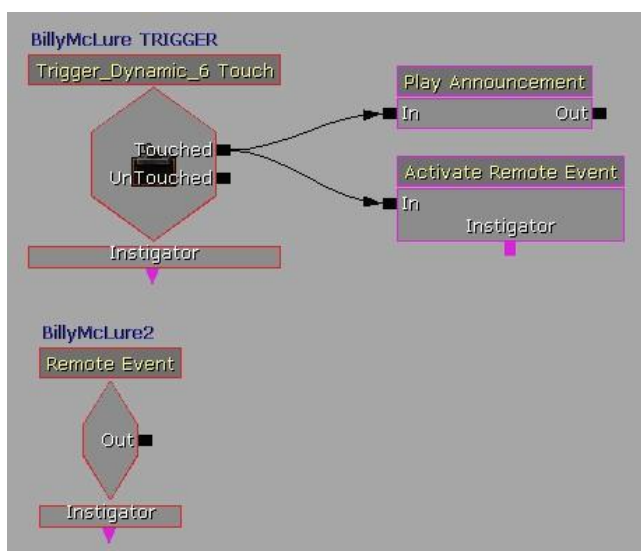
This goes for almost all quests. A lot of work is waiting! Let's get started.

Triggers

For the triggers we can use the same ones as the ones used for the announcements we made earlier. What we do is the following:

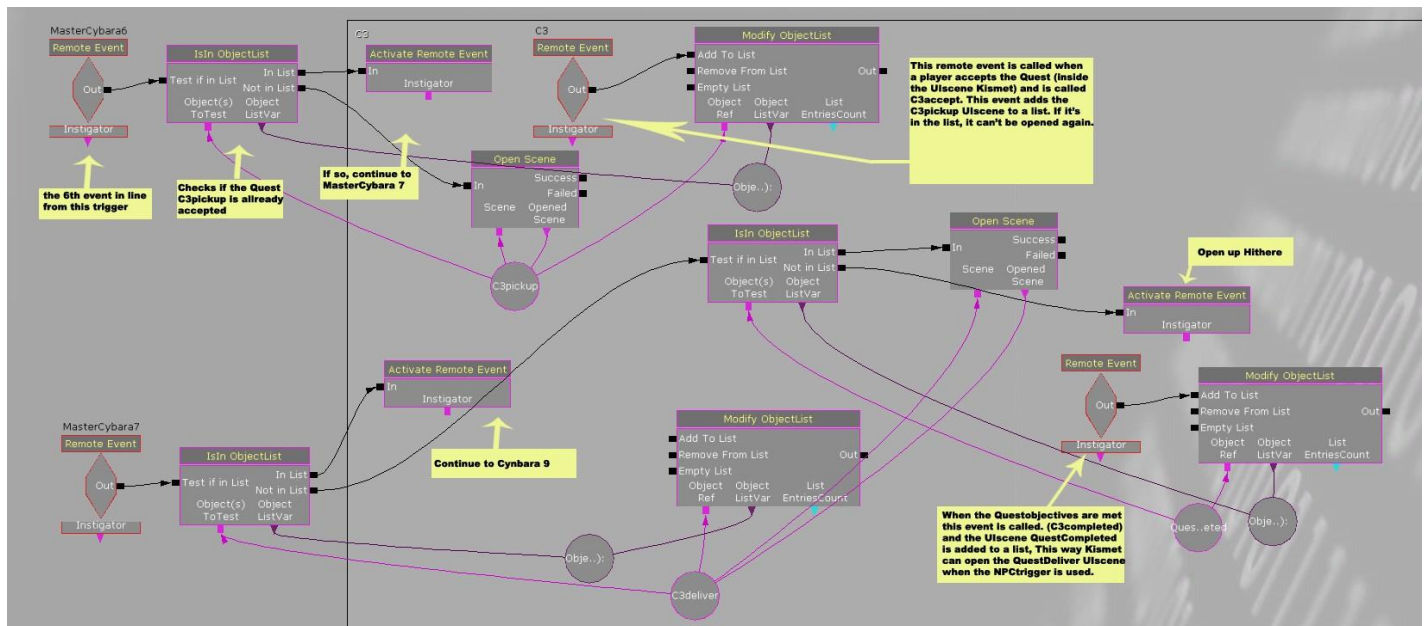
- Create an "Activate Remote Event" for each of the triggers, and set the event of the events to <name of the NPC>2 (in my case the trigger was from BillyMcLure, the ActivateRemoteEvent lead to BillyMcLure2)
- Later on during developing we make a remote event (BillyMcLure3, BillyMcLure4 etc etc) for each of the UIscenes that are possible with that particular NPC. That we we create a sequence where Kismet checks if the Quest belonging to Event2 is completed, and if so checks if the quest belonging to Event3 is completed, and if so... etc etc etc.

It should look like this:



Setting up the basic Quest sequence

Set up your Kismet as shown in the following image:



The sequence is not so complex as it seems. All that basically happens is multiple object list checks.

If you build your own sequence, start with a simple Quest series (serie A1, A2, A3...) where the next one can start if the first is completed, all given by one NPC.

What you should do is the following:

- create a Remote Event from the trigger belonging to the NPC. NPC2
- Let Kismet check if A1pickup is in an objectlist
- If so, activate remote event NPC3
- If not let Kismet open UIscene A1pickup
- Create a Remote Event that adds A1pickup to an objectlist (modify objectlist) (later on we will make the Kismet inside the UIscene that triggers the remote event that adds A1pickup to an object list)

For delivering the quest you must do the following:

- create a Remote Event from the trigger belonging to the NPC. NPC3
- Let Kismet check if A1deliver is in an objectlist
- If so, activate remote event NPC4
- If not let Kismet check if QuestCompleted is in an objectlist (inside the Quest-Kismet sequence you must call a remote event that adds QuestCompleted to an objectlist when all the Quest objectives are met.)
- When QuestCompleted is in the list, let Kismet open UIscene A1deliver
- If not, let Kismet open up UIscene Hithere.
- Create a Remote Event that adds A1deliver to an objectlist (modify objectlist) (later on we will make the Kismet inside the UIscene that triggers the remote event that adds A1deliver to an object list)

For each of the quests you do this. When the NPC where you deliver the Quest is another then the one you get the quest from, and there are no quest objectives (just go to this guy) you need to let Kismet check if the A1pickup UIscene is in the list before you let Kismet open up the A1 deliver UIscene. If you don't you will be able to deliver the Quest even before you have accepted it!

Basicly, this will make your Questing work.

Now all we need to do is call the appropriate events in the UIscene Kismet for the lighting and completion / accepting of Quests.

If you keep everything logically the same, you will need to do the following.

Inside the Kismet from a pickup UIscene, call the events:

- <NPCname>StartOff when accepted
- <NPCname>EndOn when accepted
- <Questnumber>Accepted when accepted (call the event in the standard Kismet that adds
- <Questnumber> to an objectlist so it can't be opened again.

Inside the Kismet from a deliver UIscene, call the events:

- <NPCname>EndOff
- <NPCname>StartOn when a quest can start directly after this one is completed. (if not, you will need to trigger this event somewhere else.)
- <questnumber>Completed when you need to complete different things before something can be triggered. (see next paragraph)

Complete multiple objectives before an event or Quest can be given / triggered

If for instance you have the following Questseries:

- D1 - D5 (D1pickup, D1deliver, D2pickup, D2deliver etc etc).
- E1 - E10
- F1 - F14
- G1 - G9

The following goes: The series D1 can only be started when the following objectives are met:

- E5 quest is completed and delivered
- F series is completely done
- G2 quest is accepted (but not necessary completed)

We can solve this the following way. We have a UIscene called QuestCompleted. we will need that one.

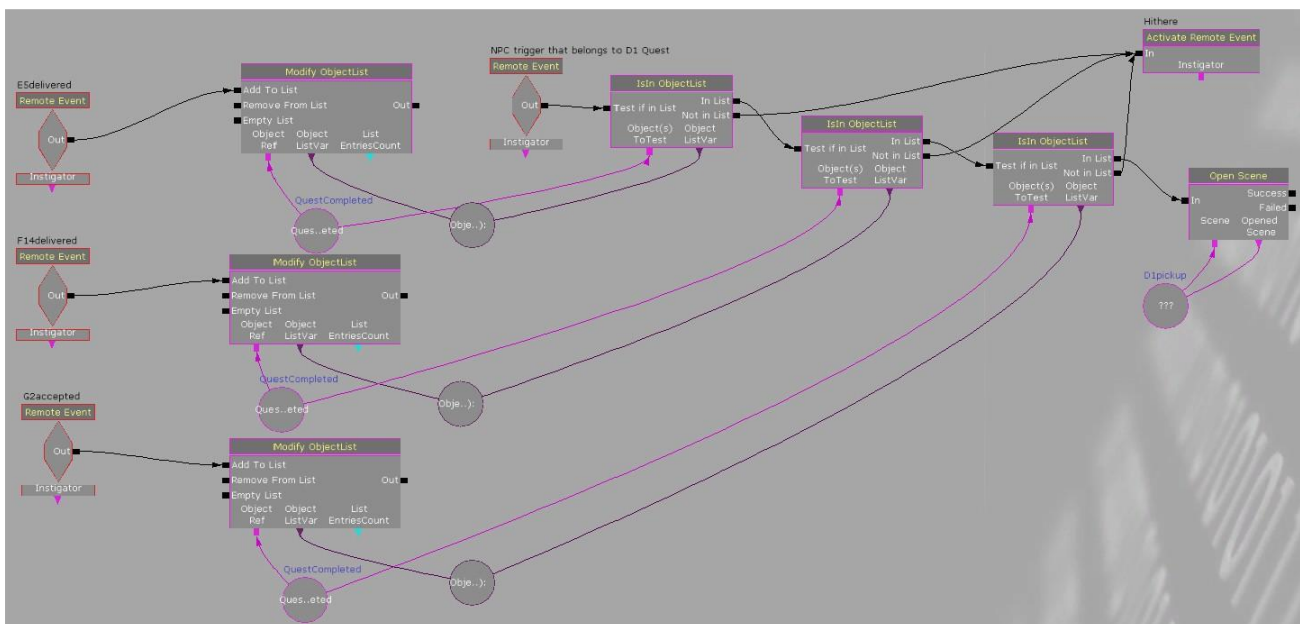
Create three Activate Remote Event objects.

1. place the first inside the UIscene Kismet from E5deliver. call it **E5delivered**
2. place the second inside the UIscene Kismet from F14delive. call it **F14delivered**
3. place the third inside the UIscene Kismet from G2pickup, and make sure that it will only be triggered when the quest is accepted (not when the UIscene is opened). call it **G2accepted**

Now create three Remote Event objects, and refer to the three as mentioned above.

Foreach of the events, create a ModifyObjectList object, and set it to Add QuestCompleted to an objectlist (you will end up with three RemoteEvent objects, three ModifyObjectList objects, three QuestCompleted object var's underneath the ModifyObjectlist object, and three object lists, each underneath it's own ModifyObjectlist object.

Furthermore you will have three IsInObjectlist object's that check if all the lists are filled. If so it opens up the 01pickup Uscene, if not, it will open up the Hithere Uscene. It will look like this:



Final words

Working with multiple quests can make things very complex. Make sure you use loads of ActivateRemoteEvent objects (this makes things a lot easier to overlook). Also, make sure you use logically chosen eventnames. If you have like 100 Quests, you will have up to 500 remote events. Imagine give 2 the same name accidently, and then figuring out where things went wrong...

Good luck,

Matthijs