Simple Quest Manager Manual

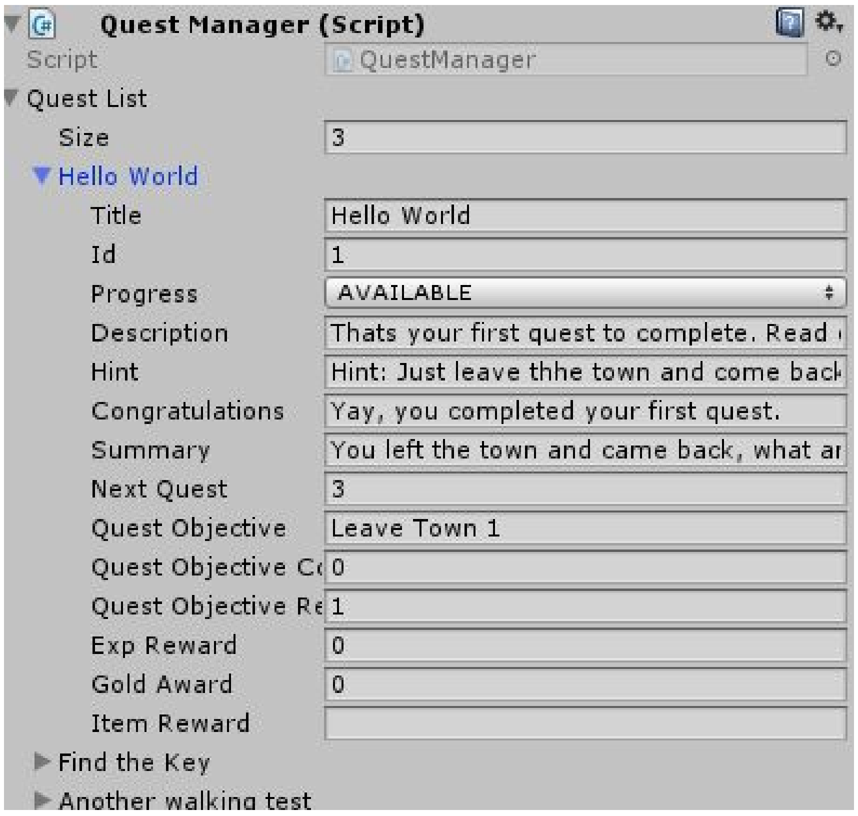
Glossary

Quest: Is a quest you can design to your needs. It contains strings for the description, hints, IDs, unlockable Quests, Rewards and so on.
QuestObjective: The Goal to achieve. Can be anything you can make contact with, destroy, collect. A string is used at the moment. You can also use an integer called ID.
QuestObject: Can be an NPC Character, any Object like a Book, a Bulletin Board or anything you can get quests from, or bring to.
QuestMarker: You can choose your own graphics of cause.

Setup Guide

Scenes: make sure to add the **QuestManager** and **Quest UIManager** Prefabs into all scenes where you need them.
Example: If you have NPC's in a town you need the QuestManager in it, as well as the UIManager.
Since the QuestManager is persistent you don't need it in a Battle Scene.

1.) The Quest Manager:
The Quest Manager is kind of the heart of the system. In here you design all your quests, and tell how many you have.


Quest Manager (Script)
Script: QuestManager
Quest List
Size: 3
Hello World
Title: Hello World
Id: 1
Progress: AVAILABLE
Description: Thats your first quest to complete. Read
Hint: Hint: Just leave thhe town and come back
Congratulations: Yay, you completed your first quest.
Summary: You left the town and came back, what ar
Next Quest: 3
Quest Objective: Leave Town 1
Quest Objective C: 0
Quest Objective R: 1
Exp Reward: 0
Gold Award: 0
Item Reward:
Find the Key
Another walking test

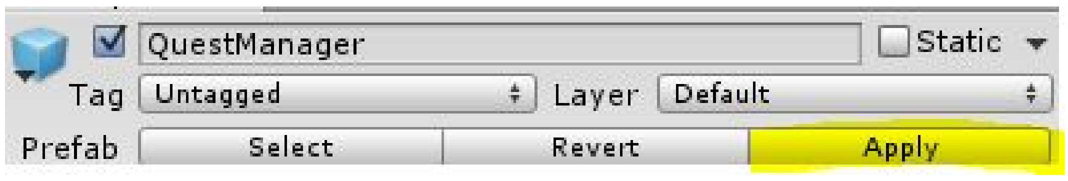
Size: Amount of Quests
Title: The title of a quest.
ID: The identifier of the quest, make sure all quest have a unique!
Progress: The current state of the quest. Only set them to NOT_AVAILABLE or AVAILABLE!

1. NOT_AVAILABLE: If this quest is unlockable later on, that's the state for.
2. AVAILABLE: This quest is available for whichever NPC has it.
3. ACTIVE: The quest has been taken, but is not completed yet. (QuestManager takes care of it.!)
4. COMPLETE: The quest is complete but not brought back to the NPC. (QuestManager takes care of it.!)
5. DONE: The has been brought back to NPC. (QuestManager takes care of it.!)

Description: Describe the Quest so people may know what to do.

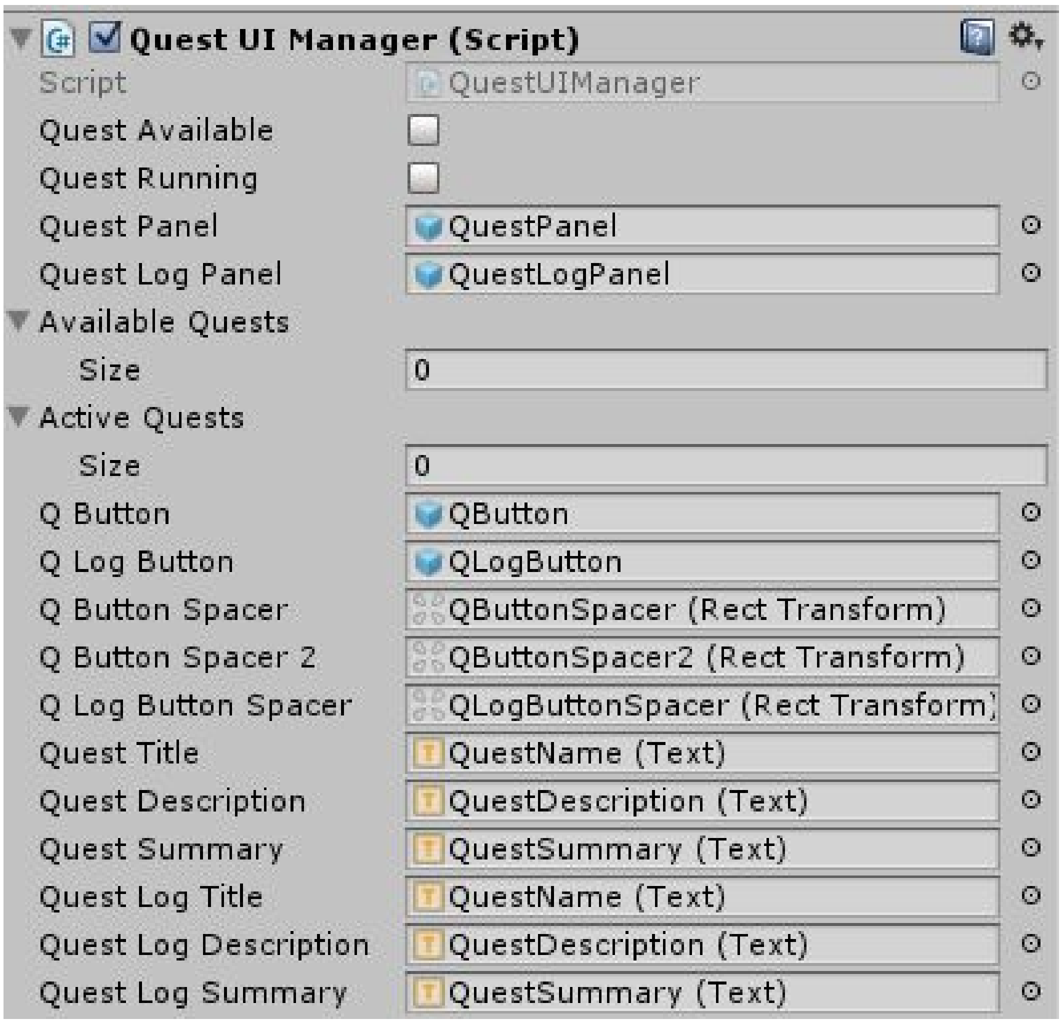
Hint: Will be shown, once the Quest is ACTIVE and the Player talks to the Quest Receiver, or looks in the QuestLog
Congratulation: Will be shown, once the Quest is COMPLETE and the Player talks to the Quest Receiver, or looks in the QuestLog
Summary: Not yet completely implemented yet.
Next Quest: Represents the next Quest ID to be unlocked(form NOT_AVAILABLE to AVAILABLE) Chain Quests!
Quest Objective: A string parameter for the objective, like enemy1. needed to send the correct data later back to the QuestManager. ID's would work as well.
Quest Objective Count: The current amount of the requested quest objective, should always be 0 at start. Example: Amount of killed enemies or picked up items.
Quest Objective Request: The requested amount of the quest objective.
Exp Reward: Not fully implemented since everyone has a different system. You need to enter your own functionality, depending on your systems.
Gold Reward: Not fully implemented since everyone has a different system. You need to enter your own functionality, depending on your systems.
Item Reward: Not fully implemented since everyone has a different system. You need to enter your own functionality, depending on your systems.

Once everything is setup here, make sure to apply the changes to the prefab!



2.) The Quest UI Manager:

The UI Manager will take care of all visible content. So when you talk an NPC, it will show the QuestPanels, and shows the QuestLog when pressing the corresponding button.
In this case it's **Q**. You can change it at any time.
If you want to change the look to fit your design, make sure you check the Prefab out first, or take the pieces used already and fit the design to your needs.



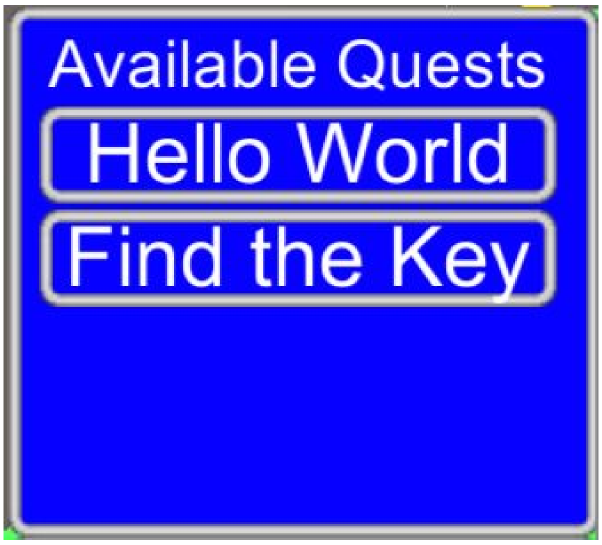
Quest UI Manager (Script)

Script	QuestUIManager
Quest Available	☐
Quest Running	☐
Quest Panel	QuestPanel
Quest Log Panel	QuestLogPanel
Available Quests	
Size	0
Active Quests	
Size	0
Q Button	QButton
Q Log Button	QLogButton
Q Button Spacer	QButtonSpacer (Rect Transform)
Q Button Spacer 2	QButtonSpacer2 (Rect Transform)
Q Log Button Spacer	QLogButtonSpacer (Rect Transform)
Quest Title	QuestName (Text)
Quest Description	QuestDescription (Text)
Quest Summary	QuestSummary (Text)
Quest Log Title	QuestName (Text)
Quest Log Description	QuestDescription (Text)
Quest Log Summary	QuestSummary (Text)

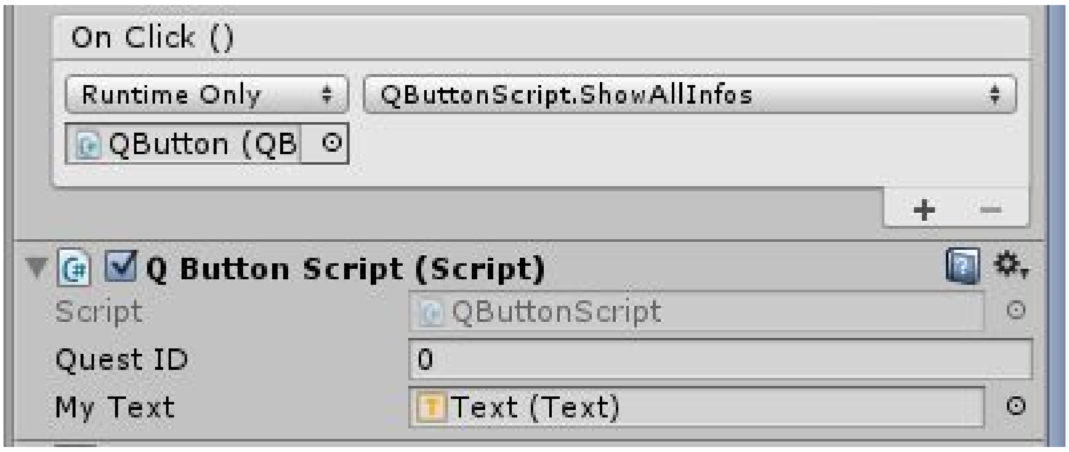
Quest Available: Leave it unchecked!
Quest Running: Leave it unchecked!
Quest Panel: Drag in your Quest Panel.
Quest Log Panel: Drag in your Quest Log Panel.
Available & Active Quests: Nothing to do here!
QButton: The Button Prefab for the Quests.
Q Log Button: The Button Prefab for the Quest Log Quests.**Q Button Spacer:** Usually an empty GameObject or Panel which holds the QButtons for Active Quests.
Q Button Spacer 2: Usually an empty GameObject or Panel which holds the QButtons for Running Quests.

Q Log Button Spacer: Usually an empty GameObject or Panel which holds the QLogButtons for Active Quests seen in the Quest Log.
Quest Title: Text component to show the clicked quest title.
Quest Description: Text component to show the clicked quest description.
Quest Summary: Text component to show the clicked quest summary.
(no direct connection to the quest summary field in the Quest Manager at the moment)
Quest Log Title: Text component to show the clicked quest title in the Quest Log.
Quest Log Description: Text component to show the clicked quest description in the Quest Log.
Quest Log Summary: Text component to show the clicked quest summary in the Quest Log.
(no direct connection to the quest summary field in the Quest Manager at the moment)

3.) Quest Button & Quest Log Button



This Buttons just showing the name of the Quests. So they just require a text component.
So the Quest Button needs the QButton Script. The Text field is the one on the Button itself.
Add an OnClick Event and set the ShowAllInfos Function.



On Click ()

Runtime Only | QButtonScript.ShowAllInfos

QButton (QB

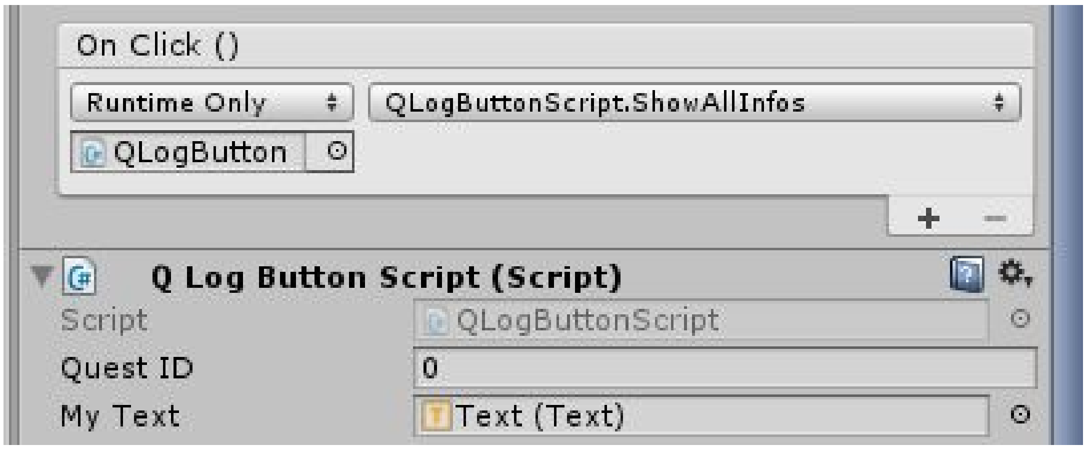
Q Button Script (Script)

Script: QButtonScript

Quest ID: 0

My Text: Text (Text)

Same For the Quest Log Button.



On Click ()

Runtime Only | QLogButtonScript.ShowAllInfos

QLogButton

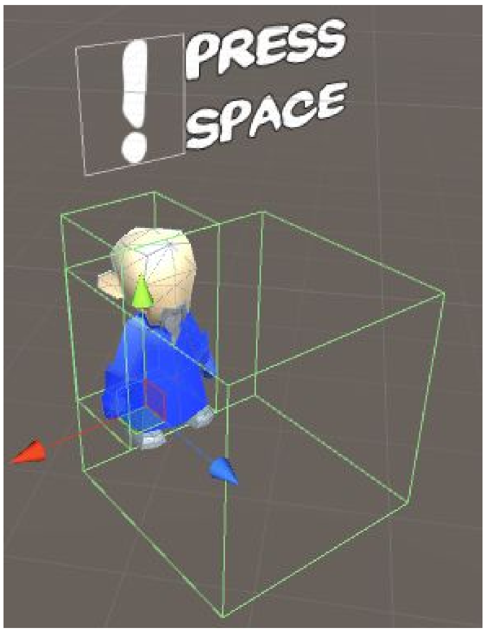
Q Log Button Script (Script)

Script: QLogButtonScript

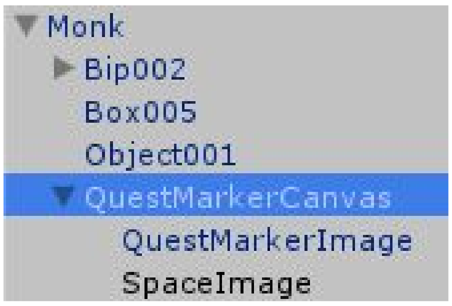
Quest ID: 0

My Text: Text (Text)

4.) The NPC's:

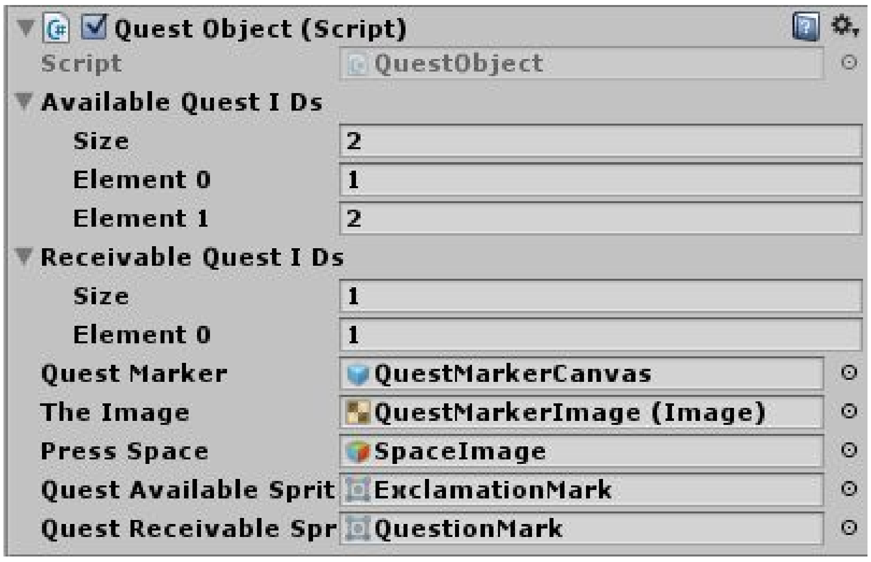


The NPC will need a Canvas for the Exclamation Mark & Question Mark.
This also needs to be child of the NPC.



```
▼ Monk
    ▶ Bip002
      Box005
      Object001
    ▼ QuestMarkerCanvas
        QuestMarkerImage
        SpaceImage
```

The big Collision Box in front of the NPC is of type Trigger, the small around the NPC is a Collider.
We need one of type trigger so the system will trigger the possibility to talk to the NPC.
Now drag the Quest Object script on the NPC.



Quest Object (Script)

Script	QuestObject
Available Quest I Ds	
Size	2
Element 0	1
Element 1	2
Receivable Quest I Ds	
Size	1
Element 0	1
Quest Marker	QuestMarkerCanvas
The Image	QuestMarkerImage (Image)
Press Space	SpaceImage
Quest Available Sprit	ExclamationMark
Quest Receivable Spr	QuestionMark

- Fill in IDs of available quests for this NPC. In my case it will be quest 1 & 2.
- In Receivable is only ID 1, which means he can only take back the Quest with ID 1.
- In QuestMarker place the canvas, in Image drag in the QuestMarkerImage.
- In Available Sprite drag your ExclamationMarker Sprite.
- In Receivable drag in the QuestionMarker Sprite.

5.) Saving & Loading

The questList will be saved in a binary format so it's kinda protected from possible cheaters. You can Drag the SaveLoad Script to any Empty GameObject and call SaveQuest() at any point you want. You can use Buttons and many more. You can also use a Singleton design pattern or make it static to access it from anywhere.

6.) Quest Objectives

If it is a collectible Quest Objective you can easily add the QuestObjective Script. Add the needed values, like string name or ID and that's it.
If it is a "Destroy Monster amount x" just use 1 simple line of code, when the monster dies:

```
QuestManager.questManager.AddQuestItem(monsterName, 1);
```

7.) Contact

Thanks alot for purchasing Simple Quest Manager.
If you have any Question or Problems feel free to contact me by Mail: octoman@arcor.de
You can also visit http://www.octomangames.com/contact and leave a message there.

I will add a video setup guide as soon as possible, to make it easier for you to follow along.

Scripting Reference

There some topics which might interest you codewise.

- **Give Rewards**
  - Depending on your system, you might different approaches to give rewards after completing a quest. Currently there are the references already set, but you need to determine for your character/s how they get rewarded with exp for example.
  - Currently you can already fill in exp, gold and a item String in the quests to reward the player. But there is no function which is handling that.
  - To give your character the correct amount of rewards, you just need to read the completed quest and pass the reward to your character.
  - Example:

In QuestManager.CS look for:

```
public void CompleteQuest(int questID)
  {
      for (int i = 0; i < currentQuestList.Count; i++)
      {
          if (currentQuestList[i].id == questID &&  currentQuestList[i].progress ==
Quest.QuestProgress.COMPLETE)
          {
              currentQuestList[i].progress = Quest.QuestProgress.DONE;
              currentQuestList.Remove(currentQuestList[i]);

              //give rewards later here
//here you need to take the data and pass it over to your character or systems
//Exp Example: yourHero.hero.exp += currentQuestList[i].expReward;
//Gold Example: GameManager.instance.gold += currentQuestList[i].goldReward;
//Inventory Example: Inventory.instance.addItem(currentQuestList[i].itemReward);
//those are all pseudo codes so check how your systems are setup
          }
       }
       //ACTIVATE A CHAIN QUEST - IF THERE IS ANY
       CheckChainQuest(questID);
    }
```

- **Use of Inventory**
    - Since there a lot different Inventory Systems out there, they will all look and work different. However if you want to use Collectibles within an Inventory you need to double check if you have all needed Items in your Inventory when you are trying to complete the quest.
    - Another approach could be when you remove items from the Inventory, check if they are quest Items, and update the Quest System accordingly. To do so you can call (it’s in Questmanager.CS):
    - QuestManager.questManager.RemoveItem(int itemAmount, string itemName)
    - For adding a specific amount back you can call the same function but use a negative Number itemAmount or copy the function, rename it, and add instead of decrease numbers

```
currentQuestList[i].questObjectiveCount += itemAmount;//increase item by passed amount
```

    -

- **Whenever i stumble across more i will place it here**