Basic Quest Tutorial

In this tutorial we will create a basic quest that will involve NPC interaction. After completing this tutorial you should be able to easily expand it to include other conditions by looking at other existing quests for inspiration.

What you need:
QuestSystemJune2016-0.3.3.pdf (in the term 3 folder)
Your server working - While working on this quest do not upload it to the main server. We will all be doing our own at this stage.
For additional info watch this video : https://youtu.be/1UeyvNpEbVA

There are three main steps to creating this quest.
Step 1: Defining the quest to be added to the quest system.
Step 2: Writing the NPC ai to handle the quest.
Step 3: Creating the NPC template to add to attach the script to.

Important things to know

The Quest system in stored in a global table for the server to use. This means that any changes to the quest config requires a server restart for them to take effect.

NPC ai changes can be reloaded by reloading the script in the NPC.

Quest System Basics

Quest System Elements

Current quest system is linear. However it can be changed to a branching system by the use of the use of quest message triggers.

Creating the files to support our quest

We are using an override script to manage all our mods.
Your quest definition file will need to be created in the GSCC/scripts/yourName folder.

Add your script to the require list in the data_quests.lua file in the GSCC directory.
E.g. require 'staples.testQuest' -- Adds the testQuest to the quest system

NPC ai scripts will need to be in the GSCC/scripts folder

Template files will need to go in the GSCC/templates folder

Defining our Quest

Quest information is defined in the AllQuests table defined in the file Data_quests.lua. We will need to add our quest definition to this table. This is done with the following syntax

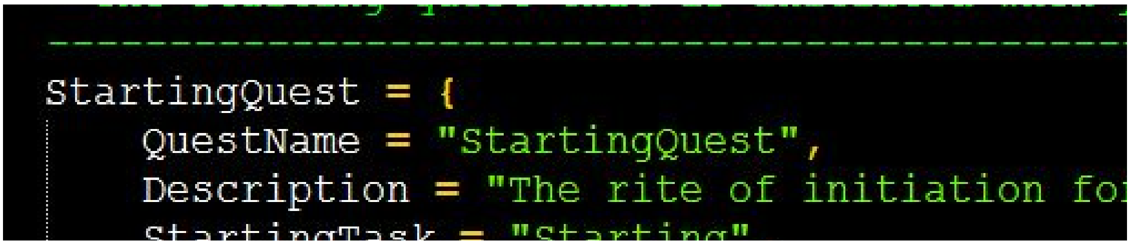
AllQuests.theNameOfTheQuest = {

Each defined quest has two main parts. The quests definition data and a list of task definitions.

The quest definition data has the following components as defined in the QuestSystem PDF.

```
        QuestName = "theNameofTheQuest",
        StartConditions = {"EnterQuestStartConditionHere"}, -- OPTIONAL - See below
        StartConditionArgs = {Some start conditions require variables},
        Description = "This is the in game description for your quest",
        StartingTask = "theNameOfTheQirstTaskInTheQuestChain",
        QuestDisplayName = "[ECD905]The name of the quest in the quest tracker[-]",
        Repeatable = true, -- OPTIONAL (makes quest repeatable)
        QuestCompleteMessage = "This is the pretty message that pops up when you finish.",
```

IMPORTANT: The name of the quest in the table and the Quest name need to be the same!

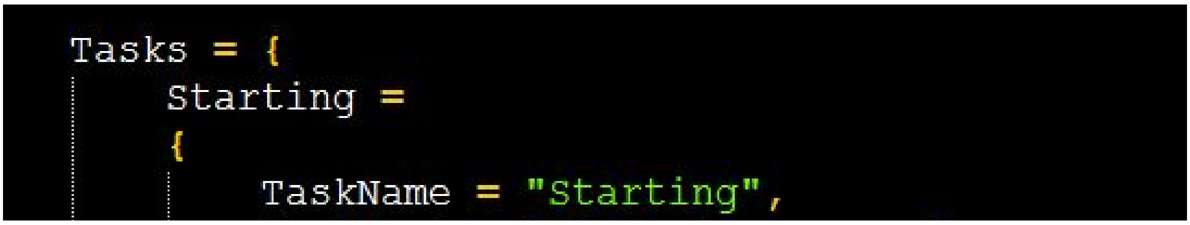


```
------------------------------------------------
StartingQuest = {
    QuestName = "StartingQuest",
    Description = "The rite of initiation fo
    StartingTask = "Starting"
```

Task Definitions

This is then followed by a list of task definitions also stored in a table.{} are used to deifine a table.

Tasks = {... tasklist ...}



```
Tasks = {
    Starting =
    {
        TaskName = "Starting",
```

IMPORTANT: Notice TaskName is the same in both places.

theNameOfTheTask = {
TaskName = "theNameOfTheTask",
Description = "Task description displayed in game",
Hint = "A hint to the player that can be displayed in game. As a start action",
DialogTitle = "This is the title of the message box for the dialog and NPCdialog action",
DialogText = "This is the text that goes in the message box for the dialog and NPCdialog action",
NextTask = "theNameOfTheNextTaskInTheQuestChain - OPTIONAL",
MapMarkers = {Used to configure the icons on the player map},
IsQuestEnding = false, -- is this the last quest in the chain?
StartActions= {Events that happen at beginning of a task - See Quest Task Actions in PDF},
FinishActions={Events that happen at end of a task - See Quest Task Actions in PDF},
CompletionConditions = {This is a list of completion conditions, separated by, all need to return true for the quest to advance},
TaskObjVarName = "theNameOfTheObjVarUsedForCompletionCondition",
OtherStuff = “You may need more info here depending on what completion conditions you use. Reward, reward choices, reward list etc check the pdf for the action requires.”
}

QuestCallBacks and advancing a quest

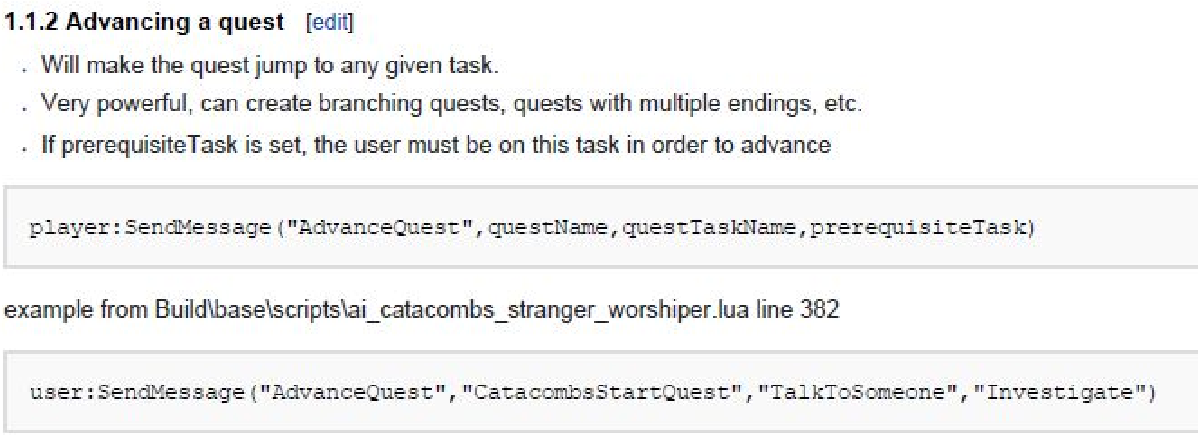
These are the condition functions that you specify in the completionConditions of the task definition. These are detailed in the base_quest_sys.lua file and documented in the PDF. If one you want isn’t there then create it and add it to the table using the same method you added your quest to the AllQuests table.

Handling quest advancement manually

The system will only automatically advance a linear quest chain. To advance a branching quest you need to do it manually. The following event handlers are loaded on all players and can be triggered, by sending a message to the player, to perform different tasks related to advancing a quest.

Quest related event handlers loaded on characters

-- advances the task
RegisterEventHandler(EventType.Message,"AdvanceQuest",AdvanceTask)



1.1.2 Advancing a quest [edit]

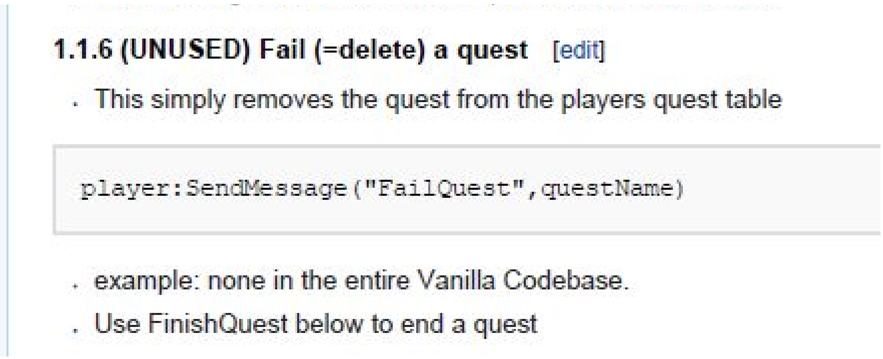
- Will make the quest jump to any given task.
- Very powerful, can create branching quests, quests with multiple endings, etc.
- If prerequisiteTask is set, the user must be on this task in order to advance

```
player:SendMessage("AdvanceQuest",questName,questTaskName,prerequisiteTask)
```

example from Build\base\scripts\ai_catacombs_stranger_worshiper.lua line 382

```
user:SendMessage("AdvanceQuest","CatacombsStartQuest","TalkToSomeone","Investigate")
```

RegisterEventHandler(EventType.Message,"FailQuest",FailQuest)
This is unused in the current quests



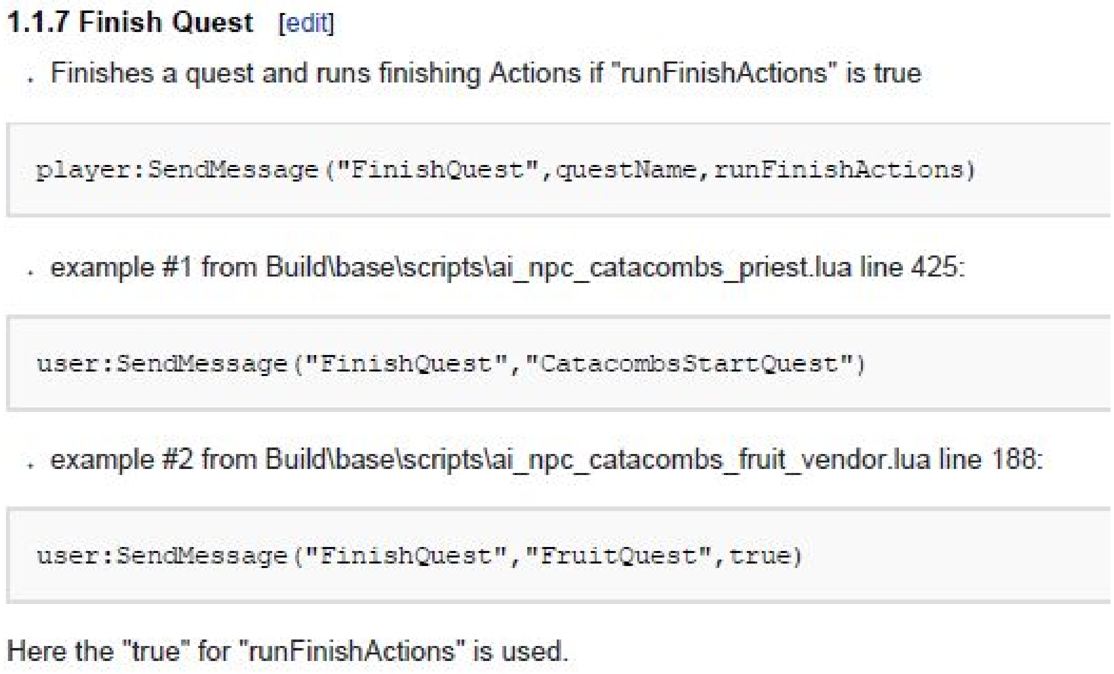
1.1.6 (UNUSED) Fail (=delete) a quest [edit]

- This simply removes the quest from the players quest table

```
player:SendMessage("FailQuest",questName)
```

- example: none in the entire Vanilla Codebase.
- Use FinishQuest below to end a quest

RegisterEventHandler(EventType.Message,"FinishQuest",FinishQuest)



1.1.7 Finish Quest [edit]

- Finishes a quest and runs finishing Actions if "runFinishActions" is true

```
player:SendMessage("FinishQuest",questName,runFinishActions)
```

- example #1 from Build\base\scripts\ai_npc_catacombs_priest.lua line 425:

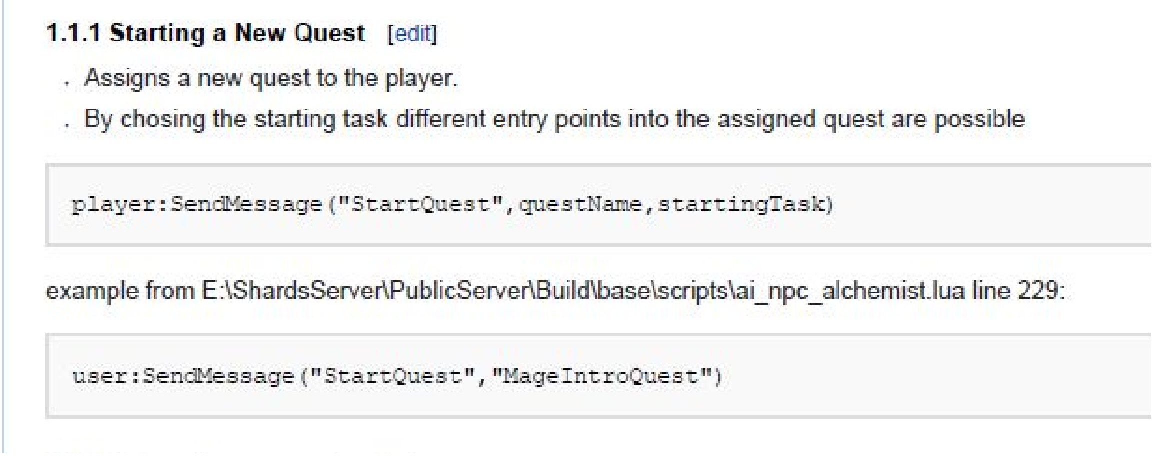
```
user:SendMessage("FinishQuest","CatacombsStartQuest")
```

- example #2 from Build\base\scripts\ai_npc_catacombs_fruit_vendor.lua line 188:

```
user:SendMessage("FinishQuest","FruitQuest",true)
```

Here the "true" for "runFinishActions" is used.

--message to send to trigger a new quest
RegisterEventHandler(EventType.Message,"StartQuest",StartNewQuest)



1.1.1 Starting a New Quest [edit]

- Assigns a new quest to the player.
- By chosing the starting task different entry points into the assigned quest are possible

```
player:SendMessage("StartQuest",questName,startingTask)
```

example from E:\ShardsServer\PublicServer\Build\base\scripts\ai_npc_alchemist.lua line 229:

```
user:SendMessage("StartQuest","MageIntroQuest")
```

NPC AI

This will only cover a very basic setup.
NPC interactions can be quite complex this will only create a very basic dialog to show how to set a variable and advance a quest.

REQUIRED AT THE START OF THE SCRIPT
require 'base_ai_npc'

AI.Settings.MerchantEnabled = false -- is the NPC a merchant
AI.Settings.EnableTrain = false -- will the NPC train skills
AI.Settings.EnableBuy = false -- will the npc buy stuff from the player
AI.Settings.SetIntroObjVar = true -- When first talking to the NPC set "have talked to variable to true"
AI.Settings.StationedLeash = true -- will the NPC stay still.

```
NPCTasks = { -- required but not explained in here.
}

-- This is a standard function on nearly all NPC’s it gives the characters intro dialog and then
-- (in base_ai_npc) sets a variable on the character so that players don’t get it again. The
-- variable is strored in the formart intro|Character Name
-- you can check for this variable as part of your quest triggers if you want.
function IntroDialog(user)

   QuickDialogMessage(user,"Hi! this is a test message you have spoken to me!")
        this:ScheduleTimerDelay(TimeSpan.FromSeconds(3),"testTimer",user)
end

--this is an event
RegisterEventHandler(EventType.Timer,"testTimer",
        function(user)
                user:SetObjVar("testQuestVariable", true)
        end)
```