The Vanilla Quest System

Contents [hide]

1 The vanilla quest system [edit]

```
File: base_quest_sys.lua
Version: 0.3.3
```

1.1 Interface [edit]

The interface consists of messages being sent to the player. There are 4 important messages:

- StartQuest
- AdvanceQuest
- UpdateQuestUI
- FinishQuest

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")
```

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")
```

1.1.3 (UNUSED) Set the active (tracked) quest in the Quest Tracker UI [edit]

- Changes the quest tracker to the Quest "questName" and updates the Tracker

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

Example: This Message is never used in Vanilla code, but the handler function is called from other handlers.

1.1.4 (BARELY USED) Refresh Quest Tracker UI with last active quest [edit]

- This one just calls SetActiveQuest with the last active quest from ObjVar

```
player:SendMessage("RefreshQuestUI")
```

example: build\base\scripts\player.lua line 1137

```
this:SendMessage("RefreshQuestUI")
```

This also is the only use inside the Vanilla code - so it's just for login handling.

1.1.5 Update Quest Tracker UI [edit]

- Redraws the Tracker with the given Quest "questName"
- When "isTask" is true the given Quest is handled as task. Must be set correctly or bad stuff will happen.

```
player:SendMessage("UpdateQuestUI",questName,isTask)
```

example from build\base\scripts\base_quest_sys.lua line 551

```
this:SendMessage("UpdateQuestUI",questName)
```

- This is used inside the function which initiates a new quest of name <questName>.
- This message can be used to to update the QuestTracker.

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

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.

1.2 Quest Task Actions [edit]

```
QuestTaskActions = {}
```

Functions that get called when a task is finished or initiated in

- StartActions or
- FinishActions

1.2.1 ConsumeItems [edit]

```
ConsumeItems = function (user,task)
    backpackObj:ConsumeItemsInContainer(task.TaskConsumeItemList,task.TaskForceConsumeItems)
```

1.2.2 ObjVar [edit]

```
ObjVar = function (user,task)
    if task.TaskDeleteObjVar ~= nil and task.TaskDeleteObjVar ~= false then
    user:DelObjVar(task.TaskObjVarName)
```

1.2.3 SetObjVar [edit]

```
SetObjVar = function (user,task)
    user:SetObjVar(task.TaskSetObjVar,task.TaskSetObjVarValue)
```

1.2.4 Resource [edit]

```
Resource = function (user,task)
    user:RequestConsumeResource(task.ResourceRequired,task.ResourceAmount,"QuestConsumeResource",this)
```

1.2.5 ObjVarValue [edit]

```
ObjVarValue = ObjVar,
```

1.2.6 SetObjVar [edit]

```
SetObjVar = function (user,task)
    user:SetObjVar(task.TaskSetObjVar,task.TaskSetObjVarValue)
```

1.2.7 Dialog [edit]

```
Dialog = function (user,task) --throw up dialog
    NPCInteraction(task.DialogText,nil,user,"Responses",response,task.DialogTitle)
```

1.2.8 NPCDialog [edit]

--THIS IS NOT GUARENTEED TO THROW A DIALOG UP IF YOU'RE OUTSIDE OF RANGE --throw up npc dialog

```
NPCDialog = function (user,task)--throw up dialog
    local response = task.DialogResponses or Template:Text="Ok.",handle="Ok"
    local npc = FindObject(SearchMulti({ SearchMobileInRange(5), SearchTemplate(task.NPCTemplate), }))
    NPCInteraction(task.DialogText,npc,user,"Responses",response,nil,NPC_SEARCH_RANGE)
```

1.2.9 PlayEffect [edit]

--play an effect

```
PlayEffect = function(user,task)
```

this:PlayEffect(task.**EffectToPlay**) -- No EffectDuration Given !!!!

1.2.10 CreateItem [edit]

--AS A RULE OF THUMB, ONLY CALL THIS IF YOU KNOW YOU WILL BE IN PROXIMITY TO A GIVEN NPC --create object lists

```
CreateItem = function(user,task)
    local creationTemplate = task.Reward or task.RewardItem or task.TaskItem or task.Item
    ... give item ...
    this:SystemMessage("[D7D700]You have received a "..name,"event")
```

1.2.11 RandomItem [edit]

```
RandomItem = function(user,task)
    local creationTemplate = task.RewardChoices[math.random(1,#task.RewardChoices)]
    ... give random item ...
    this:SystemMessage("[D7D700]You have received a "..name,"event")
```

1.2.12 CreateItems [edit]

--AS A RULE OF THUMB, ONLY CALL THIS IF YOU KNOW YOU WILL BE IN PROXIMITY TO A GIVEN NPC

```
CreateItems = function (user,task)
     for i,j in pairs (task.RewardItems) do
     ... give stuff ...
```

1.2.13 ConsumeItemsWithObjVar [edit]

--AS A RULE OF THUMB, ONLY CALL THIS IF YOU KNOW YOU WILL BE IN PROXIMITY TO A GIVEN NPC

```
ConsumeItemsWithObjVar = function (user,task)
     local items = FindObjects(SearchMulti({SearchContainer(backpackObj),
     SearchHasObjVar(task.ItemObjVarName)  }    )  )
     ... Destroy All items ...
```

1.2.14 CreateObjects [edit]

--create random mobiles or other random objects **CreateObjects** = function(user,task)

```
    for i=0,task.SpawnAmount,1 do
```

CreateObj(task.**SpawnTemplates**[math.random(1,#task.**SpawnTemplates**)], task.**SpawnLocations**[math.random(1,#task.**SpawnLocations**)],"created")

1.2.15 SendMessageToNPC [edit]

--send a message to an object

```
SendMessageToNPC = function(user,task)
        local npc = FindObject(SearchMulti({SearchTemplate(task.NPCTemplate)}))
        npc:SendMessage(task.TaskMessage,task.TaskMessageArgs)
```

1.2.16 SendMessageToPlayer [edit]

--send a message to a player

```
SendMessageToPlayer = function(user,task)
        user:SendMessage(task.TaskMessage,task.TaskMessageArgs)
```

1.2.17 Hint [edit]

--show a hint to a player

```
Hint = function(user,task)
    user:SendMessage("ShowHint",task.Hint)
```

1.3 Quest Callbacks [edit]

```
QuestCallbacks = { ... callback function list ...}
```

- Callbacks have <user>,<task> as parameters and additional optional parameters.
- They are used to check Quest/Task entry/exit conditions.
- They return true or false.

1.3.1 NPCInteraction [edit]

True when interacting with an NPC made from a given template

```
NPCInteraction = function(user,task,templateId)
        if templateId == task.NPCTemplate then return true ....
```

1.3.2 HaveFaction [edit]

True when having a minimum reputation with a given faction

```
HaveFaction = function (user,task)
        if (GetFaction(user,task.FactionName) >= task.FactionAmount) then return true ....
```

1.3.3 HaveItemWithObjVar [edit]

True when having an item with a certain ObjVar set. ("Flagged Item")

```
HaveItemWithObjVar = function (user,task)
 ... local items = FindObjects(SearchMulti({{
          SearchContainer(backpackObj),
          SearchHasObjVar(task.ItemObjVarName)}}
          ))
```

1.3.4 HasItemInList [edit]

True when having an item with a template from a given list of item templates.

```
HasItemInList = function (user,task)
... for i,j in pairs(items) do
        for i,itemId in pairs(task.ItemList) do ....
            if (j:GetCreationTemplateId() == itemId) then return true
```

1.3.5 HasPet [edit]

True when having any pet.

```
HasPet = function (user,task)
                if (user:HasObjVar("Minions")) then return true
```

1.3.6 KnowRecipe [edit]

True when knowing a given recipe.

```
KnowRecipe = function(user,task)
        if (HasRecipe(user,task.TaskRecipe)) then return true
```

1.3.7 Items [edit]

True when having the required amount of a list of items. This allows doing complex collector quests.

```
Items = function (user,task)
        return user:GetEquippedObject("Backpack"):HasItemsInContainer(task.TaskItemList)
```

1.3.8 ObjVar [edit]

True if the user has a given ObjVar (ObjVar basically is a flag)

```
ObjVar = function (user,task)
        if (user:HasObjVar(task.TaskObjVarName)) then return true
```

1.3.9 ObjVarValue [edit]

True if the user has a given ObjVar with a given value (ObjVar basically is a flag)

```
ObjVarValue = function (user,task)
        if (user:GetObjVar(task.TaskObjVarName) == task.TaskObjVarValue) then return true
```

1.3.10 ObjVarValueMin [edit]

True if the value of a given ObjVar is at least as big as the given threshold

```
ObjVarValueMin = function (user,task)
        if (user:GetObjVar(task.TaskObjVarName) >= task.TaskObjVarValue) then return true
```

1.3.11 Resource [edit]

True when the player has the required amount of a given resource.

```
Resource = function (user,task)
        if user:GetEquippedObject("Backpack"):CountResourcesInContainer(task.ResourceRequired)
             >= task.ResourceCount then return true
```

1.3.12 ExitRegion [edit]

True when the player is NOT in a given region. It is not actually leaving ...

```
ExitRegion = function(user,task)--left region
        if (not user:IsInRegion(task.TaskRegion)) then return true
```

1.3.13 EnterRegion [edit]

True when the player is in a given region. It is not actually entering ...

```
EnterRegion = function(user,task)--entered region
        if (user:IsInRegion(task.TaskRegion)) then return true
```

1.3.14 EnterRegions [edit]

True when the player is in ANY of the given region. It is not actually entering ...

```
EnterRegions = function(user,task) --left region
        for i,j in pairs(task.TaskRegions) do
            if (user:IsInRegion(j)) then return true
```

1.3.15 CompleteOnEnter [edit]

Always returns true ... WTF ???

```
CompleteOnEnter = function(user,task)
        return true
```

1.3.16 HaveSkillLevel [edit]

True if the player has reached a minimum skill level in a given skill.

```
HaveSkillLevel = function (user,task)
        if (this:GetSkillLevel(task.TaskSkillType) >= task.TaskSkillLevel) then return true
```

1.3.17 HasCombatStance [edit]

True if the player is in a given combat stance.

```
HasCombatStance = function (user,task)
        if (this:GetSharedObjectProperty("CombatStance") == task.StanceRequired) then return true
```

1.3.18 GainSkillLevel [edit]

True if the player has levelled in a given skill.

```
GainSkillLevel = function(user,task)
        local lastSkillLevel = mLastSkillLevel or this:GetSkillLevel(task.TaskSkillType)
        mLastSkillLevel = this:GetSkillLevel(task.TaskSkillType)
        if (lastSkillLevel < mLastSkillLevel) then  return true
```

1.4 Making a quest: The Questtable Structure [edit]

```
File: data_quests.lua
Version: 0.3.3
```

The Quests Main Table:

```
AllQuests = {
    HomoSapiensQuest={... QuestData ...},
    Quest_02={... QuestData ...},
    Quest_03={... QuestData ...},
    Quest_04={... QuestData ...},
    Quest_05={... QuestData ...},
    Quest_06={... QuestData ...},
}
```

1.4.1 A quest inside the Main Quests Table [edit]

Here defined outside the main quests table, using the dot notation. Thats is the way how we modularize data_quests.lua on Arcanima.

```
AllQuests.HomoSapiensQuest = {
    QuestName = "HomoSapiensQuest ",
    StartConditions = {"EnterRegion"}, -- OPTIONAL - See below
    StartConditionArgs = {TaskRegion = "FollowerHubRegion"}, -- OPTIONAL - See below
    Description = "This is an example description for a silly quest",
    StartingTask = "TalkToTheQuestgiverApe",
    QuestDisplayName = "[ECD905]Do the most stupid quest ever[-]",
    QuestPlayerScript = "quest_player_starting", -- THIS IS ONLY USED IN THE STARTING QUEST. NO SUCH SCRIPT EXISTS !!!
    Notify = false,  -- THIS IS ONLY USED IN THE STARTING QUEST. Prolly good for nuffin' ~Yorlik
    Repeatable = true, -- OPTIONAL (makes quest repeatable)
    QuestCompleteMessage = "You have proven to be a braindead individual which is stupidly following arbitrary
                           instructions. You must be of the species homo sapiens.",
    Tasks = {... tasklist ...}
}
```

1.4.1.1 Fields of the Quest Structure [edit]

- **QuestName** = String. Name of the Quest. Is the same like the handle of the Quest inside the Quests table.
- **Description** = String. Description of the quest. Used in the quest window as description.
- **StartingTask** = String. Handle of the starting task in the tasklist.
- **QuestDisplayName** = String. Quest Title shown in the quest tracker.
- **Repeatable**(optional) = Bool (true/false). Make a quest repeatable if true.
- **QuestCompleteMessage** = String. When the Quest is completed this text is displayed in the Message popping up.
- **StartConditions**(optional) = {"EnterRegion"} Callback to use for checking the conditional. See Quest Callbacks above.
- **StartConditionArgs**(optional) = {TaskRegion = "FollowerHubRegion"} - Given as arguments to the callback. See Quest Callbacks above.
- **Tasks** = A table holding the individual steps (=tasks) of a quest

1.4.2 The tasklist inside the HomoSapiensQuest Table [edit]

Here defined outside the main quest table, using the dot notation:

```
AllQuests.HomoSapiensQuest.Tasks= {
    TalkToTheQuestgiverApe = {... Task Data ...},
    TalkToTheQuestgiverApeAgain = {... Task Data ...},
    TalkToTheQuestgiverApeAgainAndAgain = {... Task Data ...},
    TalkToTheQuestgiverApeALastTime = {... Task Data ...},
}
```

1.4.3 A single task Structure [edit]

Here defined outside the tasklist table, using the dot notation:

```
AllQuests.HomoSapiensQuest.Tasks.TalkToTheQuestgiverApe ={
    TaskName = "TalkToTheQuestgiverApe",
    Description = "Talk to the ale drinking ape in the Tavern",
    Hint = "Be a Homo Sapiens and behave like one.",
    DialogText = "You feel unsecure about being a machine or a living being. You should speak to a monkey. Preferably a drunken one.",
    DialogTitle = "Passing the Turing test...",
    NextTask = "TalkToTheQuestgiverApeAgain",
    IsQuestEnding = false,
    StartActions= {}, -- Starting and Finishing Actions - See Quest Task Actions doc above
    FinishActions={},
    CompletionConditions = {
        "ObjVar",
        },
    TaskObjVarName = "Intro|TheDrinkingApe",
}
```

1.4.3.1 Fields inside the task structure [edit]

- **TaskName** = Name of the Task. Same as the handle in the tasklist.
- **Description** = Short description of the task shown in the quest tracker.
- **NextTask** = Next task after the current task is finished. Using the "AdvanceQuest" message in the AI/Dialogue System this can be overridden
- **IsQuestEnding** = (optional) Bool (true/false). Is this task ending the quest?
- **CompletionConditions** = table. Conditions defining the end of the task. Depends on the Quest Callbacks above.
- **TaskObjVarName** = Parameter for the callback "ObjVar" in the CompletionConditions. See Callback documentation above.
- **Hint** = Hint shown by the "Hint" Quest Task Action in Start of Finish Actions
- **DialogText** = Dialog text for the "Dialog" or "NpcDialog" Quest Task Actions
- **DialogTitle** = Dialog title for the "Dialog" or "NpcDialog" Quest Task Actions
- **StartActions** = Starting Actions for the task. See Quest Task Action doc above

example with 2 actions :

```
StartActions = {"Hint", "SetObjVar"}
```

- **FinishActions** = "" Like StartActions, just for for exit ...

Category: HowTo