Document Explicatif

QUÊTES

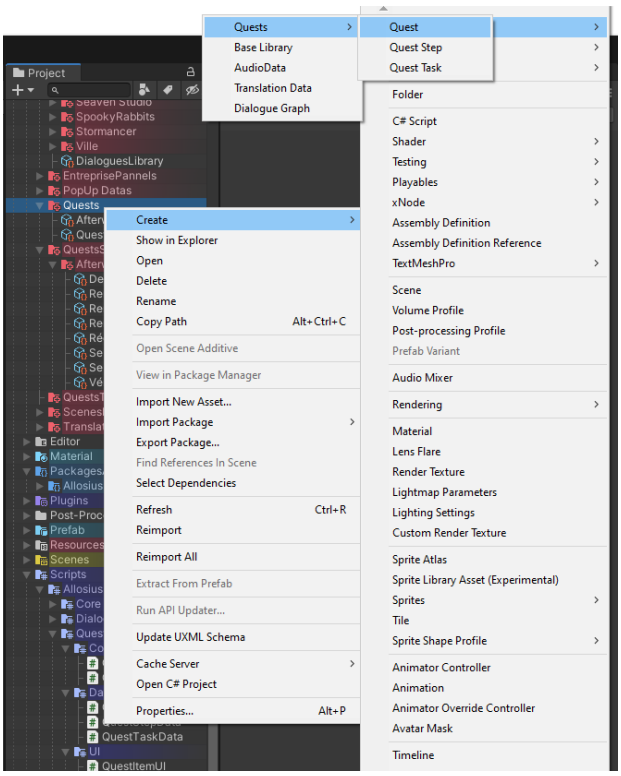
Sommaire :

1) Création et édition des datas de quêtes

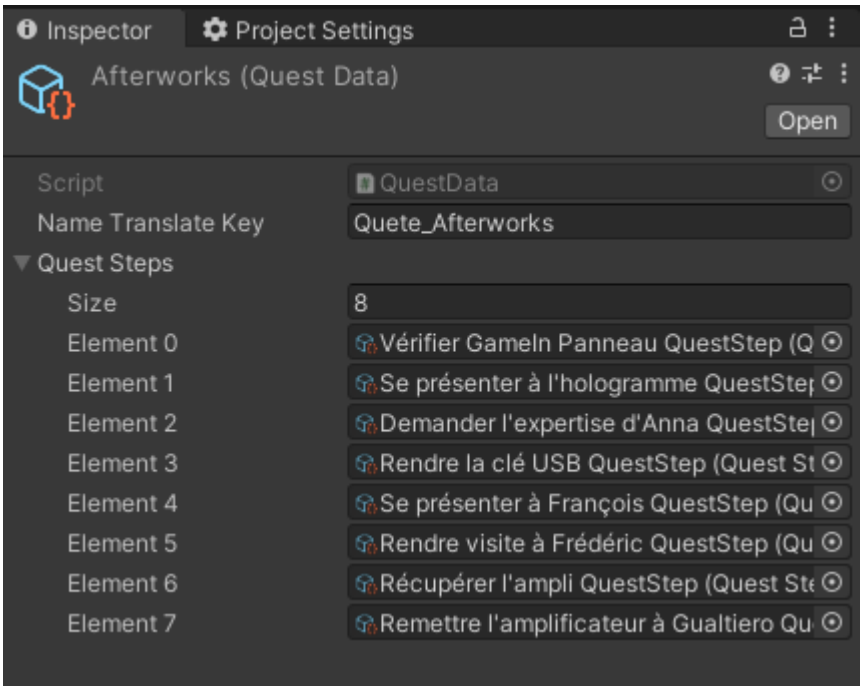
Le système de quêtes fonctionne principalement à l'aide de 2 types de datas principales gérant les quêtes et leurs informations, qui seront ensuite utilisées en jeu :

Quest Data :

On va pouvoir créer un fichier data pour chaque quête de jeu via : **Create > AllosiusDev > Quests > Quest** :



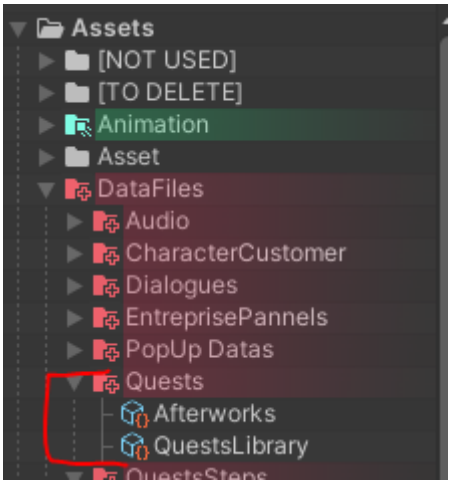
Dans ce fichier, on va définir les différents paramètres requis pour définir la quête :



- **Name Translate Key** : clé de traduction du texte souhaité à récupérer dans le dictionnaire JSON de traduction des quêtes (pour plus de précision voir **document technique système de traduction**), et qui va être utilisé pour afficher le nom de la quête dans le journal de quêtes
- **Quest Steps** : liste de fichier datas d'étapes de quêtes regroupant toutes les étapes de la quête à remplir pour la compléter, par défaut, à l'obtention de la quête, la

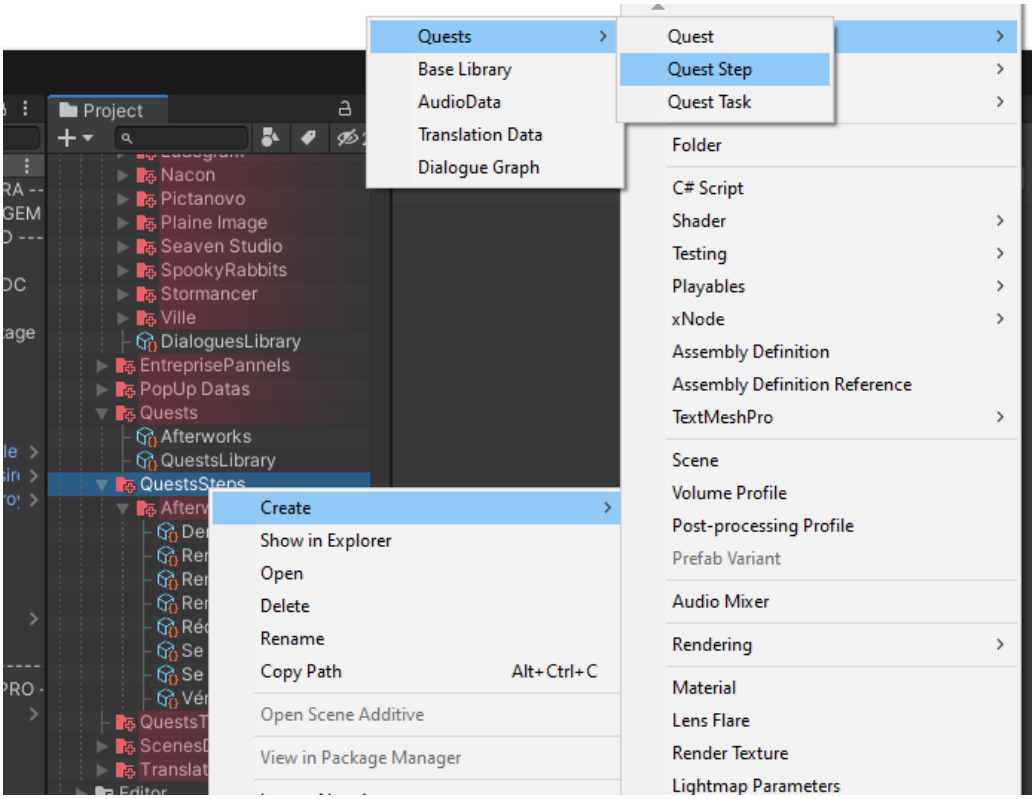
première étape de la liste est débloquée, puis l'étape suivante se débloque automatiquement dès que la précédente est complétée.

Les quests datas sont stockées au chemin : **DataFiles > Quests** :

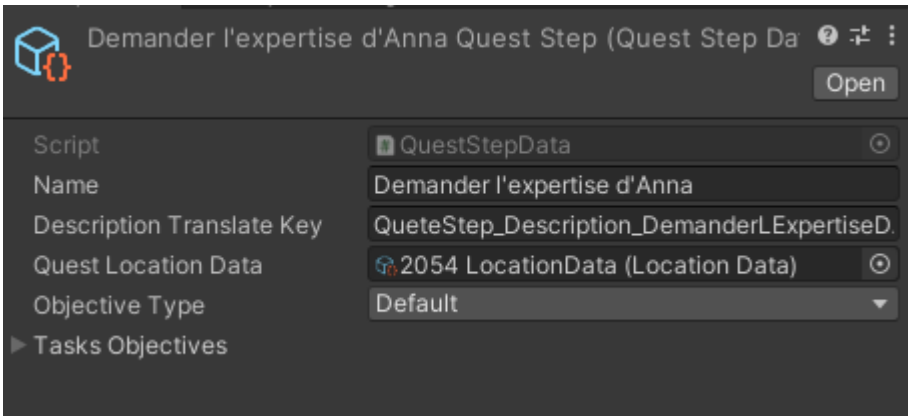


Quest Step Data :

On va pouvoir créer un fichier data pour chaque étape de quête de jeu via : **Create > AllosiusDev > Quests > Quest Step** :

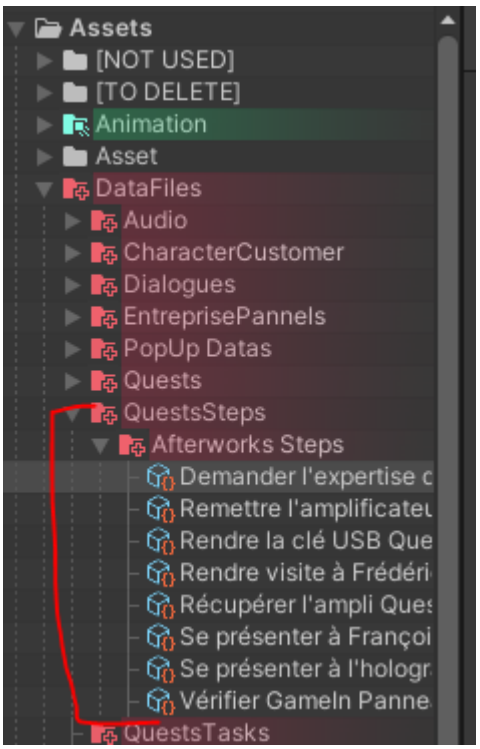


Dans ce fichier, on va définir les différents paramètres requis pour définir l'étape de quête :



- **Name** : nom de la quête (uniquement à but informatif pour s'y retrouver mais aucun impact gameplay)
- **Description Translate Key** : clé de traduction du texte souhaité à récupérer dans le dictionnaire JSON de traduction des quêtes (pour plus de précision voir **document technique système de traduction**), et qui va être utilisé pour afficher la description de l'étape de quête dans le journal de quêtes
- **Quest Location Data** : fichier data du lieu où se déroule la quête (exemple pour la quête de demander l'expertise d'Anna, référencer le **location data** de 2054)
- **Objective Type** : définit le type d'objectifs de l'étape de quête, par exemple si l'étape comporte plusieurs tâches distinctes à accomplir (mais pas implémenté actuellement, donc ne pas en tenir compte)
- **Tasks Objectives** : liste de tâches à accomplir pour compléter l'étape de quête, si objective type est sur With Tasks (pas implémenté actuellement, ne pas en tenir compte)

Les **Quests Steps Datas** sont stockées au chemin : **DataFiles > QuestsSteps** :

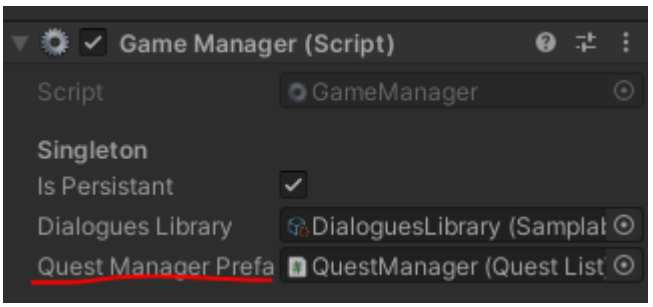


2) Fonctionnement du système de quêtes

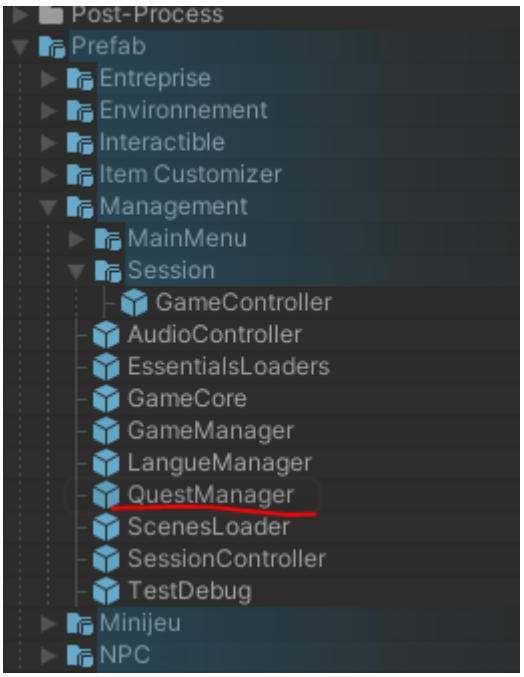
En jeu, pour que les quêtes fonctionnent et puissent être lues correctement, il est nécessaire de s'assurer que certains scripts soient présents et correctement paramétrés.

Quest List :

C'est dans le script **Quest List** que seront stockées les quêtes débloquées par le joueur et que leur état de progression sera géré. Ce script doit donc obligatoirement être présent dans le jeu. Le prefab **Quest Manager** possédant ce script est instancié au lancement par le **Game Manager** :



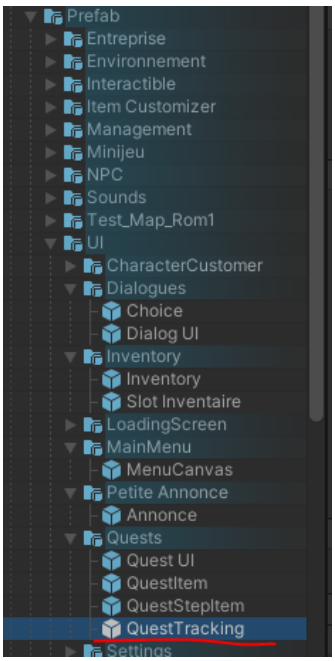
Ce prefab peut être retrouvé au chemin : **Prefab > Management** :

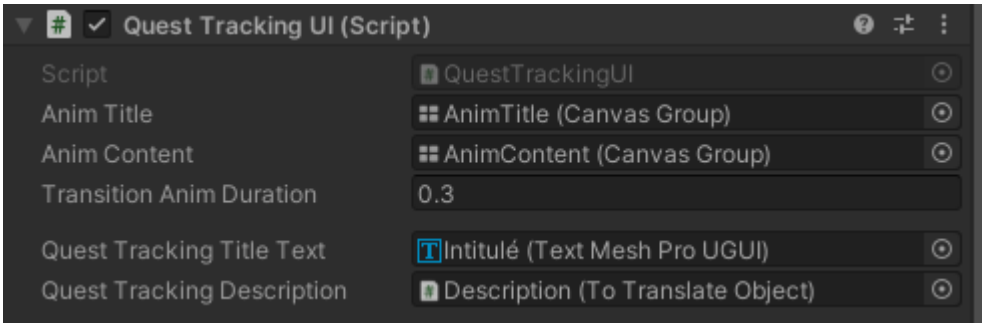


Quest Tracking UI :

Le script **Quest Tracking UI** permet de gérer l'affichage en haut à gauche de l'écran, de la quête active ainsi que son étape actuelle, il est instancié dans le canvas global au lancement du jeu par le **Game Manager** :

Le prefab du **Quest Tracking UI** peut être retrouvé au chemin : **Prefab > UI > Quests** :

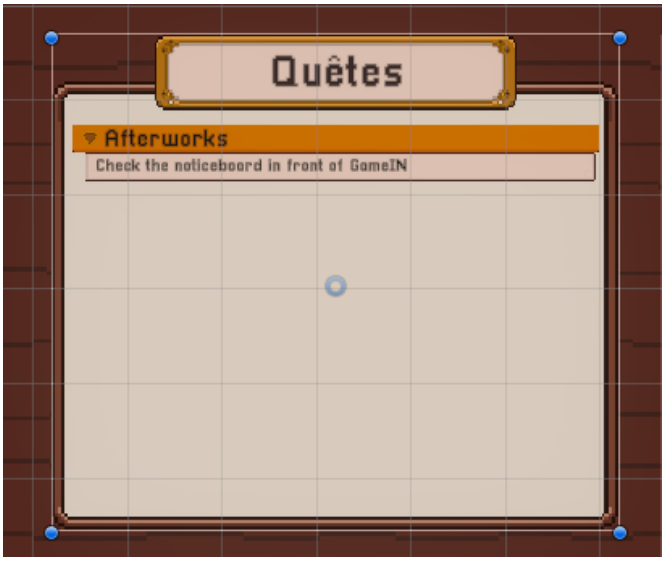




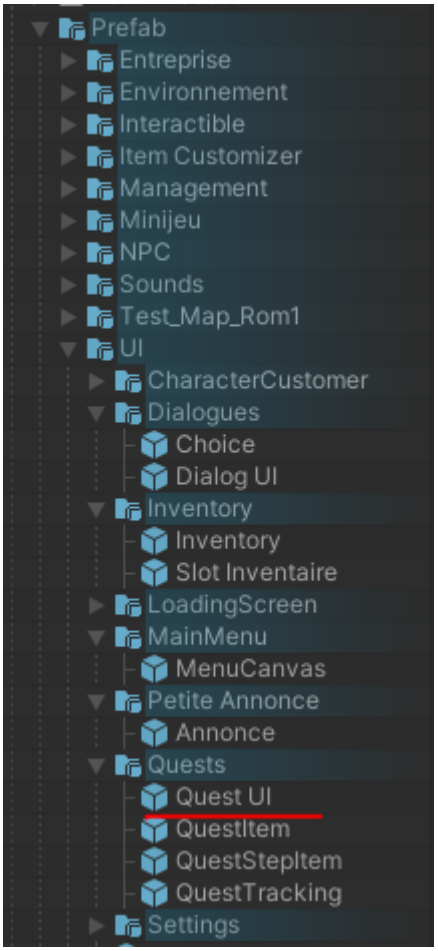
- **Anim Title** : Objet contenant l'intitulé de quête
- **Anim Content** : Objet contenant l'intitulé de l'étape active de la quête
- **Transition Anim Duration** : durée de la transition appliquée lorsqu'une étape de quête est complétée
- **Quest Tracking Title Text** : Texte d'UI affichant le nom traduit de la quête active
- **Quest Tracking Description** : Texte d'UI affichant la description traduite de l'étape de quête en cours

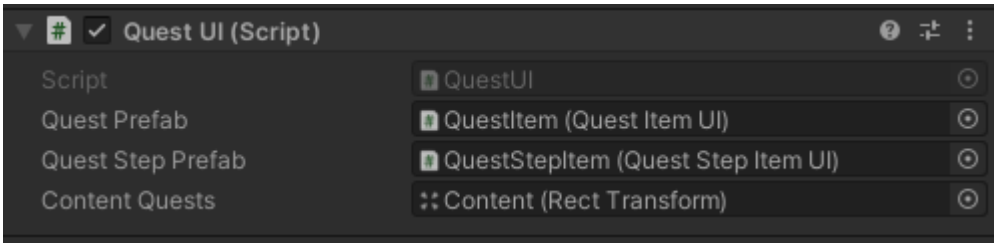
Quest UI :

Le script **Quest UI** permet d'instancier et d'afficher les différentes quêtes ainsi que leurs étapes débloquées dans le journal de quêtes :



Le prefab du **Quest UI** est instancié dans le canvas global également au lancement du jeu, son prefab peut être retrouvé au chemin :





- **Quest Prefab** : **Prefab Quest Item UI** instancié dans le journal de quête pour chaque nouvelle quête (c'est dans cette UI qu'est affichée l'intitulé de la quête)
- **Quest Step Prefab** : **Prefab Quest Step Item UI** instancié dans le journal de quête à la suite de la quête associée (c'est dans cette UI qu'est affichée une étape d'une quête spécifique)
- **Content Quests** : Objet vide parent dans lequel sont créés en enfants les **Quests Prefabs** et les **Quests Steps Prefabs** à l'initialisation du journal de quêtes, en fonction des quêtes stockées dans le **Quest Manager** cité plus haut

Enfin, on a différentes méthodes principales d'exploitation et de gestion des quêtes (voir directement dans le script **Quest List** ainsi que l'ensemble des scripts gérant les quêtes pour davantage d'informations ou toutes les possibilités potentielles du système) :

- **Ajouter une quête** : on peut ajouter une quête en utilisant les **Game Actions** (pour plus de précisions voir **Document technique système de dialogues**), on peut également ajouter une quête manuellement en appelant le script **QuestList** et en utilisant la méthode **AddQuest** prenant en paramètre la data de quête qu'on souhaite ajouter (la première étape de la quête sera automatiquement débloquée en même temps que la quête associée) ex :

```
questList.AddQuest(questToAdd);
```

- **Compléter une étape de la quête** : on peut compléter une étape de quête en utilisant les **Game Actions** (pour plus de précisions voir **Document technique système de dialogues**), on peut également compléter une étape de quête manuellement en appelant le script **QuestList** en utilisant la méthode **CompleteQuestStep** prenant en paramètres la data de la quête associée, ainsi que la data de l'étape de la quête qu'on souhaite compléter)

```
questList.CompleteQuestStep(questAssociated, questStepToComplete);
```

- **Vérifier la présence d'une quête** : on peut vérifier si le joueur possède dans son journal de quête ou non, une quête en utilisant les **Game Requirements** (pour plus de précisions voir **Document technique système de dialogues**), on peut également vérifier une quête manuellement en appelant le script **QuestList** en utilisant la méthode **HasQuest** prenant en paramètres la data de la quête dont on souhaite vérifier la possession, ainsi que son état (qu'importe, non complétée, achevée)

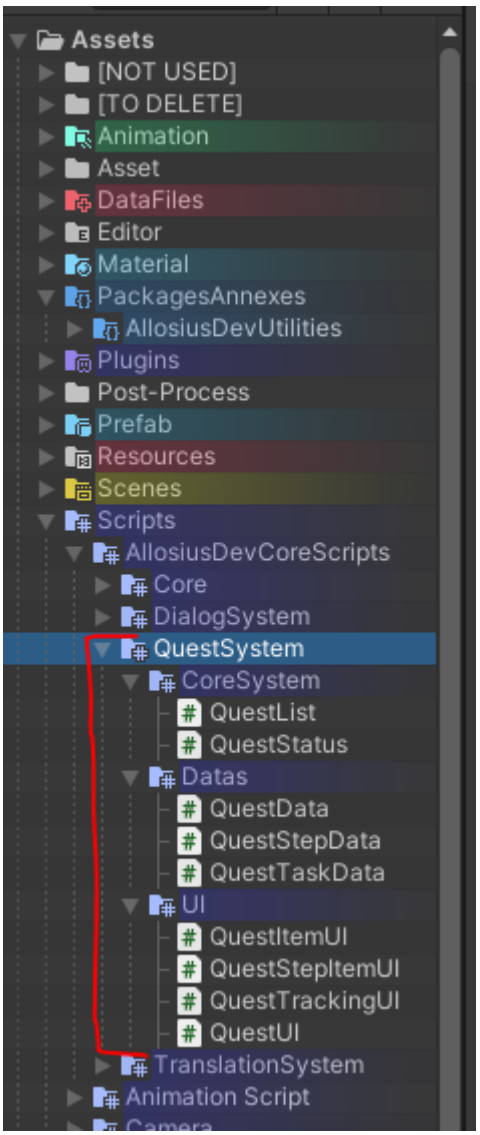
```
bool hasQuest = questList.HasQuest(questToCheck, questToCheckState);
```

- **Vérifier l'état de progression d'une quête** : on peut vérifier si le joueur a débloqué une étape spécifique d'une quête ou non, en utilisant les **Game Requirements** (pour plus de précisions voir **Document technique système de dialogues**), on peut également vérifier l'état de progression d'une quête en appelant le script **QuestList** en utilisant la méthode **HasQuestStep** prenant en paramètres la data de la quête qu'on souhaite vérifier, la data de l'étape de cette quête qu'on souhaite vérifier, ainsi que son état (peu importe, non complétée, achevée)

```
bool hasQuestStep = questList.HasQuestStep(questAssociatedToCheck, questStepToCheck, questStepToCheckState);
```

3) Répertoires des principaux scripts utilisés

Vous pouvez retrouver l'ensemble des scripts utilisés par le système de quêtes(éditeur + scripts de jeu) au chemin : **Scripts > AllosiusDevCoreScripts > QuestSystem:**


Assets
[NOT USED]
[TO DELETE]
Animation
Asset
DataFiles
Editor
Material
PackagesAnnexes
AllosiusDevUtilities
Plugins
Post-Process
Prefab
Resources
Scenes
Scripts
AllosiusDevCoreScripts
Core
DialogSystem
QuestSystem
CoreSystem
QuestList
QuestStatus
Datas
QuestData
QuestStepData
QuestTaskData
UI
QuestItemUI
QuestStepItemUI
QuestTrackingUI
QuestUI
TranslationSystem
Animation Script