

Raptor lab

Rapport de projet

Migration de l'architecture réseau d'une application en VR

Jessy Borcard

15/11/2024

1 TABLE DES MATIERES

2	Analyse Préliminaire	5
2.1	Introduction.....	5
2.2	Structure du document	5
2.3	Organisation	6
2.4	Contexte du Projet	6
2.5	Objectifs.....	6
2.6	Planification Initiale	7
2.6.1	Simplifiée	7
2.6.2	Détailée.....	8
2.6.3	Détails de la planification initiale	9
2.6.4	Déroulement.....	9
2.7	Cadre de travail	10
2.7.1	Environnement de travail.....	10
3	Analyse	13
3.1	Cahier des charges détaillé	13
3.1.1	Besoin de Raptor lab	13
3.1.2	Tâches à réaliser	13
3.1.3	Prérequis et matériel à disposition.....	14
3.1.4	Situation actuelle.....	14
3.1.5	Utilisateurs cible.....	14
3.2	Analyse comparative des solutions envisageables	14
3.2.1	Choix technologique pour le SDK Quad	15
3.2.2	Choix Technologique pour l'export et Import de contenu	21
3.3	Étude de faisabilité	23
3.3.1	Risque technique	23
3.3.2	Risque concernant le planning	24
3.4	Planification finale	24
3.5	Stratégie de test.....	24
3.5.1	Mocking.....	24
3.5.2	Methodologies TDD (Test-Driven Development) :.....	25
4	Conception	26
4.1	Use Case.....	26
4.2	Diagramme de classe	27

4.3	Diagramme de séquence	28
4.3.1	ContentManager	28
4.3.2	DownloadManager	30
4.3.3	UI_ModList	31
4.3.4	Transform Synchronizer	32
4.4	Conventions d'écriture.....	34
4.4.1	Paradigme	34
4.5	Structure du dossier Assets.....	35
5	Implémentation.....	36
5.1	Chargement de scène.....	36
5.1.1	Tests	36
5.1.2	Cas d'erreurs	37
5.2	Chargement de contenu.....	37
5.2.1	Tests	38
5.3	Mocking de Server et Client	38
5.3.1	Problèmes rencontrés.....	39
5.4	Listage de scène et contenu	40
5.5	ModIOManager	41
5.5.1	Initialisation et connexion.....	41
5.5.2	Résolution de problèmes	42
5.5.3	Abonnement.....	43
5.5.4	Recherche	44
5.6	ContentManager.....	44
5.6.1	Chargement de contenu.....	45
5.7	DownloadManager.....	46
5.7.1	Téléchargement.....	46
5.7.2	Problèmes rencontré	47
5.7.3	Résolution des problèmes.....	47
5.8	Transform Synchronizer	47
5.8.1	Problèmes rencontrés.....	49
6	Problèmes connu	50
6.1	ModTOol.....	50
6.1.1	Version	50
6.1.2	Création d'un exporter	50

6.1.3	Importer l'exporter	50
7	Conclusion.....	52
7.1	Objectifs Atteints et Non Attients.....	52
7.2	Points positifs / négatifs	52
7.3	Difficultés particulières	53
7.3.1	Problèmes rencontrés lors de la conception	53
7.3.2	Problèmes rencontrés lors de l'implémentation	53
7.4	Suites possible pour le projet	53
7.5	Planification finale	54
8	Annexes	55
8.1	Sources	55
8.2	Glossaire	55
8.2.1	Qu'est-ce qu'un objet en 3D ?	55
8.2.2	Qu'est-ce qu'un mesh ?	56
8.2.3	Qu'est-ce qu'une scène.....	56
8.2.4	Qu'est-ce qu'une texture ?.....	56
8.2.5	Qu'est-ce qu'un matériau ?	57
8.2.6	Qu'est-ce qu'un shader ?	58
8.2.7	Qu'est-ce qu'une Atlas	58
8.2.8	Qu'est-ce qu'un prefab ?.....	58
8.3	Journal de travail.....	59
8.4	Procédure d'installation	59
8.4.1	Installation d'unity	59
8.4.2	Installation de VsCode	61
8.4.3	Installation de STeam.....	61
8.5	Manuel d'utilisation	61
8.5.1	Sample Scene.....	61
8.5.2	Tests	63
8.5.3	Mise en place d'une scène simple	63
8.5.4	Build.....	64
8.5.5	Connection entre client et serveur	65
8.6	Supports d'archivage du projet	65
8.7	Table des illustrations	65

2 ANALYSE PRÉLIMINAIRE

2.1 INTRODUCTION

Ce rapport documente le travail de diplôme effectué dans le cadre de la formation de Technicien ES en développement d'applications au Centre Professionnel du Nord Vaudois, à Sainte-Croix. Le rapport contiendra les éléments suivants :

- Analyse
- Conception
- Réalisation

Le projet, intitulé « Migration de l'architecture réseau d'une application en VR », s'inscrit dans le cadre des activités de développement de l'entreprise Raptor Lab SàRL, spécialisée dans le domaine des technologies immersives et des solutions de streaming de jeux pour casques de réalité virtuelle (VR). Ce travail de diplôme vise à répondre à un besoin croissant d'optimisation des applications VR pour les casques autonomes.

2.2 STRUCTURE DU DOCUMENT

Ce document présentera le déroulement du travail de diplôme à travers les différentes étapes du projet, ainsi que les réflexions associées aux choix effectués tout au long de celui-ci.

Voici les détails de structure :

- Analyse préliminaire : introduction au projet, structure du document, mise en place du contexte de travail et planification initiale.
- Notions communes : explications de mots clés, notions liées à Raptor lab, Unity, et autres SDK, qui sont fortement lié au domaine du jeu vidéo.
- Analyse : cahier des charges détaillé, étude de faisabilité, planification finale et stratégie de test.
- Conception : détails sur les plans topologiques.
- Réalisation : phase d'implémentation de la solution définie.
- Tests : validation des différents tests pensés lors de l'analyse sur les fonctionnalités implémentées lors de la réalisation.
- Conclusion : bilan final, rétrospective du projet et avis personnel.

2.3 ORGANISATION

Rôle	Nom	Prénom	Courriel	Téléphone
Étudiant	Borcard	Jessy	Jessy.borcard@eduvaud.ch	078 912 32 41
Responsable	Tanoh	Kevin	Kevin.tanoh.k@gmail.com	076 823 80 75
Responsable CPNV	Hurni	Pascal	PasEcal.hurni@eduvaud.ch	024 55 76081
Expert	Sabourin	Thierry	tsabourin@vaudoise.ch	
Expert	Ansermot	David	David.ansermot@gmail.com	078 611 41 17

Période de réalisation : 15 novembre au 30 décembre

2.4 CONTEXTE DU PROJET

Raptor Lab s'efforce de développer un système de streaming avancé permettant aux utilisateurs de casques de réalité virtuelle autonomes de profiter d'une expérience immersive, tout en contournant les limitations imposées par la puissance de calcul de ces appareils. Contrairement aux casques filaires connectés à un PC puissant, les casques autonomes disposent de ressources matérielles limitées, rendant difficile l'exécution de calculs intensifs en temps réel. Face à cette contrainte, Raptor Lab a conçu une solution innovante où la majorité des calculs et traitements est déléguée à un serveur, laissant au casque le seul rôle d'afficher les scènes.

2.5 OBJECTIFS

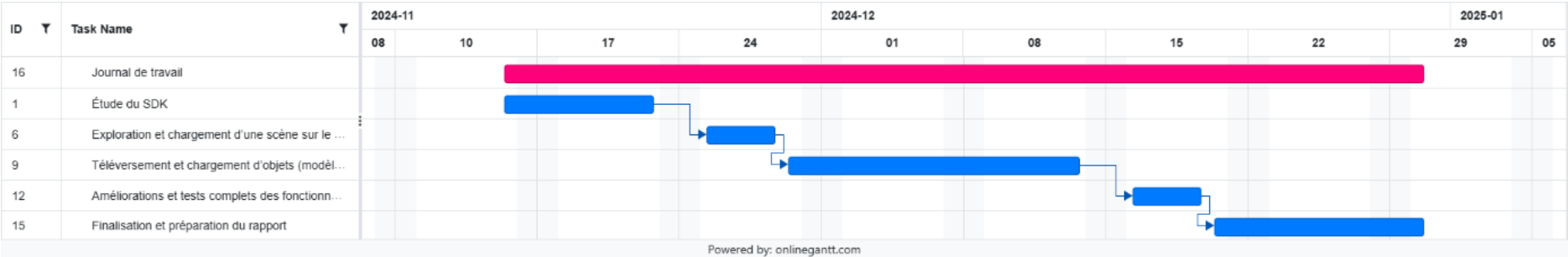
Les objectifs principaux de ce projet sont les suivants :

- Migrer et améliorer l'architecture réseau existante de l'application VR.
- Permettre un téléversement et un chargement efficace des scènes et des objets via la technologie cloud.
- Utiliser et intégrer la plateforme Mod.IO pour le stockage et la gestion des ressources.
- Optimiser l'expérience utilisateur en réduisant la charge de calcul des casques autonomes.
- Assurer la synchronisation en temps réel des objets et des scènes.
- Réaliser des tests approfondis de performance et de robustesse.

2.6 PLANIFICATION INITIALE

2.6.1 SIMPLIFIÉE

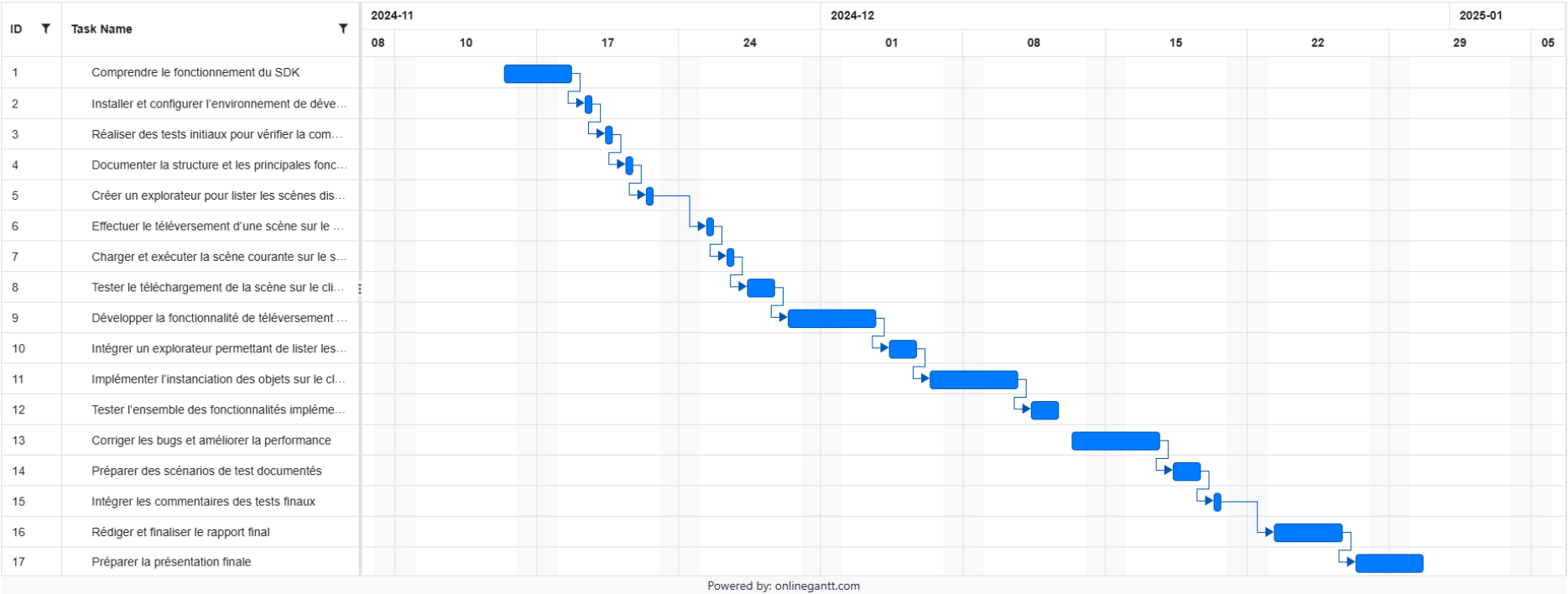
Voici une version simplifiée de la planification qui démontre les objectifs principaux.



2.6.2 DÉTAILÉE

Voici une version détaillée de la planification initiale qui résume toutes les tâches à effectuer et à quel moment.

Vous trouverez en dessus de l'image la liste des tâches complète.



- Comprendre le fonctionnement du SDK
- Installer et configurer l'environnement de développement
- Réaliser des tests initiaux pour vérifier la compréhension des principales fonctions du SDK
- Documenter la structure et les principales fonctions du SDK
- Créer un explorateur pour lister les scènes disponibles sur le cloud (Mod.IO)
- Effectuer le téléversement d'une scène sur le cloud
- Charger et exécuter la scène courante sur le serveur
- Tester le téléchargement de la scène sur le client sans exécution du code
- Développer la fonctionnalité de téléversement d'objets sur le cloud
- Intégrer un explorateur permettant de lister les objets disponibles
- Implémenter l'instanciation des objets sur le client et leur synchronisation pendant l'exécution
- Tester l'ensemble des fonctionnalités implémentées
- Corriger les bugs et améliorer la performance
- Préparer des scénarios de test documentés
- Intégrer les commentaires des tests finaux
- Rédiger et finaliser le rapport final
- Préparer la présentation finale

2.6.3 DÉTAILS DE LA PLANIFICATION INITIALE

La première semaine sera dédiée à l'analyse, incluant la rédaction du cahier des charges, la définition des différents objectifs ainsi que l'étude de leur faisabilité. Cette phase comprendra également la conception de plusieurs diagrammes UML.

De la deuxième à la quatrième semaine, le temps sera consacré à l'implémentation des diverses fonctionnalités.

La cinquième semaine sera réservée aux tests de l'application, à la documentation de ces derniers, ainsi qu'à la correction des éventuels problèmes ou erreurs restants.

Enfin, la dernière semaine sera utilisée pour finaliser le rapport et documenter les modifications apportées par rapport à la conception initiale.

2.6.4 DÉROULEMENT

Le travail de diplôme a débuté le vendredi 18 novembre 2024 et se terminera le lundi 30 décembre 2024, soit un peu plus de six semaines, sans compter les jours de congé officiels et les week-ends.

2.7 CADRE DE TRAVAIL

Raptor Lab est une entreprise suisse à la pointe de la technologie immersive. Fondée sur l'innovation et animée par une passion pour les expériences VR, elle développe des solutions de streaming performantes, spécialement adaptées aux casques autonomes. Son expertise englobe la conception de systèmes de réalité virtuelle avancés, où la recherche et le développement permettent de repousser les limites technologiques pour offrir des expériences fluides et véritablement immersives.

2.7.1 ENVIRONNEMENT DE TRAVAIL

Pour la durée de mon travail, j'ai utilisé mon ordinateur personnel avec :

- OS : Windows 11 23H2
- CPU: Intel Core i9-9900K
- RAM: 32 GO DDR4
- GPU: Nvidia RTX 2080ti

Il est impératif de bien noter la version d'Unity utilisée. Si vous choisissez une autre version, vous risquez de rencontrer de gros problèmes. Lors de l'installation d'Unity, la version appropriée de Visual Studio Community est également installée automatiquement.

Pour les applications pour le développement installées :

- Unity 2022.3.30f1
- Visual Studio Community 2022

Pour les SDKs et librairies installées :

SDK :

- Quad : fournit par Raptor lab
- ModTool : fournit par Raptor lab
- ModIO : fournit par Raptor lab
- Fusion : fournit par Raptor lab

Librairies :

- com.meta.xr.sdk.all: 66.0.0
- com.unity.2d.sprite: 1.0.0
- com.unity.2d.tilemap: 1.0.0
- com.unity.ads: 4.4.2
- com.unity.ai.navigation: 1.1.5
- com.unity.analytics: 3.8.1
- com.unity.collab-proxy: 2.3.1
- com.unity.ide.rider: 3.0.28
- com.unity.ide.visualstudio: 2.0.22

- com.unity.ide.vscode: 1.2.5
- com.unity.nuget.mono-cecil: 1.10.2
- com.unity.nuget.newtonsoft-json: 3.2.1
- com.unity.purchasing: 4.11.0
- com.unity.render-pipelines.core: 14.0.6
- com.unity.render-pipelines.universal: 14.0.6
- com.unity.shadergraph: 14.0.6
- com.unity.test-framework: 1.1.33
- com.unity.textmeshpro: 3.0.6
- com.unity.timeline: 1.7.6
- com.unity.ugui: 1.0.0
- com.unity.visualscripting: 1.9.4
- com.unity.xr.legacyinputhelpers: 2.1.10
- com.unity.xr.oculus: 4.2.0
- com.unity.modules.ai: 1.0.0
- com.unity.modules.androidjni: 1.0.0
- com.unity.modules.animation: 1.0.0
- com.unity.modules.assetbundle: 1.0.0
- com.unity.modules.audio: 1.0.0
- com.unity.modules.cloth: 1.0.0
- com.unity.modules.director: 1.0.0
- com.unity.modules.imageconversion: 1.0.0
- com.unity.modules.imgui: 1.0.0
- com.unity.modules.jsonserialize: 1.0.0
- com.unity.modules.particlesystem: 1.0.0
- com.unity.modules.physics: 1.0.0
- com.unity.modules.physics2d: 1.0.0
- com.unity.modules.screencapture: 1.0.0
- com.unity.modules.terrain: 1.0.0
- com.unity.modules.terrainphysics: 1.0.0
- com.unity.modules.tilemap: 1.0.0
- com.unity.modules.ui: 1.0.0
- com.unity.modules.uitable: 1.0.0
- com.unity.modules.umbra: 1.0.0
- com.unity.modules.unityanalytics: 1.0.0
- com.unity.modules.unitywebrequest: 1.0.0
- com.unity.modules.unitywebrequestassetbundle: 1.0.0
- com.unity.modules.unitywebrequestaudio: 1.0.0
- com.unity.modules.unitywebrequesttexture: 1.0.0
- com.unity.modules.unitywebrequestwww: 1.0.0
- com.unity.modules.vehicles: 1.0.0
- com.unity.modules.video: 1.0.0
- com.unity.modules.vr: 1.0.0

- com.unity.modules.wind: 1.0.0
- com.unity.modules.xr: 1.0.0

Pour la documentation :

- Suite office 365
- Visual Studio Code Avec PlantUML (pour les diagrammes)

Applicatif :

- Steam : <https://store.steampowered.com/?l=french>

Le projet nécessite Steam pour fonctionner, car la connexion se fait via cette application.

3 ANALYSE

3.1 CAHIER DES CHARGES DÉTAILLÉ

3.1.1 BESOIN DE RAPTOR LAB

Le projet répond à un besoin crucial : maximiser la fluidité et la performance des applications VR sur casques autonomes. La solution proposée par Raptor Lab repose sur le développement d'un système permettant d'effectuer la quasi-totalité des calculs côté serveur, ne transmettant au client (le casque) que les informations essentielles, garantissant ainsi un affichage fluide et réactif.

3.1.2 TÂCHES À RÉALISER

3.1.2.1 ÉTUDE DU SDK (1 SEMAINE)

- Comprendre le fonctionnement du SDK
- Test du SDK

3.1.2.2 EXPLORATION ET CHARGEMENT D'UNE SCENE SUR LE SERVEUR ET LES CLIENTS (1 SEMAINE)

- Explorateur pouvant lister les scènes disponibles sur le cloud (Mod.IO)
- Téléversement d'une scène sur le cloud.
- Chargement de la scène courante sur le serveur et exécution du code
- Téléchargement de la scène sur le client sans exécution du code

3.1.2.3 TELEVERSEMENT ET CHARGEMENT D'OBJETS (MODELE 3D + CODE) (1 SEMAINE)

- Téléversement d'objets
- Explorateur pouvant lister les objets
- Instanciation (chargement) d'objets synchronisés pendant l'exécution

3.1.3 PRÉREQUIS ET MATÉRIEL À DISPOSITION

3.1.3.1 PRÉREQUIS :

- Connaissance du C#
- Connaissance du logiciel Unity
- Base de connaissance de la programmation de systèmes connectés par internet

3.1.3.2 MATERIEL A DISPOSITION :

- SDK Quad (Quest as Display) de Raptor lab
- SDK Fusion
- Accès au Fusion Cloud de raptor Lab

3.1.4 SITUATION ACTUELLE

Raptor Lab a développé plusieurs jeux, notamment WarDust, une solution qui était initialement adaptée à une plateforme VR plus répandue à l'époque. Ces jeux utilisent Photon, un SDK proposé par la même entreprise qui a créé Fusion, et WarDust utilise également ModTool.

Cependant, depuis quelque temps, une autre plateforme VR, la Meta Quest, qui se distingue par sa portabilité et son accessibilité pour les particuliers, prend de plus en plus d'ampleur. Cela a entraîné une augmentation du nombre de joueurs utilisant cette plateforme. Le problème, c'est que les jeux de Raptor Lab ne sont pas du tout optimisés pour Meta Quest. Bien que nous puissions lancer nos projets sur cette plateforme, des problèmes de performance majeurs persistent.

3.1.5 UTILISATEURS CIBLE

Tous les détenteurs des jeux Raptor Lab utilisant la plateforme Meta Quest, ainsi que les développeurs et futurs développeurs chez Raptor Lab, sont concernés par cette problématique.

3.2 ANALYSE COMPARATIVE DES SOLUTIONS ENVISAGEABLES

Plusieurs options technologiques s'offraient à nous pour améliorer les performances de nos projets. Je vais vous expliquer notre choix concernant l'infrastructure réseau, puis aborder le choix de la gestion du contenu additionnel et la manière dont nous le packageons.

3.2.1 CHOIX TECHNOLOGIQUE POUR LE SDK QUAD

Quad est un SDK développé par Raptor lab qui se repose sur Fusion

3.2.1.1 CONTRAINTES LIÉES À LA SITUATION ACTUELLE

Une contrainte importante est le temps perdu à développer une technologie dont l'efficacité n'est pas optimale. En effet, passer beaucoup de temps à développer une solution pour des résultats peu satisfaisants n'est pas une approche optimale.

Par exemple, pour WarDust, si nous ne mettons pas en place ce nouveau système, le projet entier devrait être reprogrammé depuis le début. En effet, il est beaucoup plus complexe d'adapter le sujet de mon diplôme que de réécrire complètement WarDust en utilisant ce sujet comme base.

Ainsi, notre choix a été influencé par plusieurs facteurs :

- Le temps de développements
- Les performances du système
- Retours des utilisateurs
- Contrainte de plateforme

Cela dit, je vais également vous présenter d'autres solutions que nous avons envisagées.

3.2.1.2 SYSTÈME HYBRIDE AVEC FUSION ET PHOTON

Cette solution visait à éviter de devoir reprogrammer entièrement WarDust, tout en conservant des performances acceptables. Cependant, il s'est avéré plus judicieux de passer entièrement à Fusion, car Photon est désormais déprécié.

3.2.1.3 PRÉSENTATION ET COMPARAISON DES SOLUTIONS

3.2.1.3.1 RÉDUIRE LES GRAPHISMES

Durant le stage, nous avons testé plusieurs méthodes pour augmenter les performances en réduisant les graphismes de différentes manières. Cependant, cette approche n'a pas été bien reçue par la communauté, ce qui nous a empêchés d'envisager cette solution.

3.2.1.3.2 PRÉSENTATION DU CHOIX

Ainsi, avec **Fusion** comme choix technologique, nous avons dû repartir depuis le début. Mon travail de diplôme se concentre uniquement sur un ajout à ce SDK.

3.2.1.3.3 CHOIX TECHNOLOGIQUE POUR L'HÉBERGEMENT DE CONTENU

Je vous ai précédemment expliqué comment et pourquoi nous avons fait ces choix pour le SDK Quad. Maintenant, je vais vous expliquer comment et pourquoi ces choix impactent mon travail de diplôme.

3.2.1.3.3.1 CONTRAINTES LIÉES À LA SITUATION ACTUELLE

Comme mentionné précédemment, étant donné que la plateforme Meta Quest est désormais plus populaire, nous avons dû envisager la mise en place d'un pipeline de déploiement spécifique pour cette plateforme, ce qui pose un véritable problème. Il faut savoir que Meta, le fabricant du Meta Quest, a des concurrents directs dans le domaine de la VR, notamment Steam, et c'est ce dernier que nous utilisons pour gérer l'ajout de contenu additionnel.

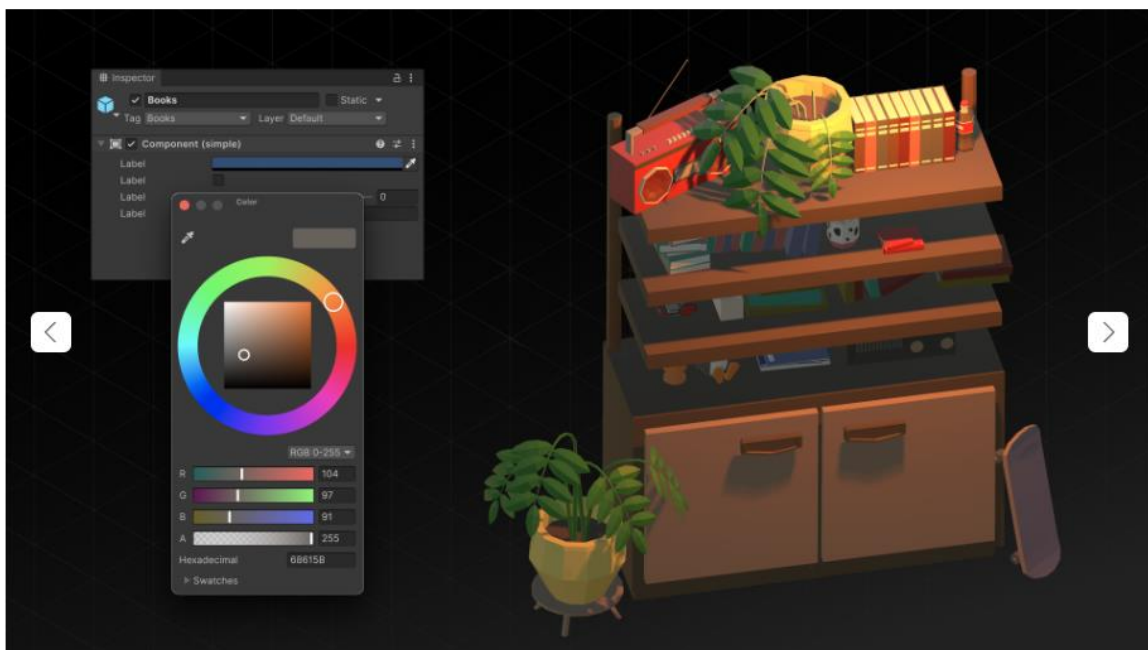
Cependant, le problème est que Meta bloque toute mention de Steam, y compris son système de gestion du contenu additionnel, ce qui nous empêche de déployer nos projets sur Meta Quest. C'est pour cette raison que nous avons cherché une solution alternative à Steam, qui ne soit pas un concurrent direct. C'est ainsi que nous avons opté pour Mod.IO.

Une autre contrainte est le coût de la plateforme. Unity propose un système similaire, mais il devient payant après un certain seuil, contrairement à Mod.IO, qui est gratuit. Ce choix s'est donc imposé de manière évidente, mais je vais tout de même présenter d'autres aspects intéressants pour comparer ces solutions.

3.2.1.4 PRÉSENTATION ET COMPARAISON DES SOLUTIONS

3.2.1.4.1 UNITY

Unity propose une bibliothèque de services dédiée à son propre produit, mais un service en particulier nous est particulièrement utile.



User Generated Content (UGC) est un outil proposé par Unity, qui permet aux développeurs de gérer du contenu créé par les joueurs directement depuis l'environnement Unity. L'avantage principal de ce service réside dans le fait qu'il est conçu spécifiquement pour Unity, ce qui rend son intégration particulièrement simple. De plus, la documentation est bien complète, et, surtout, cet outil ne pose aucun problème de compatibilité avec les exigences de Meta.

Overview

production

Contents (All-time)

11

Downloads (All-time)

8

Subscriptions (All-time)

-

Reports (All-time)

-

Most popular creators








Creator	Contents	Rating (Avg)
Cn4sHVhsrwyWxAoFNdirIj2loyOW	11	0
Rows per page: 5 ▾ 1-1 of 1 < >		

Creators with most reported content

Creator	Contents	Reports
Cn4sHVhsrwyWxAoFNdirIj2loyOW	11	0
Rows per page: 5 ▾ 1-1 of 1 < >		

1 Exemple d'un petit projet avec UGC

Avec UGC, il est possible de visualiser, directement depuis leur plateforme web, les contenus ajoutés au projet par les joueurs. En fonction des rôles assignés à une personne (par exemple : modérateur), il est également possible de supprimer, via cette même plateforme, les contenus problématiques. Cela s'avère très efficace pour la modération.

Q Search			
Visibility ▾ Created ▾ Updated ▾			
Content Name	Version	Flagged	Status
 trulyATest Added on: August 15, 2024	acba5d82-17df-40b9-89f2-96e7091a1a37		Public
 444 Added on: August 9, 2024 Deleted: August 15, 2024	e0aecf6f-c20c-4dfe-98fe-95409c07bace		Public
 e4wr Added on: August 9, 2024	f4d66151-e395-4ac3-b431-3aab6ce3a42a		Public
 e4w Added on: August 9, 2024	d08e0415-96dc-426b-a8ac-f5e00b622ed9		Public
 vrroom Added on: August 9, 2024	361a9a91-ffd4-4869-ba4a-6a299a5b1273		Public
 test Added on: August 9, 2024	cc559807-5c43-486b-907c-89e814b56157		Public
 wqe Added on: August 9, 2024	0411076b-5e06-4ec1-88e5-3bb3ebae4ea3		Public

2 Liste de contenu sur UGC

On a également la possibilité d'obtenir des métriques détaillées, ce qui peut être très utile, tant pour les développeurs que pour les créateurs de contenu.

Cependant, un des principaux inconvénients de ce service est son coût. Si vous possédez une licence personnelle ou étudiante, les services sont gratuits. En revanche, pour Raptor Lab, qui dispose d'une licence professionnelle, des tarifs et des limites sont imposés, ce qui peut représenter un frein.

User Generated Content			
Storage	①	10 GiB / month	\$0.04 per GiB
Download (Egress)	①	100 GiB / month	\$0.005 per GiB

3 Politique des prix de Unity pour UGC

Nous avons droit à 10 GiB de stockage par mois et 100 GiB de téléchargement, ce qui est suffisant pour de petits projets. Cependant, pour Raptor Lab, ces limites sont largement insuffisantes. Nous dépasserons ces quotas à de nombreuses reprises.


Si la limite est dépassée, le coût du service devient basé sur l'utilisation. Cela peut sembler avantageux, mais cela pose un problème important : il est difficile de prédire précisément le stockage et les téléchargements mensuels au-delà des limites. Nous savons que nous allons dépasser ces seuils, mais sans pouvoir estimer de combien, ce qui peut entraîner des coûts très élevés de manière imprévisible.

3.2.1.5 STEAM

Steam à son propre système, appelé Workshop.

Workshop Steam

Documentation Steamworks > Fonctionnalités > Workshop Steam



EN RÉSUMÉ

Le Workshop Steam est un lieu de partage destiné aux fans et aux membres de la communauté qui leur permet de participer à la création de contenu pour votre jeu.

NIVEAU D'INTÉGRATION


Assez avancé, mais cela dépend de l'usage que vous souhaitez faire du Workshop. Vous devrez développer un moyen de proposer du contenu dans votre jeu ou créer un outil externe. Le contenu devra soit être intégré manuellement à votre jeu, ou ce dernier devra charger les mods téléchargés.

Présentation

Le Workshop Steam a été pensé comme un lieu de partage destiné aux fans et aux membres de la communauté pour leur permettre de participer à la création de contenu pour votre jeu. La forme prise par ces créations des membres de la communauté peut varier en fonction de la nature du jeu et du degré de contrôle que vous désirez avoir sur le contenu de votre jeu.

Cette page a pour but de présenter le Workshop Steam et les différents modèles disponibles. Pour les détails techniques relatifs à l'implémentation du Workshop Steam pour votre titre, veuillez consulter l'article [Guide d'implémentation de Workshop Steam](#).

Vidéo : [introduction au contenu généré par la communauté et au Workshop Steam](#).



Vidéo YouTube™: [Embracing User Generated Content \(Steam Dev Days 2014\)](#)

Vues : 24,377

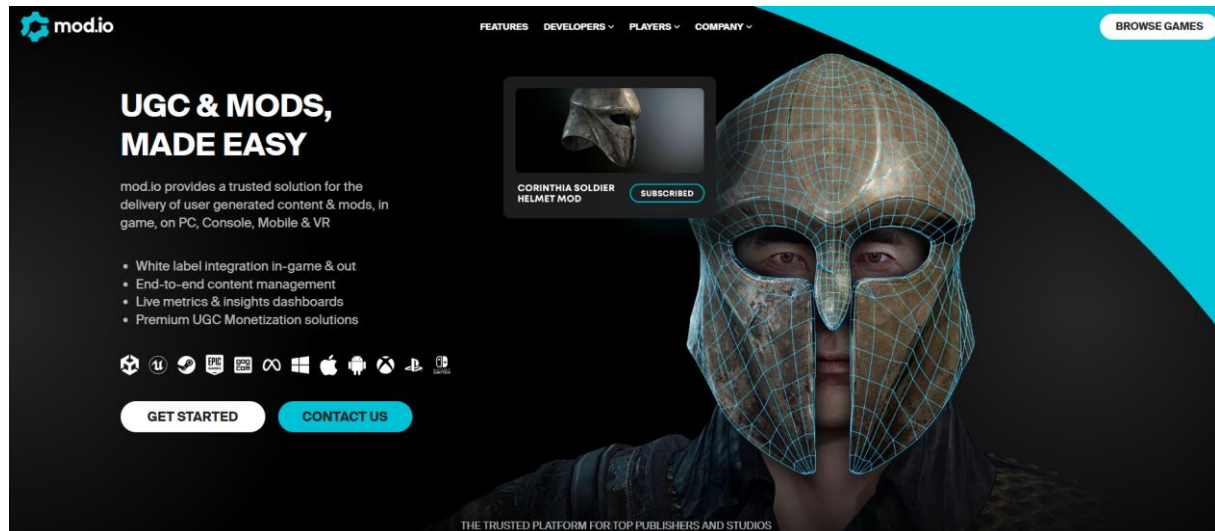
Building your game with user generated content in mind can add significant ongoing value to your product and create a deeper engagement with customers.

4 Présentation de Workshop sur leur documentation

Malheureusement, étant incompatible avec Meta, il était hors de question d'envisager Steam.

3.2.1.6 MODIO

ModIO propose un service similaire à ceux de Steam et Unity, permettant de gérer le contenu ajouté par les utilisateurs.



5 Page d'accueil de ModIO

Le principal avantage de ModIO réside dans son coût : il est entièrement gratuit, sans limites apparentes, ce qui représente un atout majeur pour notre projet. Comme Unity, ModIO propose une plateforme Web permettant de visualiser les contenus, avec l'avantage supplémentaire de pouvoir ajouter facilement du contenu directement depuis cette interface.

Cependant, un problème majeur réside dans la documentation : elle est très peu complète et offre peu de réponses aux questions techniques, ce qui complique l'implémentation des fonctionnalités. Cela a engendré plusieurs blocages lors de mon stage, où j'ai rencontré des difficultés pour mettre en place certaines fonctions.

Un autre avantage notable de ModIO est son indépendance vis-à-vis de plateformes comme Steam ou Unity, réduisant ainsi les risques de voir ce service interdit sur une plateforme comme Meta Quest.

Malgré les défis rencontrés, notamment liés à la documentation, nous avons finalement opté pour ModIO.

3.2.1.7 TEMPS DE DÉVELOPPEMENT

Toutes ces solutions présentent des degrés de complexité variés en termes d'implémentation. Certaines sont spécifiquement conçues pour une plateforme donnée, offrant une intégration simple et efficace, tandis que d'autres s'avèrent plus difficiles à mettre en œuvre.

ModIO, bien qu'avantageux sur certains points, souffre d'un manque de documentation détaillée et d'une communauté majoritairement composée de professionnels. Cela rend les réponses et les ressources disponibles sur le Web rares. En conséquence, l'intégration de nouvelles fonctionnalités demande plus de temps et d'efforts.

Mais heureusement, le temps perdu par rapport à d'autres plateformes est négligeable, de plus, si vous avez face à un problème sur cette plateforme, le prochain problème similaire, prendras beaucoup moins de temps à résoudre

3.2.1.8 JUSTIFICATION DE LA SOLUTION RETENUE

Steam étant inenvisageable, le choix se réduisait principalement à ModIO et Unity. Cependant, en raison des contraintes financières significatives imposées par Unity, ModIO s'est rapidement imposé comme le choix évident.

De plus, ModIO présente un autre avantage crucial : son indépendance vis-à-vis d'Unity. Ainsi, en cas de problème avec Unity Cloud, notre projet pourra continuer à fonctionner grâce à ModIO. Cette garantie de résilience n'est pas assurée avec les services proposés par Unity.

3.2.2 CHOIX TECHNOLOGIQUE POUR L'EXPORT ET IMPORT DE CONTENU

Dans le cadre de ce projet, il est nécessaire d'exporter et d'importer du contenu via ModIO. Je vais donc vous détailler les contraintes rencontrées ainsi que les choix technologiques effectués pour gérer ces opérations efficacement.

3.2.2.1 CONTRAINTES LIÉES À LA SITUATION ACTUELLE

Il est nécessaire de disposer d'une solution indépendante de ModIO, Unity Cloud, ou Workshop. L'objectif est de permettre un changement de stockage de contenu sans affecter la solution choisie pour l'application. Nous avons besoin d'une solution capable de charger du contenu depuis une source externe pendant l'exécution de l'application. Cela implique de gérer des bibliothèques compilées, rendant impossible l'idée d'exporter une simple classe C# et de l'importer à la volée en espérant que cela fonctionne. Une telle solution nécessite une gestion plus complexe et structurée des ressources et des dépendances.

3.2.2.2 PRÉSENTATION ET COMPARAISON DES SOLUTIONS

3.2.2.2.1 MODTOOL

ModTool est une solution permettant l'exportation et l'importation de contenu depuis des sources externes comme ModIO, ou simplement localement sur une machine. L'un de ses avantages est qu'il s'agit d'un SDK fonctionnel et relativement simple d'utilisation, tant pour les développeurs que pour les joueurs. Cependant, son principal inconvénient est son ancienneté : il n'a pas été mis à jour depuis deux ans. Cette situation est problématique, car la dernière version de ce SDK n'est pas compatible avec Unity 2022.3.30f1, la version utilisée dans le projet. Pour contourner ce problème, Raptor Lab m'a fourni une version spécifique. De plus, tout comme

ModIO, la documentation de ModTool est assez basique, ce qui a posé plusieurs difficultés lors de la mise en place des fonctionnalités pendant mon stage.

3.2.2.2.2 SOLUTION RAPTOR LAB

Durant mon stage, j'avais développé une solution qui aurait pu être utilisée pour mon travail de diplôme. Cependant, cette solution n'étant pas terminée, elle n'a pas pu être envisagée pour le projet final. Si elle avait été achevée, elle aurait permis d'accomplir les mêmes tâches que ModTool, mais d'une manière bien plus optimisée et adaptée spécifiquement pour la réalité virtuelle. Cette solution aurait offert des performances accrues et une intégration plus fluide, ce qui était crucial pour le contexte du projet.

3.2.2.3 JUSTIFICATION DE LA SOLUTION RETENUE

Le choix était relativement simple, car la solution développée en interne par Raptor Lab n'était pas encore prête à ce moment-là. Il était donc impératif de trouver une solution déjà existante et fonctionnelle. Étant donné que ModTool avait déjà été utilisé par Raptor Lab dans des projets précédents, il a naturellement été privilégié. ModTool offrait une solution fiable pour l'export et l'import de contenu, ce qui a facilité sa sélection pour ce projet, malgré ses limitations et sa documentation insuffisante.

3.3 ÉTUDE DE FAISABILITÉ

3.3.1 RISQUE TECHNIQUE

3.3.1.1 MANQUE DE DOCUMENTATION

Comme mentionné précédemment, tant ModIO que ModTool souffrent d'une documentation insuffisante, ce qui peut compliquer le développement de certaines fonctionnalités. Il est donc possible que je me retrouve bloqué sur une fonctionnalité particulière, simplement parce que la documentation ne fournit pas de détails suffisants sur le fonctionnement exact de certaines fonctions. Ce manque de clarté nécessite parfois de passer plus de temps à tester et à expérimenter pour comprendre comment implémenter correctement une fonctionnalité, ce qui peut rallonger le processus de développement.

3.3.1.2 SDK QUAD

Étant donné que cette technologie est nouvelle pour moi, il est possible que je rencontre certains problèmes lors de l'implémentation de nouvelles fonctionnalités. Cependant, étant développé par Raptor Lab, j'ai la chance de pouvoir solliciter leur aide directement pour résoudre les éventuels obstacles, ce qui facilite grandement le processus de développement.

3.3.1.3 ASYNCHRONISATION

Le Quad SDK, ModIO et ModTool reposent largement sur l'asynchrones. Bien que cela ne constitue pas un problème en soi, il est important que je reste vigilant afin de ne pas introduire d'erreurs liées à une mauvaise gestion des appels asynchrones.

.

3.3.2 RISQUE CONCERNANT LE PLANNING

Un des risques majeurs réside dans le manque de documentation, ce qui peut me bloquer lors de l'implémentation de certaines fonctionnalités. Bien que Raptor Lab puisse m'apporter une aide précieuse sur certains problèmes, ils ne maîtrisent pas entièrement ModIO. Il est donc possible que je me retrouve dans une situation où Raptor Lab ne puisse pas répondre à ma question et où la documentation de ModIO ne fournisse aucune solution, et où les recherches sur Internet ne donnent aucun résultat concluant. Dans ce cas, il me faudra prendre davantage de temps pour résoudre le problème par moi-même.

3.4 PLANIFICATION FINALE

TODO

3.5 STRATÉGIE DE TEST

3.5.1 MOCKING

Pour le processus de test, nous utiliserons le mocking afin d'isoler les différentes parties de notre application et d'assurer que les modifications du code n'affectent pas le comportement existant. Nous allons créer quatre mocks distincts pour simuler les interactions avec des composants externes, tout en nous concentrant sur la logique interne de notre code. Ces mocks incluront un pour le contenu additionnel, un pour le serveur, un pour l'utilisateur et un pour la scène.

1. Mock du contenu additionnel : Ce mock servira à simuler les données ou fonctionnalités supplémentaires ajoutées par les mods dans le jeu. Il sera utilisé pour tester que l'intégration et la gestion des mods ne perturbent pas la logique principale du jeu, en s'assurant que les nouvelles ressources sont correctement traitées sans introduire de bugs dans le système de base.
2. Mock du serveur : Ce mock permettra de simuler les interactions avec un serveur multijoueur ou toute autre infrastructure de réseau. Cela inclut des tests sur la gestion des connexions, la synchronisation des objets et la communication entre les clients et le serveur. Il garantira que la logique liée au réseau fonctionne indépendamment de la configuration réelle du serveur.
3. Mock de l'utilisateur : Un mock pour l'utilisateur sera créé afin de simuler les actions des joueurs dans le jeu. Cela inclut les actions comme l'authentification, les entrées utilisateur (clics, mouvements, interactions), et les réactions du système en fonction de ces entrées. Cela nous permettra de tester la réponse du système à des actions d'utilisateur sans nécessiter un véritable utilisateur.
4. Mock de la scène : Ce mock simule l'environnement du jeu, notamment la gestion des scènes, les objets présents dans celles-ci. En utilisant ce mock, nous pourrions tester l'intégration de nouveaux contenus, comme les objets de mod, dans différentes scènes sans devoir charger ou manipuler de véritables scènes.

3.5.2 METHODOLOGIES TDD (TEST-DRIVEN DEVELOPMENT) :

Chaque mock sera développé en suivant la méthodologie de Test-Driven Development (TDD). Cela signifie que les tests pour chaque fonctionnalité seront écrits avant l'implémentation de la logique elle-même. En écrivant les tests d'abord, nous garantissons que chaque partie du code est vérifiable et conforme aux exigences, tout en nous permettant d'ajuster facilement l'implémentation sans risque de casser des fonctionnalités existantes. Cette approche permet également de documenter les attentes de chaque composant par des tests clairs, ce qui facilite la maintenance et l'extension du projet.

Objectifs du mocking dans ce contexte :

- Isoler le code : En utilisant des mocks, nous pourrions tester les fonctionnalités de manière isolée sans dépendre de services externes, d'un serveur en ligne ou de données réelles.
- Assurer la compatibilité avec différents mods : Cette stratégie de tests garantit que l'intégration de contenu modifié, provenant de divers moddeurs ou de versions différentes du projet, ne cassera pas les fonctionnalités de base du jeu. Les tests permettent de s'assurer que les changements dans le code n'affectent pas négativement les utilisateurs utilisant des versions de mods précédentes.
- Faciliter les changements de code : Le mocking assure que les tests sont exécutés de manière rapide et efficace, ce qui permet aux développeurs de faire des changements de code en toute confiance, tout en validant leur impact sur le système global.

Ces tests seront effectués régulièrement pendant les cycles de développement pour garantir la stabilité du code à chaque itération. L'utilisation des mocks permettra de créer un environnement de test robuste et prévisible, essentiel pour un développement agile et pour la gestion des risques dans des projets de grande envergure, comme le nôtre.

4 CONCEPTION

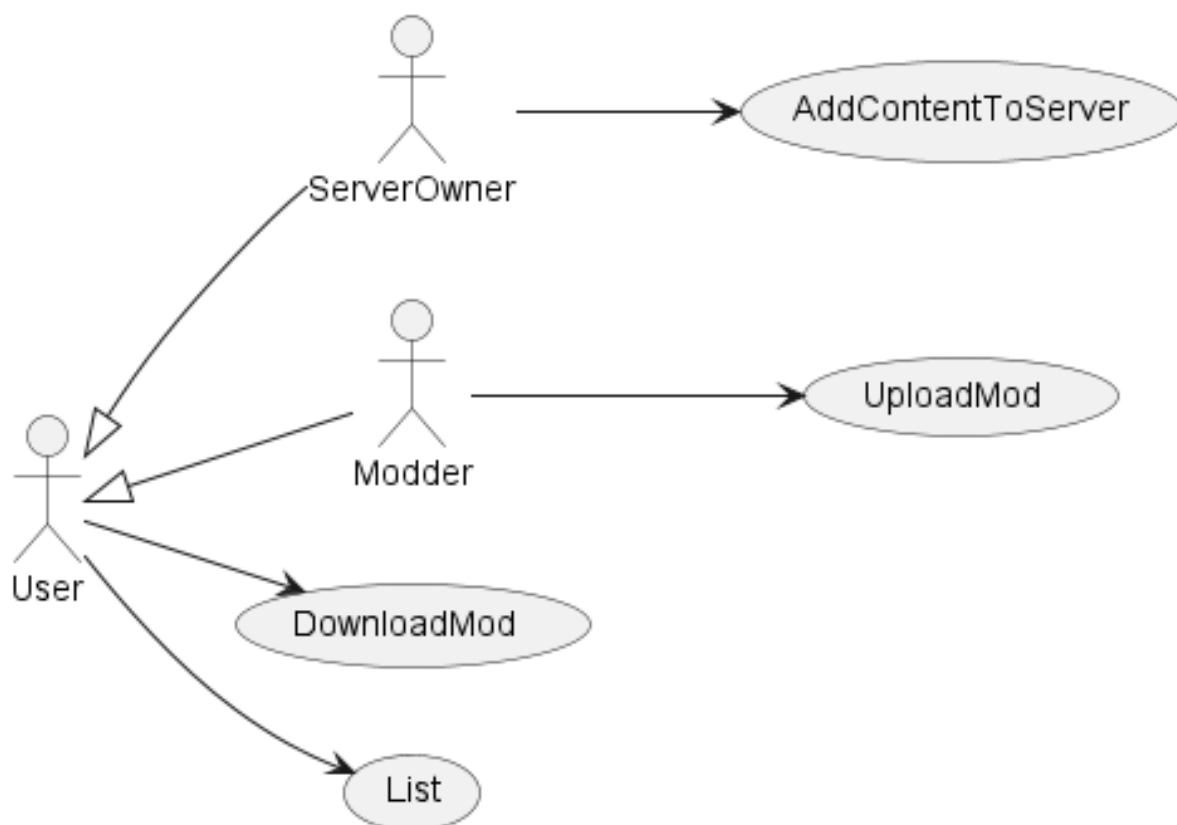
4.1 USE CASE

Dans ce projet, les rôles et actions sont limités et clairement définis. Les personnes ayant les droits d'administrateur sur le serveur, appelées ServerOwners, ont la possibilité d'ajouter du contenu sur le serveur. Cela oblige tous les utilisateurs connectés au même serveur à télécharger ce contenu ou les contenus additionnels. Les ServerOwners héritent des droits des Users et peuvent donc également télécharger du contenu et afficher une liste de contenus disponibles.

Il est important de faire la distinction entre deux actions clés :

- Download : Cela correspond au téléchargement local du contenu additionnel sur un utilisateur.
- AddContentToServer : Cela correspond à l'ajout de contenu sur le serveur, accessible par tous les utilisateurs du serveur.

Les Modders, quant à eux, ont la possibilité de télécharger et de télécharger des contenus additionnels sur ModIO. Comme les ServerOwners, ils héritent des droits des Users.



4.2 DIAGRAMME DE CLASSE

ModIOManager, DownloadManager, ContentManager, UI_ModItem et UI_ModList sont les classes qui ont été créées pour le projet.

ModScene, ModManager et Mod sont des classes appartenant à ModTool, et TransformSynchronizer fait partie du Quad SDK.

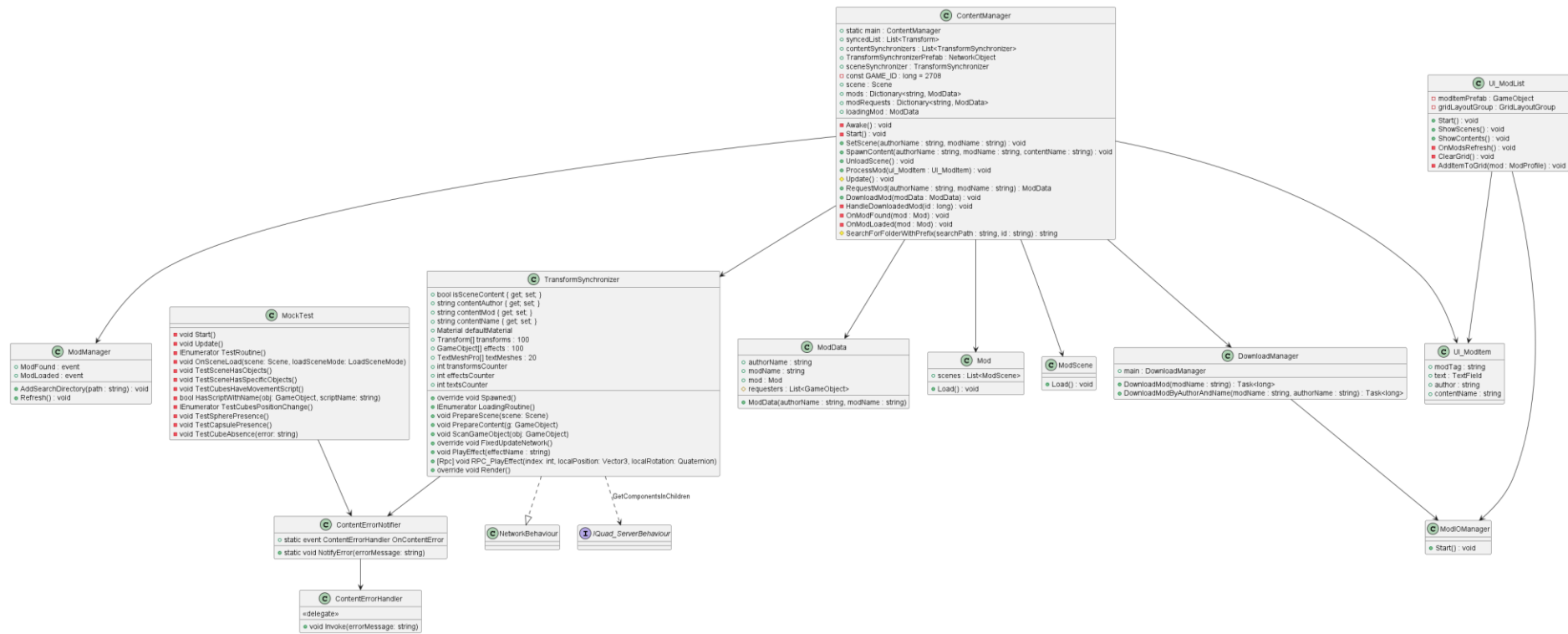
UI_ModList et UI_ModItem sont présentes pour l'affichage visuel du contenu additionnel aux utilisateurs, afin d'éviter de tout mettre dans ContentManager, et de respecter la séparation des responsabilités (SoC).

DownloadManager gère le téléchargement du contenu, synchronise une liste de contenu à travers le réseau et ordonne, selon les cas, à l'utilisateur de télécharger du contenu.

Cependant, ContentManager se contente d'utiliser ce que DownloadManager télécharge, et réagit de manière différente selon les cas.

Il y a une volonté que ContentManager et UI_ModList accèdent toutes deux à ModIOManager. ModIOManager est déjà une couche d'abstraction de ModIO, et comme UI_ModList se contente de faire du listage simple, je me suis permis de l'utiliser directement. En revanche, pour ContentManager, une autre couche d'abstraction est nécessaire, c'est pourquoi ContentManager utilise DownloadManager, qui à son tour utilise ModIOManager. À noter que ContentManager n'utilise jamais directement ModIOManager.

TransformSynchronizer s'occupe de synchroniser les transformations des objets présents sur le réseau, cela inclut les effets.



4.3 DIAGRAMME DE SÉQUENCE

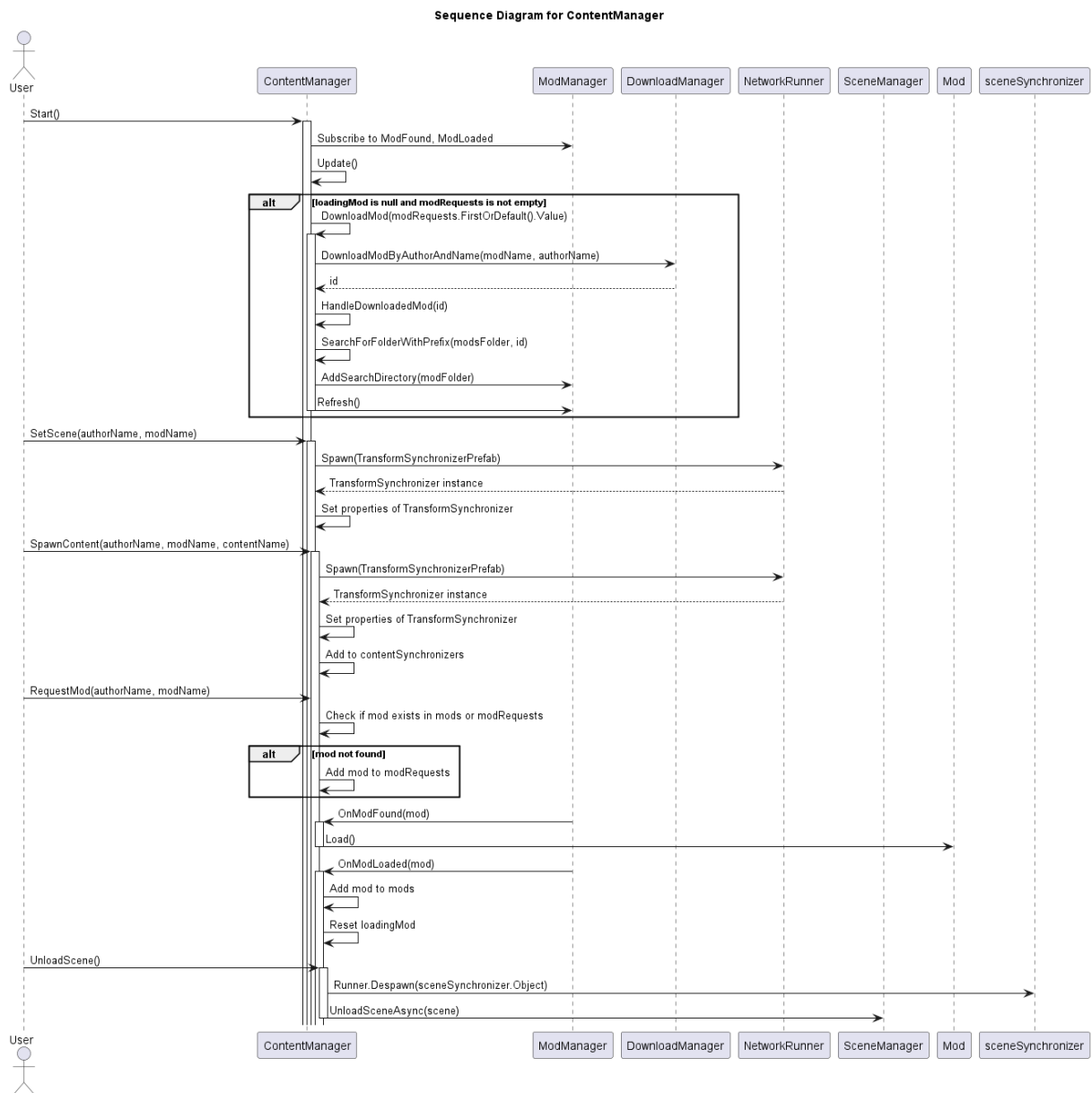
4.3.1 CONTENTMANAGER

Ce script gère l'intégration des mods dans Unity, en utilisant ModTool pour télécharger et ajouter les mods au jeu. Il commence par s'authentifier avec mod.io et écoute les événements des mods, comme lorsqu'un mod est trouvé ou chargé. Lorsqu'un mod est demandé, il vérifie s'il est déjà chargé, et s'il ne l'est pas, il le télécharge.

Pour la synchronisation multijoueur, le script gère la synchronisation des GameObjects entre le serveur et les clients. Il recherche les objets dont le nom se termine par "_S" et les ajoute à une liste. Lorsque le serveur est actif, ces objets sont synchronisés en créant un TransformSynchronizer pour maintenir leur position cohérente chez tous les joueurs.

Le script gère également le téléchargement des mods de manière asynchrone. Une fois qu'un mod est téléchargé, il recherche le dossier du mod à l'aide de l'ID du mod, l'ajoute au ModManager et le rafraîchit pour le charger dans le jeu. Une fois le mod chargé, le script s'assure que tout le contenu ou les scènes associées sont correctement intégrés dans le jeu.

Cependant, à aucun moment le contenu des mods n'est téléchargé à ce niveau ; cette partie est gérée par un autre code.



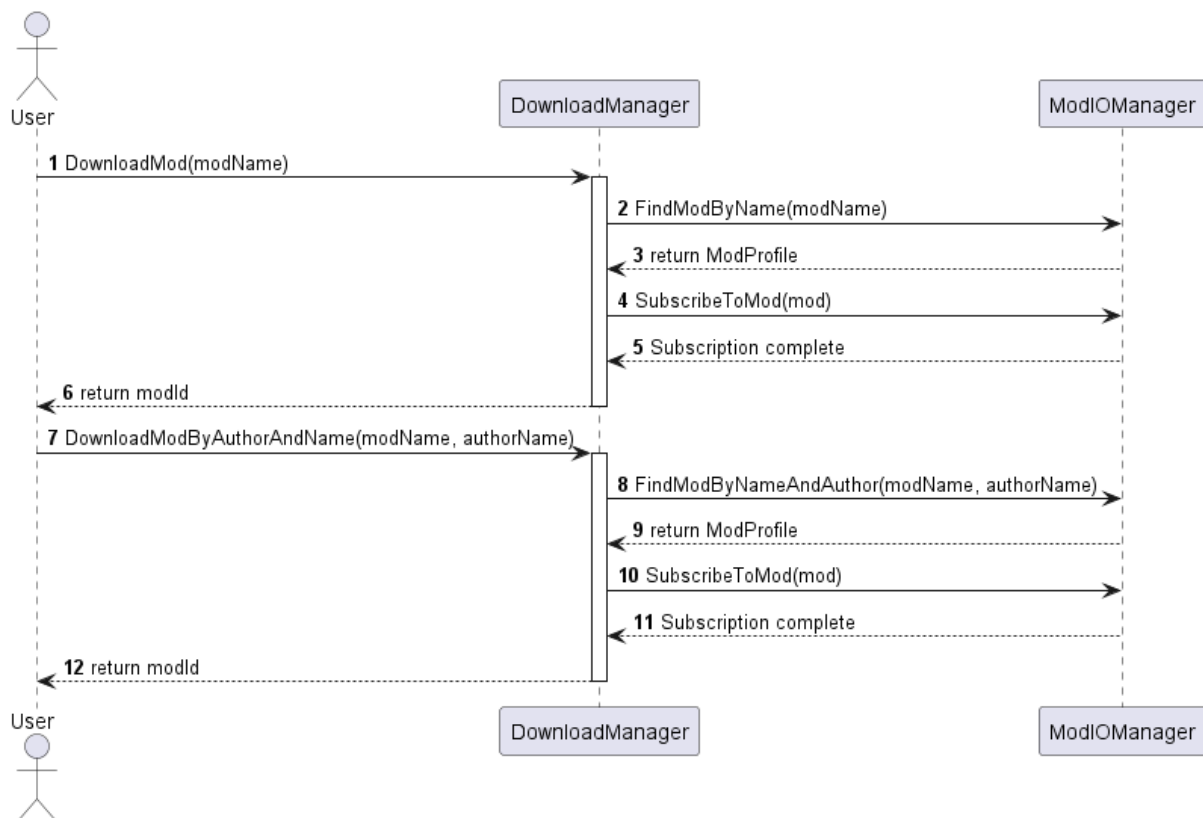
4.3.2 DOWNLOADMANAGER

DownloadManager lie ContentManager à ModIOManager.

ModIOManager ne télécharge jamais deux fois le même contenu. Si vous essayez de télécharger le contenu plusieurs fois, il ne fait rien, car il détecte que le contenu est déjà présent localement et ne télécharge pas à nouveau.

Lorsque DownloadManager est appelé pour télécharger du contenu, il commence par chercher la référence dans la base de données ModIO et s'abonne au contenu.

ModIOManager trouvera de toute manière la référence, car la référence est donnée par UI_ModItem, et les références sont fournies par UI_ModList, qui elles-mêmes viennent de ModIO. Il n'y a donc aucun cas où ModIO ne parviendrait pas à trouver le contenu. De plus, les moddeurs n'ayant pas le droit de supprimer le contenu, il est impossible d'avoir une liste de contenu où un élément aurait été supprimé entre-temps. La référence sera donc toujours trouvée, et il n'y a aucune raison de faire des tests de nullité.

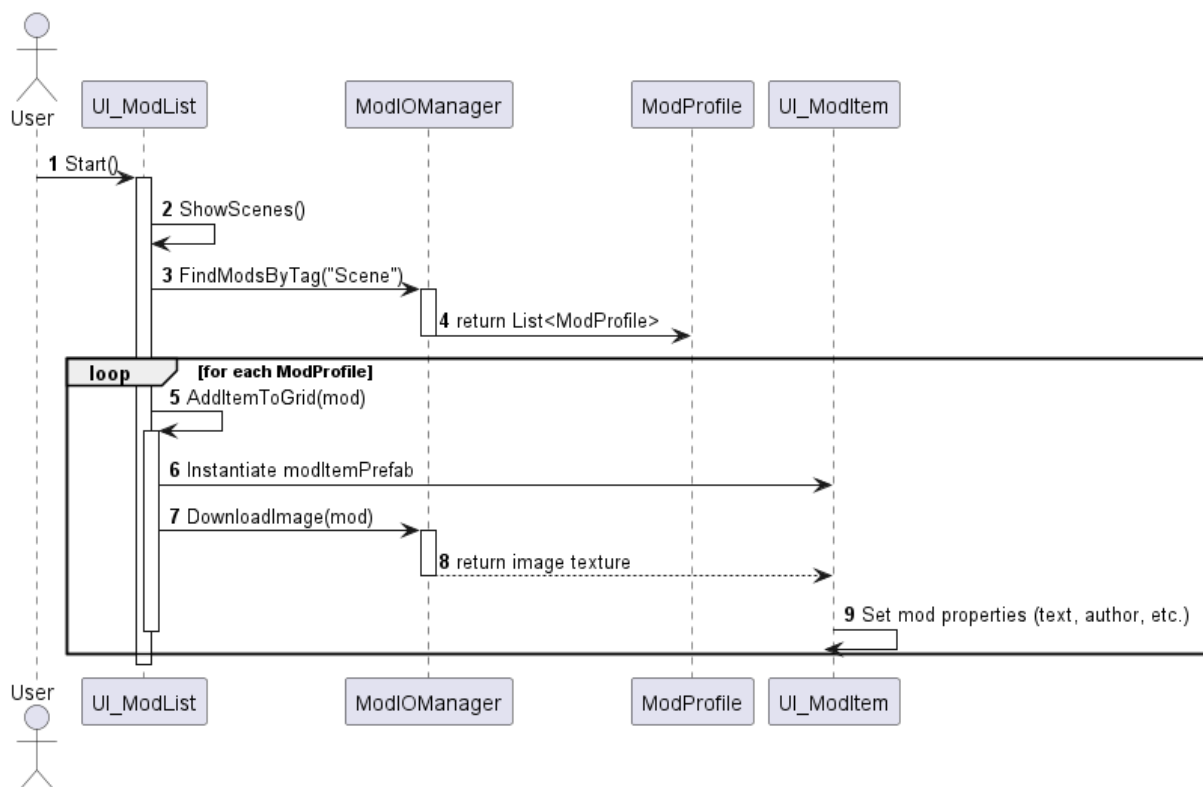


4.3.3 UI_MODLIST

Comme indiqué plus haut, UI_ModList affiche simplement une liste de contenu disponible à télécharger et affecte les références des contenus trouvés à UI_ModItem. Ce dernier peut exister plusieurs fois, donc ce que vous voyez est ce que vous pouvez télécharger. Si un mod n'existe pas, il n'est pas affiché, et donc il est impossible à télécharger. Cela rejoint le fait que faire des tests de nullité est redondant.

À noter que ShowScenes est appelé par Start, pour afficher une liste par défaut. Cependant, par des boutons, comme sur un site web, il est aussi possible d'afficher les scènes ou les contenus. Si vous regardez le diagramme de classes, ShowContent est aussi disponible. Il fait la même chose que ShowScenes, il change juste le tag.

Les tags ont été créés manuellement sur ModIO, via leur console de gestion, et servent uniquement à catégoriser les contenus.



4.3.4 TRANSFORM SYNCHRONIZER

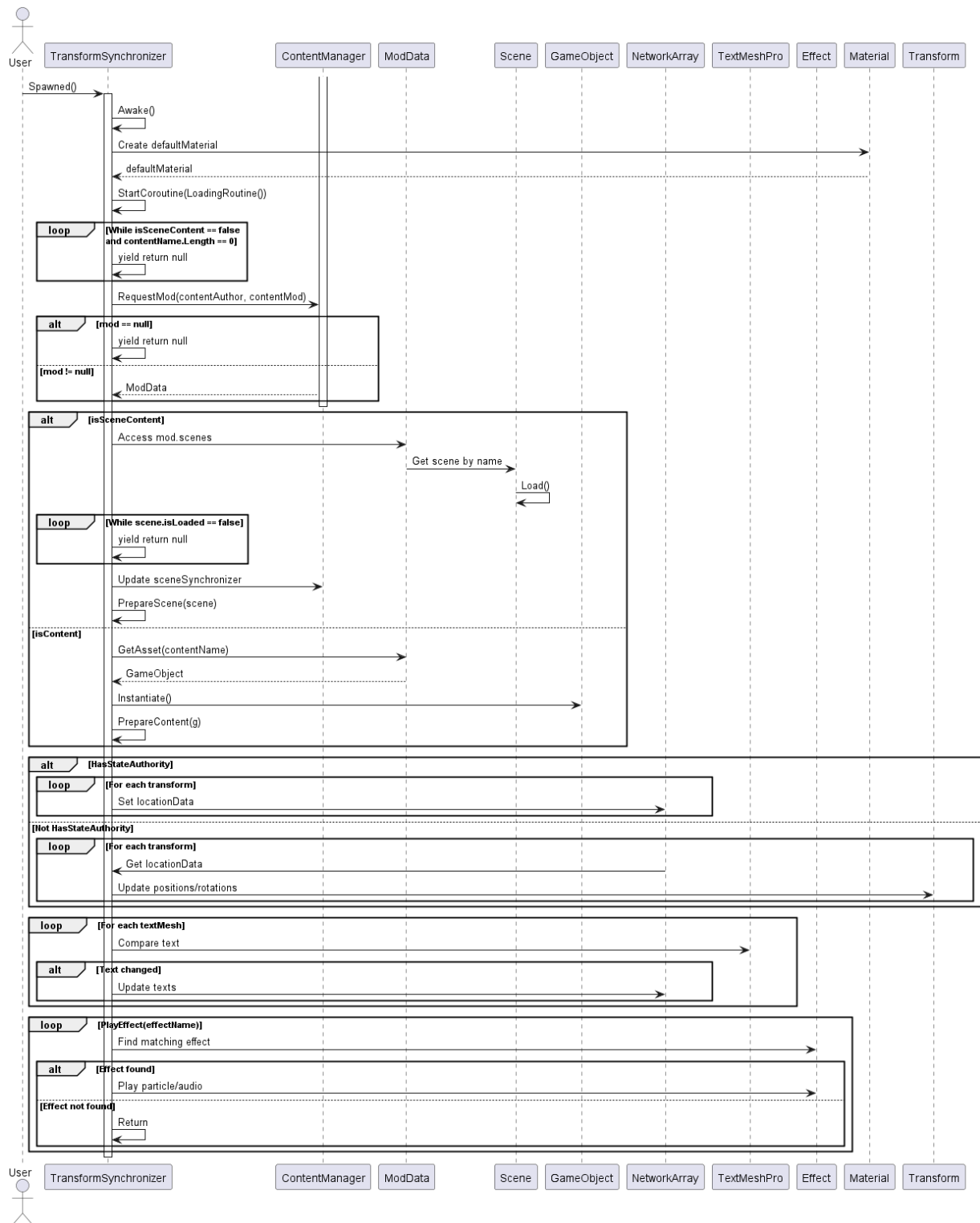
TransformSynchronizer gère le processus de chargement d'une scène ou d'un contenu dans une application. L'utilisateur (Fusion) déclenche l'action en appelant la méthode Spawned() sur l'objet TransformSynchronizer, ce qui lance une coroutine appelée LoadingRoutine(). Cette coroutine commence par une boucle qui attend que l'une des deux conditions soit remplie : soit la variable isSceneContent est vraie, indiquant qu'une scène doit être chargée, soit un nom de contenu (contentName) est défini. Pendant ce temps, la coroutine retourne temporairement null.

Ensuite, TransformSynchronizer fait appel au ContentManager pour récupérer un mod en utilisant RequestMod(contentAuthor, contentMod). Si le mod est inexistant, le processus retourne null. Si le mod est valide, les données du mod (ModData) sont utilisées pour continuer. Si isSceneContent est vrai, TransformSynchronizer accède à la liste des scènes disponibles dans le mod, puis utilise le nom de la scène pour la charger via l'objet Scene avec la méthode Load(). Une boucle s'assure que la scène est complètement chargée (scene.isLoaded == true) avant de synchroniser les données de la scène et de la préparer avec la méthode PrepareScene().

Dans le cas où isSceneContent est faux, ContentManager récupère un asset correspondant au nom de contenu via GetAsset(contentName). Cet asset est ensuite instancié en tant que GameObject avec Instantiate() et préparé via la méthode PrepareContent().

Le processus est structuré avec des boucles et des conditions qui permettent de gérer ces deux cas principaux : le chargement d'une scène ou celui d'un contenu sous forme de GameObject.

TransformSynchronizer synchronise aussi les effets. Les effets sont composés d'un système de particules et d'une source audio. L'objectif est de jouer les effets au même endroit et en même temps à travers le réseau, pour tous les utilisateurs.



4.4 CONVENTIONS D'ÉCRITURE

Unity propose sa propre convention d'écriture que j'implémente dans le projet, pour seulement ce que je code moi-même, je ne modifie en aucune manière les SDK qui m'ont été fourni, même s'ils utilisent des conventions différentes.

<https://unity.com/how-to/naming-and-code-style-tips-c-scripting-unity>

Si certain cas ne sont pas pris en charge, j'utilise ce que Microsoft suggère.

<https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions>

4.4.1 PARADIGME

Unity propose plusieurs paradigmes :

- Object-Oriented Programming
- Component-Based Programming
- Data-Oriented Programming
- Event-Oriented Programming

Il faut savoir qu'il est très compliqué de s'attacher à un paradigme particulier dans la nature d'Unity. En temps normal, vous allez principalement faire du Object-Oriented Programming et du Component-Based Programming.

Pour ce projet, c'est principalement du Object-Oriented Programming, mais je suis aussi obligé d'intégrer du Component-Based Programming.

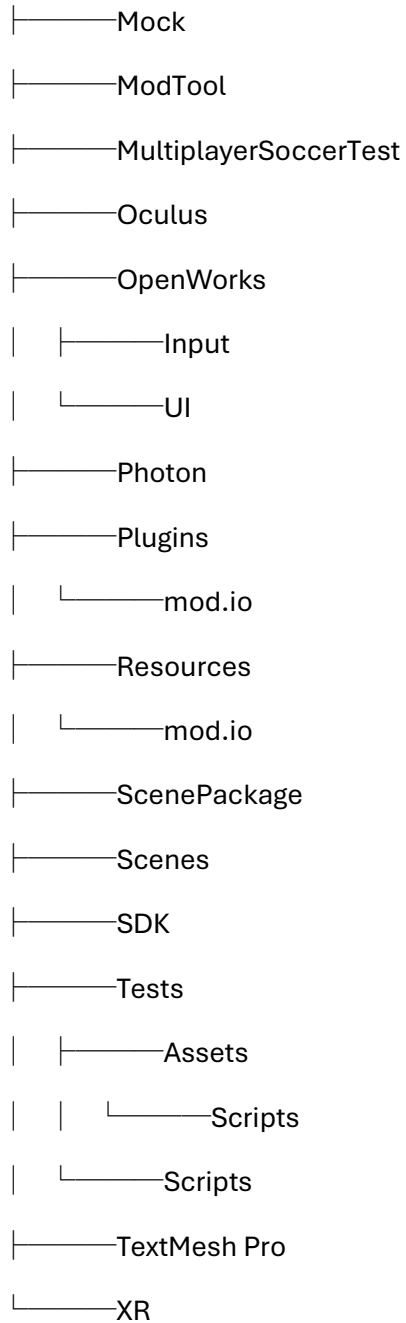
La raison de ce problème multi-paradigmes est assez simple. Quand vous voulez ajouter une classe à un objet physique présent dans la scène, vous créez tout d'abord une classe C#, puis vous devez l'ajouter à l'objet, ce qui crée un composant pour l'objet.

Prenez par exemple un script de mouvement avec de l'héritage : on veut que les cubes, sphères et pyramides se comportent différemment, ce qui nécessite de l'héritage, un aspect de la programmation orientée objet. Mais dès que vous ajoutez ces scripts aux objets, vous intégrez aussi des notions de Component-Oriented Programming.

C'est pour cela qu'il est compliqué sur Unity de se limiter à un seul paradigme.

4.5 STRUCTURE DU DOSSIER ASSETS

Assets

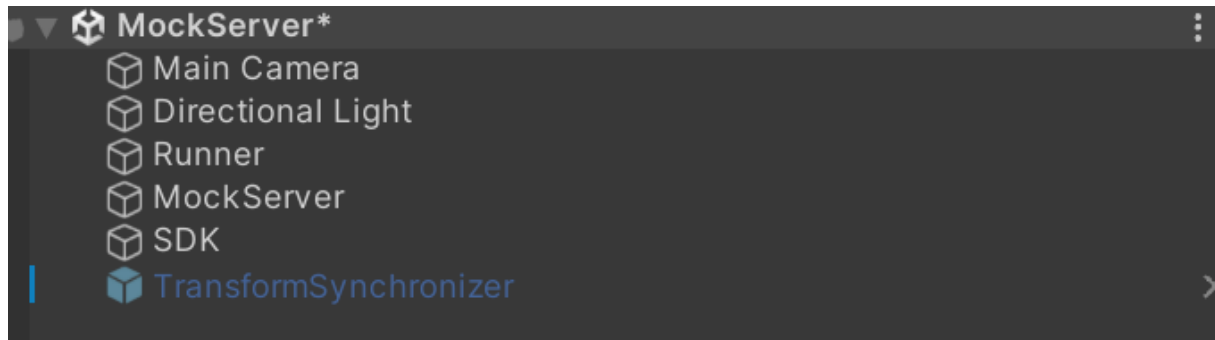


5 IMPLÉMENTATION

5.1 CHARGMENT DE SCÈNE

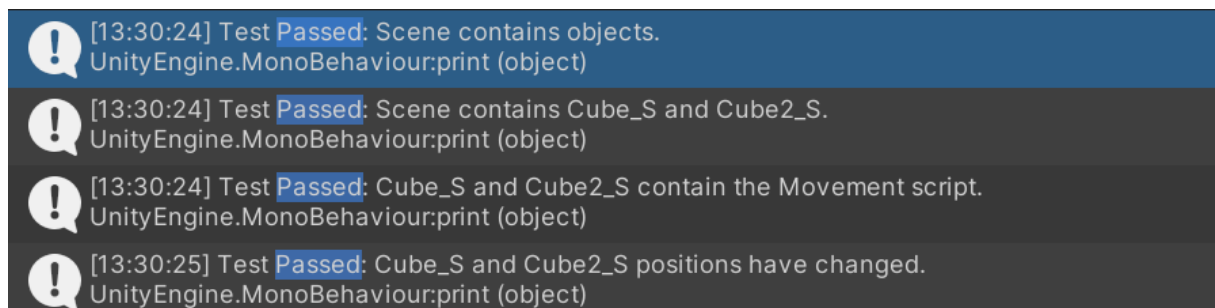
5.1.1 TESTS

Pour tester que la scène fonctionne bien, nous sommes obligés de simuler le serveur.



La scène de simulation contient ces éléments, mais le plus important est **MockServer**. Afin de tester le chargement de la scène, nous avons un mod créé par mes soins qui se trouve sur ModIO. Ce mod contient quelques cubes et un script de mouvement qui déplace aléatoirement les cubes.

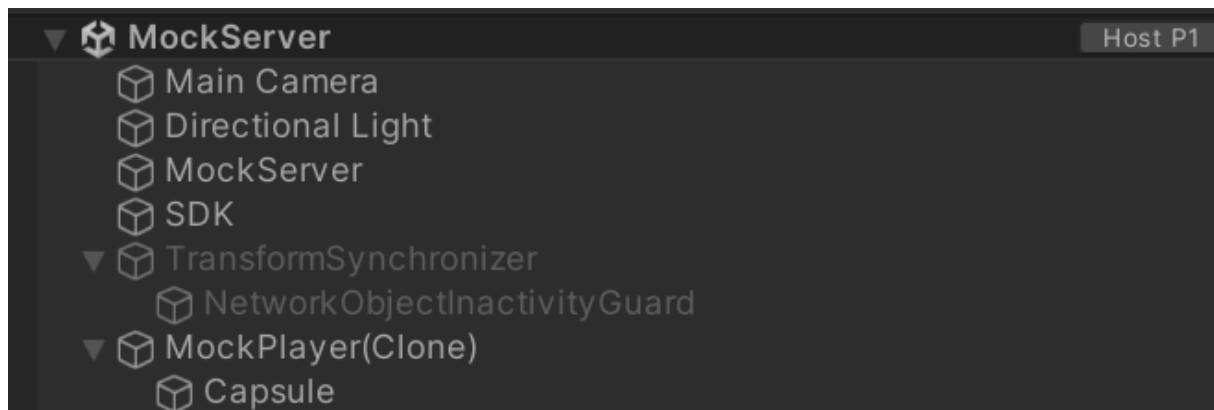
Il y a deux raisons à cela : premièrement, nous voulons nous assurer que nous pouvons charger une scène avec son contenu, les cubes, etc., ainsi que les scripts qui y sont associés ; et deuxièmement, nous voulons vérifier que le mouvement des cubes est fluide et non saccadé.



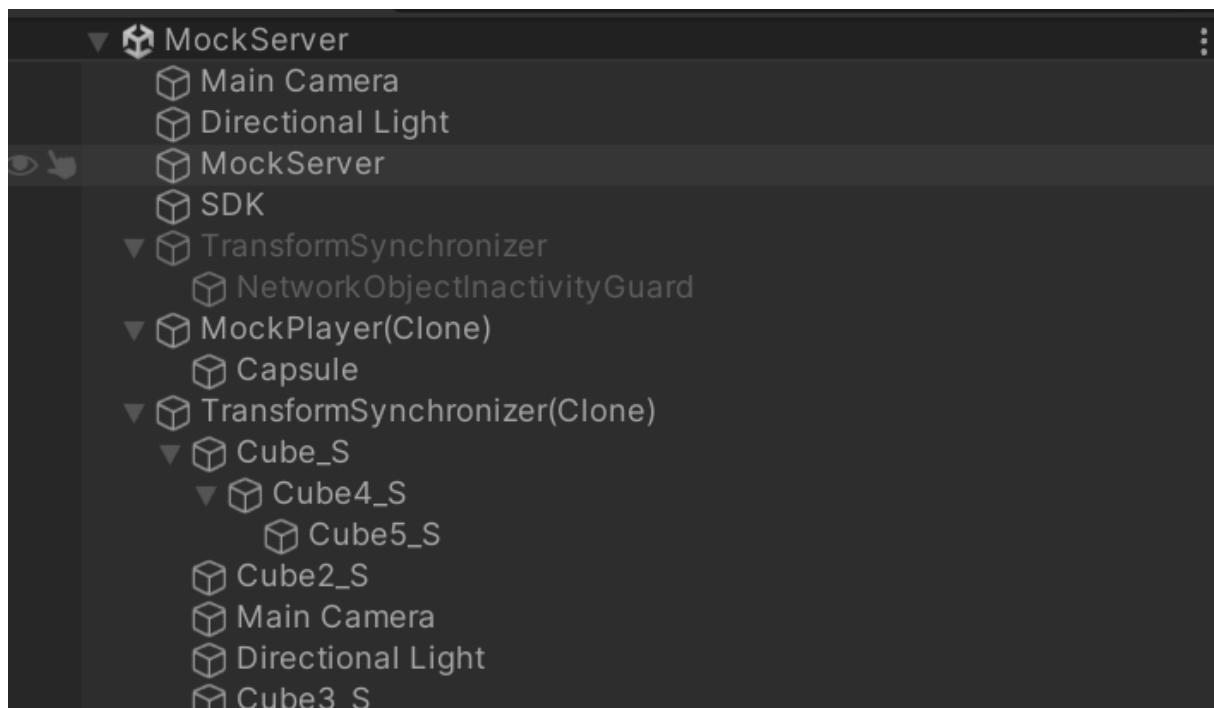
On peut voir que tous les tests passent.

Le premier test vérifie si la scène contient des objets, peu importe lesquels, tant que ce sont des objets.

Le deuxième test vérifie la présence de deux objets nommés Cube_S et Cube2_S, les noms correspondant à ceux que j'ai donnés aux cubes dans mon mod de test. Si, pour des raisons inconnues, la scène de test ne se chargeait pas, ces tests ne passeraient pas.



Voilà le comportement d'une scène avant le chargement d'un mod



Et voici la même scène après avoir chargé un mod, on peut voir que les cube que les tests cherche sont présent aussi.

5.1.2 CAS D'ERREURS

Dans Unity, en utilisant ModTool, il est impossible d'exporter des scènes, ce qui fait que le cas où la scène n'est pas présente dans un contenu additionnel est impossible. Il y a de toute façon toujours une scène.

5.2 CHARGEMENT DE CONTENU

Pour le chargement de contenu, cela fonctionne de la même manière que le chargement de scène, à la différence que nous chargeons aucune scène.

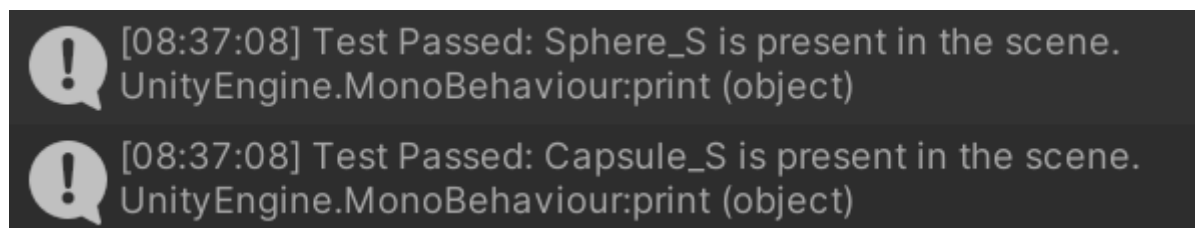
5.2.1 TESTS

Le comportement initial est le même que pour le chargement de scène.

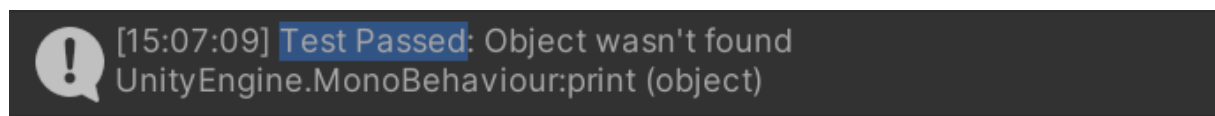
Les tests chargent deux contenus, un qui s'appelle Sphere_S et un autre Capsule_S.

Un problème auquel il faut faire attention est que pour charger du contenu, vous devez utiliser Instantiate(GameObject). Unity charge donc l'objet instancié dans la scène courante, mais rajoute "(Clone)" à la fin de chaque objet chargé par Instantiate().

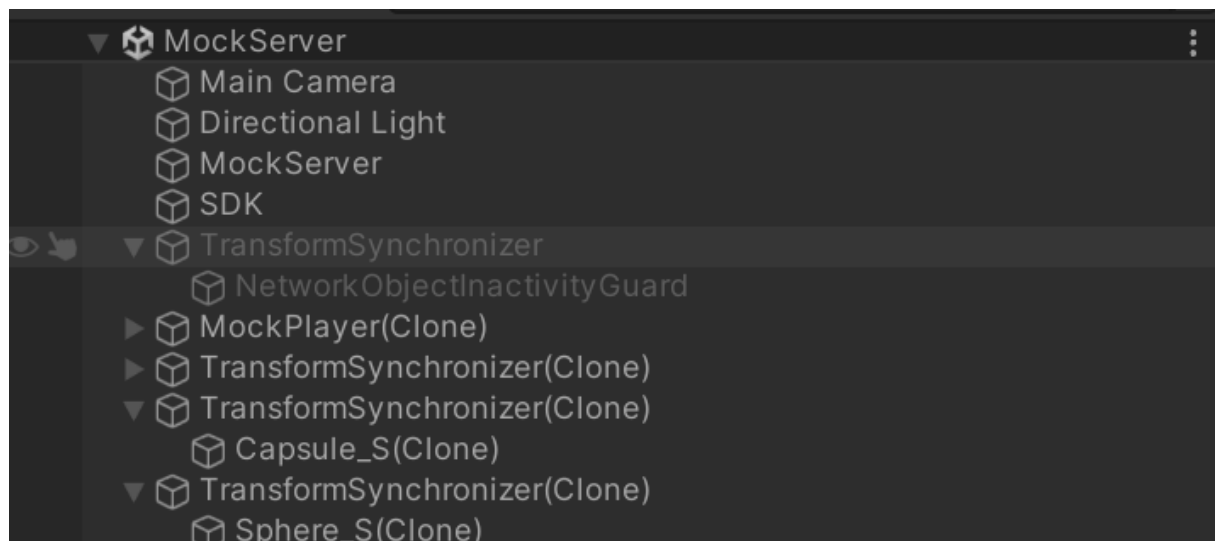
Il faut donc dans les tests attendre Sphere_S(Clone) et Capsule_S(Clone) et non Sphere_S et Capsule_S. Ce n'est pas un problème bloquant car ce comportement est le même à chaque fois, mais c'est tout de même une chose à prendre en compte.



Ce test a pour but de tester un cas où il n'y a pas d'objets à charger.



On peut voir que les tests passent.



Et voici le comportement après le chargement des deux contenus, on peut voir Capsule_S(Clone) et Sphere_S(Clone).

5.3 MOCKING DE SERVER ET CLIENT

Afin de tester que le test fonctionne sur un client et un serveur, j'ai dû implémenter un mock pour le client et le serveur.

Comme indiqué plus haut, lorsqu'un client se connecte au projet, il devient un serveur, et on peut exécuter les tests sur ce serveur, mais nous voulons également faire de même pour un client. La structure reste exactement la même, à la différence que, au lieu d'instancier des objets par le chargement de scène et de contenu, le SDK QuaD se charge de synchroniser les objets. Vous trouverez donc tout ce qui se trouve sur le serveur. Une différence majeure étant que je dois désactiver les scripts serveur.

```
if (!Runner.IsServer)
{
    Component[] quad_ServerBehaviours = GetComponentsInChildren<IQuad_ServerBehaviour>() as Component[];
    foreach (Component quad_ServerBehaviour in quad_ServerBehaviours)
    {
        Destroy(quad_ServerBehaviour);
    }
}
```

Tous les scripts serveur implémentent l'interface `IQuad_ServerBehaviour`, et je supprime donc ces scripts pour le client. Nous voulons tout d'abord éviter que le client exécute des scripts par souci de performances, mais cela protège aussi les autres clients de comportements malveillants, tels que des scripts se donnant des avantages à soi-même. Seul le serveur doit exécuter les scripts serveur.

5.3.1 PROBLÈMES RENCONTRÉS

J'ai eu quelques soucis à implémenter ce mocking en raison de la fragilité de la séquence d'appel. Par exemple, j'ai rencontré ce problème où le serveur essayait de faire télécharger du contenu au joueur avant que ce dernier ne soit identifié sur ModIO. ModIO étant la plateforme où se trouve le contenu, il est donc impossible de télécharger du contenu depuis ModIO si vous n'êtes pas authentifié.

5.3.1.1 RÉOLUTION

Grâce à des événements, j'ai pu faire attendre le serveur la connexion sur ModIO du client, puis lui permettre de télécharger du contenu.

5.4 LISTAGE DE SCÈNE ET CONTENU

Afin de donner à l'utilisateur un moyen d'afficher du contenu additionnel, j'ai dû créer des classes qui me permettaient de créer une liste simple.



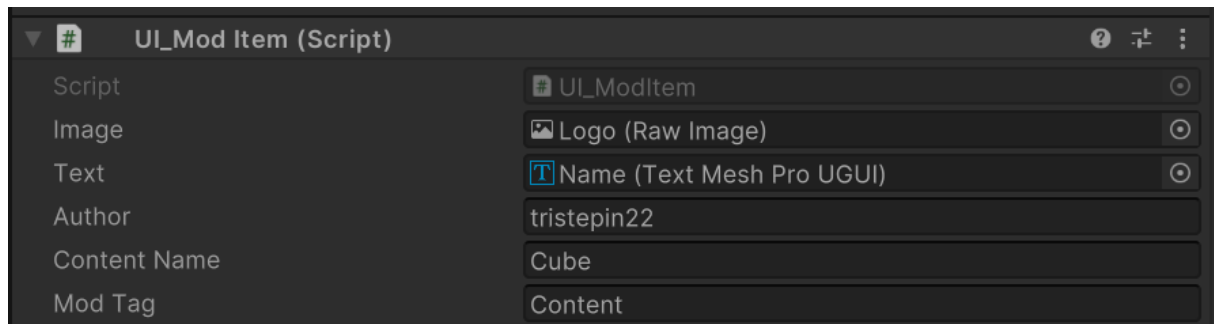
Voici comment la liste s'affiche : elle se présente sous forme d'une grille de carrés. Lorsque vous cliquez sur un panneau, le contenu additionnel est chargé. Il est également possible de cliquer sur les boutons en haut pour sélectionner le type de contenu à afficher.

```

public class UI_ModItem : MonoBehaviour
{
    [SerializeField] public RawImage image;
    [SerializeField] public TextMeshProUGUI text;
    public string author;
    public string contentName;
    public string modTag;
    0 références
    public void Click()
    {
        ContentManager.main.ProcessMod(this);
    }
}

```

Chaque panneau a cette classe attachée, ce qui permet de référencer les informations liées au contenu.



Voici à quoi cela ressemble sur Unity.

L'image et le texte sont auto-référencés grâce à un prefab. L'auteur, le contentName et le modTag sont référencés par UI_ModList.

Ces informations me permettent de retrouver le contenu sur ModIO.

5.5 MODIOMANAGER

ModIOManager est le lien entre ModIO et le projet.

Il faut savoir qu'avec ModIO, vous ne téléchargez rien directement, vous vous abonnez à un contenu, et il est téléchargé, mais jamais deux fois. Le contenu est retéléchargé uniquement si une mise à jour est détectée ou si le contenu a été supprimé localement sur votre disque.

5.5.1 INITIALISATION ET CONNEXION

```
SteamAPI.Init();  
while (!SteamManager.Initialized)  
    yield return null;
```

Tout d'abord, j'ai besoin d'initialiser Steam afin d'utiliser Steam comme moyen de connexion automatique. Ensuite, j'attends que Steam ait terminé de s'initialiser.

```
Result result = ModIOUnity.InitializeForUser("default");  
if (!result.Succeeded())  
    yield return 0;
```

J'initialise ensuite ModIO et je quitte ma coroutine si ModIO n'a pas réussi à s'initialiser.

```
byte[] k_unSecretData = System.BitConverter.GetBytes(0x5444);
Debug.Log("Requesting encrypted app ticket.");
SteamAPICall_t handle = SteamUser.RequestEncryptedAppTicket(k_unSecretData, sizeof(uint));
yield return new WaitForSeconds(2f);
OnInit();
```

Je démarre ensuite la séquence de connexion via Steam. Pour des raisons qui me sont inconnues, Handle peut être null avant que OnInit() soit appelé. Malheureusement, je n'ai pas trouvé de moyen d'attendre un événement de Steam, que je pourrais recevoir, afin d'éviter d'attendre un délai fixe de 2 secondes.

```
ModIOUnity.GetTermsOfUse(async (response) =>
{
    if (response.result.Succeeded())
    {
        termsHash = response.value;

        await ModIOUnityAsync.AuthenticateUserViaSteam(steamToken, null, termsHash.hash);
        Debug.Log("accepted");
    }
    else
    {
        Debug.LogError("Failed to authenticate: " + response);
    }
}
```

Je me connecte ensuite à Steam, après avoir accepté les conditions d'utilisation (TOS) de ModIO. Il est à noter que cette étape est nécessaire uniquement si vous utilisez un moyen de connexion tiers. Le type de connexion proposé par ModIO ne nécessite pas cette étape.

```
subscribedMods = GetSubscribedMods();
```

Après m'être connecté, je récupère tous les mods auxquels je me suis abonné, et j'utilise cette variable à travers ModIOManager.

5.5.2 RÉOLUTION DE PROBLÈMES

```
encryptedAppTicketCallback = Callback<EncryptedAppTicketResponse_t>.Create(OnInit);
```

J'avais commencé par créer un callback en utilisant ce que Steam proposait, mais cette solution ne fonctionnait pas.

J'ai donc essayé plusieurs autres manières de faire un callback, et j'ai finalement trouvé une autre solution que Steam proposait.

```
encryptedAppTicketCallResult = CallResult<EncryptedAppTicketResponse_t>.Create(OnInit);
encryptedAppTicketCallResult.Set(handle);
```

Le nom était très similaire, mais son comportement était très différent. Callback n'était jamais appelé, même si Steam effectuait une action censée déclencher un callback.

En revanche, CallResult fonctionnait dans la même situation : il suffit d'utiliser CallResult.Set(SteamAPICall_t).

Avec Callback, il est impossible d'utiliser Callback.Set().

5.5.3 ABONNEMENT

Comme indiqué plus haut, je ne télécharge pas de contenu à travers ModIOManager, je m'abonne simplement à un contenu.

```
public static async Task SubscribeToMod(ModProfile modToSubscribe)
{
    if (subscribedMods.Any(mod => mod.id == modToSubscribe.id))
        return;

    Result result = await ModIOUnityAsync.SubscribeToMod(modToSubscribe.id);
    if (!result.Succeeded())
    {
        Debug.LogError($"SubscribeToMod failed: {result.message}");
        return;
    }

    Debug.Log($"Subscribed to mod: {allMods.First(mod => mod.id == modToSubscribe.id).name}");
}
```

Cette fonction se charge de s'abonner à du contenu. Je vérifie d'abord si le contenu n'est pas déjà présent dans ma liste d'abonnements. Si ce n'est pas le cas, je m'abonne. Le contenu est ensuite automatiquement téléchargé par ModIO.

5.5.4 RECHERCHE

ModIO se charge aussi de rechercher des contenus spécifiques.

```
public static async Task<ModProfile> FindModByNameAndAuthor(string name, string author)
{
    ResultAnd<ModPage> resultAnd = await ModIOUnityAsync.GetMods(new SearchFilter());
    if (!resultAnd.result.Succeeded())
    {
        Debug.LogError($"GetMods failed: {resultAnd.result.message}");

        return new ModProfile();
    }

    // Will search for all the subscribers mods first, then will search in the mod.io DataBase
    try
    {
        var matchingMod = subscribedMods.FirstOrDefault(mod => mod.name == name && mod.creator.username == author);
        if (matchingMod.id != 0)
        {
            return matchingMod;
        }
        else
        {
            return SearchDataBase(name, resultAnd);
        }
    }
    catch (ArgumentNullException ex)
    {
        return SearchDataBase(name, resultAnd);
    }
}
```

Cette fonction va d'abord chercher dans la liste d'abonnements si le mod est déjà présent. Si ce n'est pas le cas, elle va ensuite effectuer une recherche sur ModIO.

5.6 CONTENTMANAGER

ContentManager se charge de charger les contenus via ModTool et les ajoute ensuite aux TransformSynchronizers pour assurer leur synchronisation.

5.6.1 CHARGEMENT DE CONTENU

```

1 référence
public void SetScene(string authorName, string modName)
{
    var g = NetworkRunner.Instances[0].Spawn(TransformSynchronizerPrefab, null, null, null, (Runner, NO) =>
    {
        TransformSynchronizer ts = NO.GetComponent<TransformSynchronizer>();
        ts.isSceneContent = true;
        ts.contentAuthor = authorName;
        ts.contentMod = modName;
    });
}

1 référence
public void SpawnContent(string authorName, string modName, string contentName)
{
    var g = NetworkRunner.Instances[0].Spawn(TransformSynchronizerPrefab, null, null, null, (Runner, NO) =>
    {
        TransformSynchronizer ts = NO.GetComponent<TransformSynchronizer>();
        ts.isSceneContent = false;
        ts.contentAuthor = authorName;
        ts.contentMod = modName;
        ts.contentName = contentName;
        contentSynchronizers.Add(ts);
    });
}

```

La première étape pour charger un contenu consiste à appeler `SpawnContent` ou `SetScene`. Ces méthodes ajoutent un `TransformSynchronizer` à `Fusion` pour gérer la synchronisation du contenu.

```

1 référence
public ModData RequestMod(string authorName, string modName)
{
    string key = authorName + "_" + modName;

    if(mods.ContainsKey(key))
    {
        return mods[key];
    }

    if (modRequests.ContainsKey(key))
        return null;

    if (loadingMod != null && loadingMod.authorName == authorName && loadingMod.modName == modName)
        return null;

    modRequests.Add(key, new ModData(authorName, modName));

    return null;
}

```

`TransformSynchronizer` se chargera d'appeler cette fonction au bon moment. Ensuite, `ContentManager` appellera `DownloadManager`, qui, à son tour, appellera `ModIOManager` pour télécharger les mods.

5.7 DOWNLOADMANAGER

DownloadManager se charge de connecter ModIOManager et ContentManager, avec pour objectif de pouvoir remplacer ModIO par une autre solution sans impacter ContentManager.

5.7.1 TÉLÉCHARGEMENT

Comme indiqué plus haut, ModIOManager, et donc DownloadManager, ne se charge pas réellement du téléchargement direct des fichiers. Toutefois, du point de vue de l'utilisateur, le processus fonctionne comme un téléchargement.

```
private void Awake()
{
    Debug.Log("calling modio");
    StartCoroutine(ModIOManager.Start());
    main = this;
}
```

La première chose à faire est d'initialiser ModIOManager.

```
1 référence
public async Task<long> DownloadModByAuthorAndName(string modName, string authorName)
{
    while (true)
    {
        Result authResult = await ModIOUnityAsync.IsAuthenticated();

        if (authResult.Succeeded())
        {
            Debug.Log("User authenticated successfully!");
            break;
        }

        Debug.Log("Waiting for user authentication...");
        await Task.Yield();
    }
    while (!ModIOManager.afterAuthCheck)
    {
        await Task.Yield();
    }

    ModProfile mod = await ModIOManager.FindModByNameAndAuthor(modName, authorName);
    if (mod.id != 0)
    {
        await ModIOManager.SubscribeToMod(mod);
    }
    return mod.id;
}
```

Ensuite, nous téléchargeons le contenu choisi par l'utilisateur.

5.7.2 PROBLÈMES RENCONTRÉ

Il arrivait souvent que `DownloadModByAuthorAndName` soit appelé avant que `ModIOManager` ait fini de s'initialiser, à cause du délai d'attente de 2 secondes. Par conséquent, je suis obligé d'attendre que, d'une part, `ModIOManager` soit initialisé, et d'autre part, que la variable `subscribedMods` dans `ModIOManager` soit correctement remplie avec la liste des abonnements de l'utilisateur.

5.7.3 RÉOLUTION DES PROBLÈMES

Après avoir corrigé l'erreur liée à la connexion Steam, j'ai pu supprimer l'attente de l'initialisation de `ModIO`, puisque Steam ne peut plus être connecté sans que `ModIO` soit initialisé.

```
public async Task<long> DownloadMod(string modName)
{
    while (!ModIOManager.afterAuthCheck)
    {
        await Task.Yield();
    }
}
```

Cependant, je dois tout de même attendre que l'utilisateur soit connecté et que la liste des abonnements soit remplie.

5.8 TRANSFORM SYNCHRONIZER

`TransformSynchronizer` s'assure que le contenu et sa position soient synchronisés à travers le réseau en utilisant `Fusion`.

```
var scene = mod.mod.scenes.FirstOrDefault();
string sceneName = scene.name;

scene.Load();
```

`TransformSynchronizer` charge notamment les scènes. Il est important de noter que `ModTool` peut contenir plusieurs scènes, mais dans notre cas, nous ne voulons que la première scène de chaque contenu.


```
var g = (GameObject)Instantiate(mod.mod.GetAsset(contentName),transform.position, transform.rotation);
g.name = contentName;
```

Il instancie aussi des objets, comme un cube, à condition que leur nom se termine bien par _S.

```
0 références
public override void FixedUpdateNetwork()
{
    print("Tick 0");

    if (HasStateAuthority)
    {
        for (int i = 0; i < counter; i++)
        {
            locationData data;
            data.localPosition = transforms[i].localPosition;
            data.localRotation = transforms[i].localRotation;
            locations.Set(i, data);
        }
    }
    /*
    else
        for (int i = 0; i < counter; i++)
        {
            transforms[i].SetLocalPositionAndRotation(locations[i].localPosition, locations[i].localRotation);
        }
    */

    base.FixedUpdateNetwork();
}
```

Ensuite, dans FixedUpdateNetwork, j'override la fonction provenant de NetworkBehaviour et je synchronise toutes les positions et rotations qui doivent être synchronisées.

```
0 références
public override void Render()
{
    print("Render 0");

    if (!HasStateAuthority)
    {
        for (int i = 0; i < counter; i++)
        {
            transforms[i].SetLocalPositionAndRotation(locations[i].localPosition, locations[i].localRotation);
        }
    }

    base.Render();
}
```

Et je fais de même dans Render.

```
if (!Runner.IsServer)
{
    var quad_ServerBehaviours = GetComponentsInChildren<IQuad_ServerBehaviour>();

    foreach (var quad_ServerBehaviour in quad_ServerBehaviours)
    {
        Destroy(quad_ServerBehaviour as Component);
    }
}
```

Il est aussi important de noter que les scripts serveur ne doivent pas être présents sur le client. Il suffit simplement de les détruire. Chaque script qui doit être exécuté sur le serveur hérite de IQuad_ServerBehaviour. Si un script n'hérite pas de IQuad_ServerBehaviour, il sera exécuté

également sur le client. Cela n'est pas un problème de sécurité, car si un script n'hérite pas de `IQuad_ServerBehaviour`, il n'a aucun accès aux fonctions qui se trouvent sur le serveur.

```
1 référence
public void PlayEffect(string effectName)
{
    for (int i = 0; i < effectsCounter; i++)
    {
        if (effects[i].name == effectName)
        {
            RPC_PlayEffect(i, effects[i].transform.localPosition, effects[i].transform.localRotation);
            return;
        }
    }
}

[Rpc(RpcSources.StateAuthority, RpcTargets.All, HostMode = RpcHostMode.SourceIsServer)]

1 référence
public void RPC_PlayEffect(int index, Vector3 localPosition, Quaternion localRotation)
{
    effects[index].transform.SetLocalPositionAndRotation(localPosition, localRotation);
    effects[index].GetComponent<AudioSource>().Play();
    effects[index].GetComponent<ParticleSystem>().Play();
}
```

`TransformSynchronizer` va jouer les effets, la demande pour jouer les effets est initiée par le serveur, et ils ont joué pour tous les joueurs, la position est aussi donnée, mais les effets ne sont jamais synchronisés pour chaque tick réseau, la raison étant que les effets ne durent que quelques secondes, voire moins, il n'y a pas besoin de les synchroniser.

Cependant, le type de particle, ou le clip audio qui doit être joué par `transformSynchroniser` n'est pas donné, du fait qu'ils sont déjà affectés à l'objet grâce à un prefab

5.8.1 PROBLÈMES RENCONTRÉS

Un problème que j'ai eu, c'est que `FixedUpdateNetwork`, qui est censé être appelé sur tous les clients et le serveur, ne s'exécutait pas sur le client, ce qui n'est pas normal, mais je n'ai pas trouvé de solution.

`FixedNetworkBehaviour` étant écrit dans `NetworkBehaviour`, une classe appartenant à Fusion, je ne peux pas modifier Fusion pour faire fonctionner mon code. J'ai consulté la documentation, les forums, lu le code source, changé des paramètres, mais je ne vois pas ce qui pourrait causer ce problème.

6 PROBLÈMES CONNU

6.1 MODTOOL

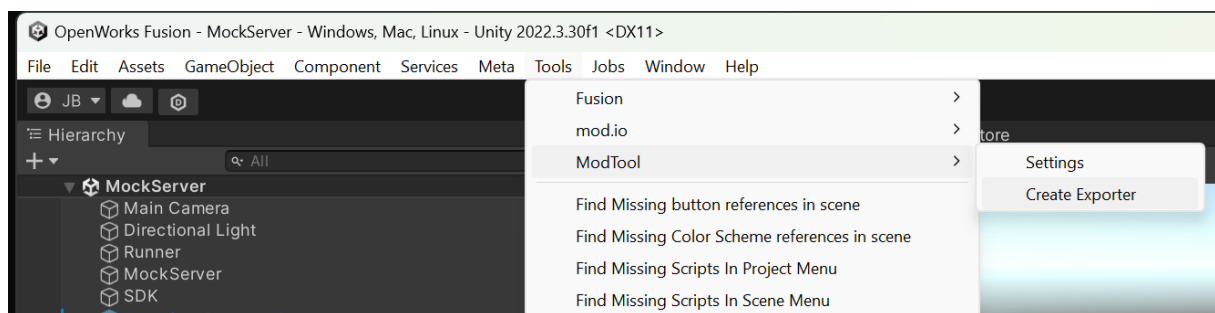
Nous avons rencontré beaucoup de problèmes avec ModTool, qui bloquaient totalement le développement. Nous avons finalement pu trouver un workaround :

6.1.1 VERSION

Nous utilisons la dernière version de ModTool

6.1.2 CRÉATION D'UN EXPORTER

Dès que vous avez installé modTool, créez un exporter.



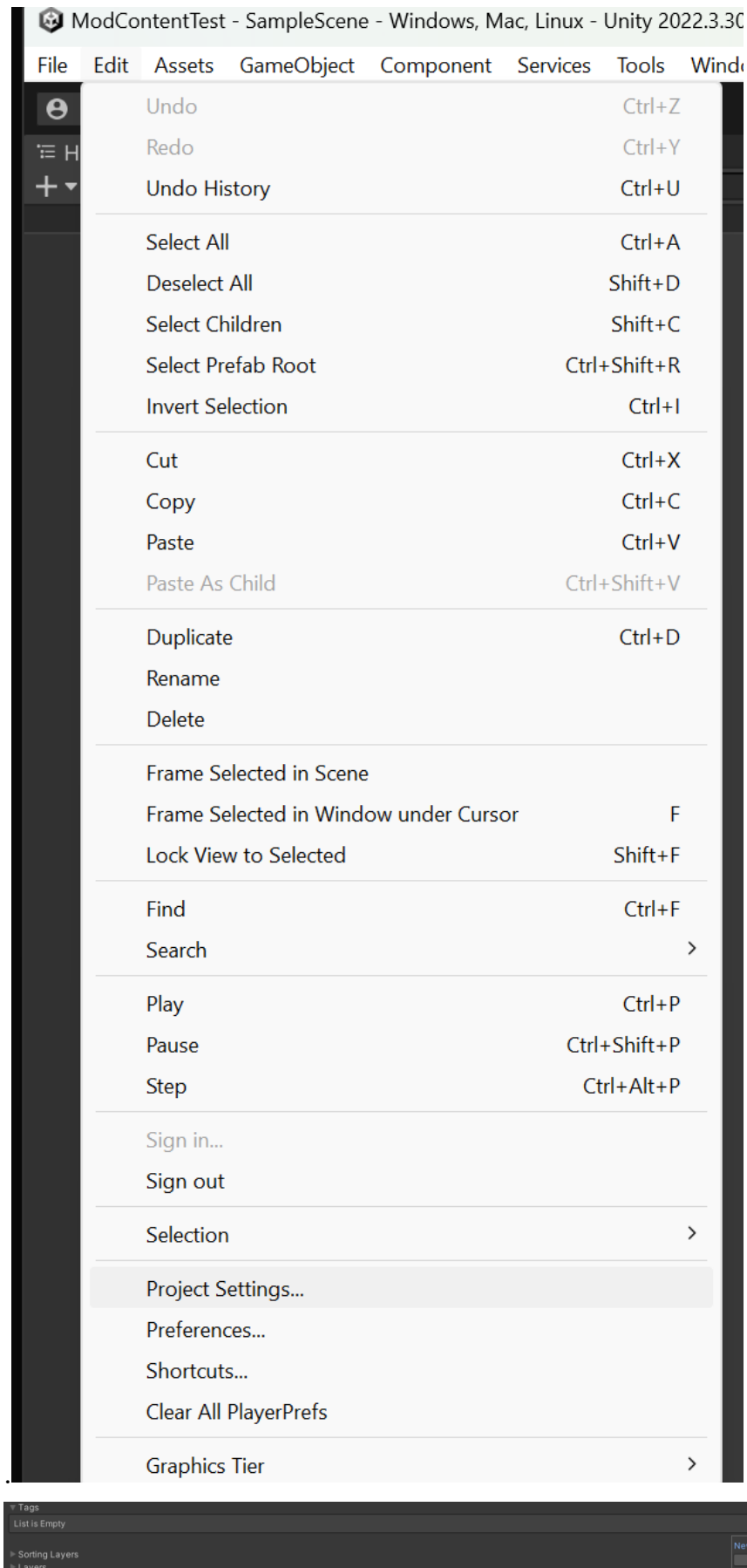
Unity va créer, à la racine du projet, un package que vous devrez installer sur un nouveau projet.

6.1.3 IMPORTER L'EXPORTER

Lorsque la création du nouveau projet est terminée, il est impératif de suivre cette manipulation :

Allez dans les paramètres du projet, sous "Tags and Layers", et créez un nouveau tag.

Vous pouvez ensuite importer l'exporter



7 CONCLUSION

7.1 OBJECTIFS ATTEINTS ET NON ATTIENTS

En prenant le cahier des charges et les tâches effectuées, tous les points qui se trouvent dans le cahier des charges ont été implémentés.

Voici les points abordés par le cahier des charges :

- Étude du SDK (1 semaine)
 - Comprendre le fonctionnement du SDK
 - Test du SDK
- Exploration et chargement d'une scène sur le serveur et les clients (1 semaine)
 - Explorateur pouvant lister les scènes disponibles sur le cloud (Mod.IO)
 - Téléversement d'une scène sur le cloud.
 - Chargement de la scène courante sur le serveur et exécution du code
 - Téléchargement de la scène sur le client sans exécution du code
- Téléversement et Chargement d'objets(modèle 3D + code) (1 semaine)
 - Téléversement d'objets
 - Explorateur pouvant lister les objets
 - Instanciation (chargement) d'objets synchronisés pendant l'exécution
- Tâches à réaliser si le temps le permet :
 - Prédiction de Transformations et d'Effets visuels coté client
 - Anticiper le mouvement des mains ou de la tête de l'utilisateur

Et si on compare le projet, on peut voir que toutes les tâches ont été réalisées, même l'objectif secondaire.

7.2 POINTS POSITIFS / NÉGATIFS

J'ai bien aimé le projet, il était intéressant et mélangeait plusieurs facettes, entre l'asynchrone, l'utilisation de SDK, et la recherche de solutions à des problèmes. Je me suis aussi donné un défi de faire le projet proprement, depuis les schémas jusqu'à la ligne de code.

Mais j'ai tout de même des choses que j'ai moins appréciées dans le projet, notamment diverses erreurs liées à ModTool et à l'asynchrone. Même si je trouve l'asynchrone intéressant, il devient vite compliqué : attendre des appels, etc. Il est donc très facile d'introduire des erreurs ou de la complexité. Mais dans l'ensemble, j'ai bien apprécié le projet.

7.3 DIFFICULTÉS PARTICULIÈRES

7.3.1 PROBLÈMES RENCONTRÉS LORS DE LA CONCEPTION

J'avais longuement réfléchi à la manière de mettre en place les fonctionnalités. Traduire un diagramme de cas d'utilisation en un diagramme de séquence était très difficile.

Même si je savais comment le projet devait fonctionner dans sa globalité, je ne savais pas comment chaque fonctionnalité allait interagir avec les différents SDK et bibliothèques, surtout en prenant en compte l'asynchrone. J'ai beaucoup changé de direction et refait les diagrammes.

7.3.2 PROBLÈMES RENCONTRÉS LORS DE L'IMPLÉMENTATION

J'ai eu beaucoup de soucis avec ModTool et ModIO lors de l'implémentation de Steam. Je pensais avoir déjà tout vu durant mon stage par rapport à ces outils, mais finalement, c'était l'inverse.

ModTool a décidé de ne plus fonctionner en plein milieu du projet, alors que dans l'autre projet fait durant le stage, je n'avais jamais eu de tels problèmes.

Par la suite, j'ai eu des problèmes en essayant d'intégrer Steam à ModIO, surtout avec l'asynchrone. J'ai passé beaucoup de temps à résoudre ce problème.

7.4 SUITES POSSIBLE POUR LE PROJET

Je sais que le projet va être intégré à un des jeux de Raptor Lab pour remplacer la solution SteamWorkshop, mais cela va demander un peu plus de travail.

Je vais devoir notamment adapter ce projet pour qu'il puisse facilement fonctionner avec les autres projets de Raptor Lab.

Particulièrement quand toute la procédure de chargement se lance, parce que pour l'instant, seulement la liste de mods et la scène des tests peuvent charger des mods. Mais par la suite, ce sera une vraie personne, en VR, qui va initialiser la procédure de chargement, donc je dois en tenir compte et adapter mon projet.

7.5 PLANIFICATION FINALE

Si on prend comme référence le journal de travail. Le projet s'est déroulé comme suit :

Type	Durée
Analyse	Du 15.11.2024 au 15.11.2024
Conception	Du 18.11.2024 au 18.11.2024
Implémentation	Du 20.11.2024 au 09.12.2024

Tout le projet a été relativement rapide, entre l'analyse et l'implémentation, sans prendre en compte la synchronisation des effets.

Je suis passé 5 jours entiers à résoudre des problèmes. J'ai eu de la chance de savoir exactement quoi implémenter et comment, sinon je n'aurais pas eu le temps d'implémenter beaucoup de choses.

J'ai aussi analysé, conceptualisé et implémenté la synchronisation des effets, du 10.12.2024 au 17.12.2024.

8 ANNEXES

8.1 SOURCES

1. **Unity.** Télécharger Unity. [En ligne] <https://unity.com/fr/products/unity-student>.
2. —. Unity Manual. [En ligne] <https://docs.unity3d.com/2022.3/Documentation/Manual/UnityManual.html>.
3. **ModIO.** Documentation ModIO. [En ligne] <https://docs.mod.io/>.
4. —. Accueil. [En ligne] <https://mod.io/>.
5. **Unity.** ModTool - AssetStore. [En ligne] https://assetstore.unity.com/packages/tools/integration/modtool-75580?srltid=AfmBOoi3_d6d77jvDFhc4CgMhiRV8Fv1zdrPTFO5P14fh0INKMNIKp7.
6. **ModTool.** Documentation ModTool. [En ligne] <https://hellomeow.net/modtool/documentation/html/96083abe-78cf-43e9-a3f6-59c4e717b7e7.htm>.
7. **Unity.** Naming and code styles. [En ligne] <https://unity.com/how-to/naming-and-code-style-tips-c-scripting-unity>.
8. **Microsoft.** C# naming convention. [En ligne] <https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions>.

8.2 GLOSSAIRE

8.2.1 QU'EST-CE QU'UN OBJET EN 3D ?

Un **objet** en 3D est une représentation numérique d'un élément ou d'un modèle dans un espace tridimensionnel. Il peut s'agir de n'importe quel élément visible dans une scène 3D : un personnage, une voiture, un bâtiment, etc. Les objets 3D sont utilisés dans des domaines tels que les jeux vidéo, la modélisation 3D, la réalité virtuelle et la simulation. Ils sont composés de formes géométriques de base comme les sphères, les cubes, les cylindres, ou plus complexes.

Chaque objet a plusieurs attributs, tels que :

- **La position** : coordonnées de l'objet dans l'espace 3D (x, y, z).
- **La rotation** : angle selon lequel l'objet est orienté.
- **L'échelle** : taille de l'objet.
- **Les matériaux et textures** : des propriétés qui définissent l'apparence visuelle (couleur, texture).

8.2.2 QU'EST-CE QU'UN MESH ?

Un **mesh** (ou maillage en français) est la structure géométrique qui compose un objet 3D. Il s'agit d'un ensemble de points (appelés sommets ou "vertices") reliés entre eux par des arêtes (edges), formant des polygones (souvent des triangles ou des quadrilatères) qui définissent la surface de l'objet. En résumé, un mesh est la "carcasse" de l'objet 3D, qui lui donne forme et volume.

Les meshes peuvent être :

- **Static** : des objets fixes qui ne se déforment pas.
- **Dynamic** : des objets pouvant être déformés, animés, ou fracturés en temps réel.

Le mesh est généralement habillé d'une **texture** pour lui donner une apparence plus réaliste, comme des motifs de bois, de métal, de peau, etc. Il est aussi soumis à la lumière, aux ombres, et aux réflexions pour s'intégrer visuellement dans l'environnement 3D.

8.2.3 QU'EST-CE QU'UNE SCÈNE

Une **scène** est un conteneur qui représente un niveau ou un environnement dans un jeu ou une application. Elle peut inclure divers éléments tels que des modèles 3D, des lumières, des caméras, des effets sonores, des scripts et d'autres objets. Les scènes permettent aux développeurs de structurer et d'organiser les différents aspects d'un projet de manière modulaire. Chaque scène peut être indépendante ou interconnectée, permettant ainsi de créer des transitions fluides entre différents niveaux ou environnements dans le jeu. Une scène dans Unity est souvent sauvegardée sous la forme d'un fichier. Unity, et plusieurs scènes peuvent être chargées dans un projet. Les développeurs peuvent facilement passer d'une scène à l'autre à l'aide de scripts, ce qui permet de créer une expérience de jeu cohérente et immersive.

8.2.4 QU'EST-CE QU'UNE TEXTURE ?

Une **texture** dans notre contexte est une image appliquée à la surface d'un modèle 3D. Les textures ajoutent des détails visuels, comme des couleurs, des motifs et des nuances. Par exemple, une texture peut simuler la surface d'un matériau comme le bois, le métal ou la pierre, une texture est aussi un ensemble de pixels, chaque pixel pouvant être de la même couleur ou pas.

Les textures sont souvent utilisées en combinaison avec des matériaux, qui définissent comment la texture interagit avec la lumière et affecte l'apparence globale de l'objet. Dans Unity, les textures peuvent être importées et utilisées directement sur des objets 3D, et elles peuvent également être regroupées dans des **atlases de textures**.

8.2.5 QU'EST-CE QU'UN MATÉRIAU ?

Un matériau (Material) est un composant utilisé pour déterminer l'apparence visuelle d'un objet 3D ou 2D en contrôlant comment il interagit avec la lumière et les textures. Il permet de définir la couleur, la transparence, la réflectivité, et bien d'autres propriétés d'une surface. Voici les principaux aspects d'un matériau dans Unity :

- **Shaders** : Les matériaux sont associés à des shaders, qui sont des programmes définissant comment les pixels de l'objet doivent être rendus. Par exemple, un shader peut spécifier si la surface est opaque, transparente, brillante, etc.
- **Textures** : Un matériau peut utiliser des textures (images) pour donner des détails à la surface, comme des motifs ou des imperfections. Cela peut inclure une texture de couleur principale (albedo), une texture normale pour simuler des reliefs ou une texture de réflexion.
- **Propriétés visuelles** : Les matériaux permettent d'ajuster des paramètres tels que la couleur de base, la rugosité, l'intensité spéculaire (brillance), et d'autres effets comme l'émission de lumière.
- **Application** : Un matériau peut être appliqué à un objet en le rendant dans un mesh renderer ou un sprite renderer. Chaque matériau définit l'apparence finale de la surface de cet objet en fonction des interactions avec la lumière de la scène et des autres paramètres définis dans le shader.

En résumé, un matériau dans Unity est essentiel pour définir comment un objet apparaîtra visuellement dans la scène, que ce soit en termes de couleur, de texture, de brillance ou d'interaction avec la lumière.

8.2.6 QU'EST-CE QU'UN SHADER ?

Un shader est un petit programme qui détermine la manière dont les surfaces d'un objet sont rendues en fonction des interactions avec la lumière, les ombres, et d'autres éléments visuels. Les shaders jouent un rôle crucial dans la création des effets visuels sur les objets dans une scène 3D ou 2D. Voici les aspects principaux des shaders :

- **Fonctionnement** : Un shader détermine comment chaque pixel ou vertex d'un objet est calculé, influençant la couleur finale, la brillance, la transparence, les reflets, etc. Il est exécuté par la carte graphique pour optimiser le rendu des objets en temps réel.
- **Types de shaders** :
 - **Vertex Shader** : Il manipule les positions des sommets (vertex) d'un objet, modifiant ainsi la géométrie avant de la rendre visible.
 - **Fragment (Pixel) Shader** : Il calcule la couleur des pixels, définissant comment la lumière et les textures affectent l'apparence visuelle.
 - **Surface Shader** : Ce type de shader plus abstrait permet de définir des matériaux interactifs avec la lumière, facilitant la création de matériaux complexes.
- **Effets spéciaux** : Les shaders permettent de créer des effets visuels comme les réflexions, la transparence, la réfraction, les ombrages, et bien plus encore. Par exemple, un shader d'eau peut imiter les ondulations et les reflets à la surface.
- **Programmation** : Les shaders peuvent être écrits en différents langages comme HLSL (High-Level Shading Language) ou via Unity Shader Graph, un outil visuel permettant de créer des shaders sans avoir à coder.

8.2.7 QU'EST-CE QU'UNE ATLAS

Un atlas est une grande image qui regroupe plusieurs petites textures dans une seule texture, afin d'optimiser le rendu des objets dans Unity. Il est principalement utilisé dans les projets 2D et 3D pour réduire les appels de rendu et améliorer les performances.

8.2.8 QU'EST-CE QU'UN PRÉFAB ?

Un **préfab** désigne un **modèle réutilisable d'objet de jeu**. Il s'agit d'une configuration d'un ou plusieurs GameObjects, incluant leurs composants et propriétés, que l'on sauvegarde pour en faciliter la réutilisation et la gestion à travers un projet.

8.3 JOURNAL DE TRAVAIL

Le journal de travail entier est dans un fichier Excel en annexe pour en avoir une meilleure visibilité et éviter de surcharger ce document.

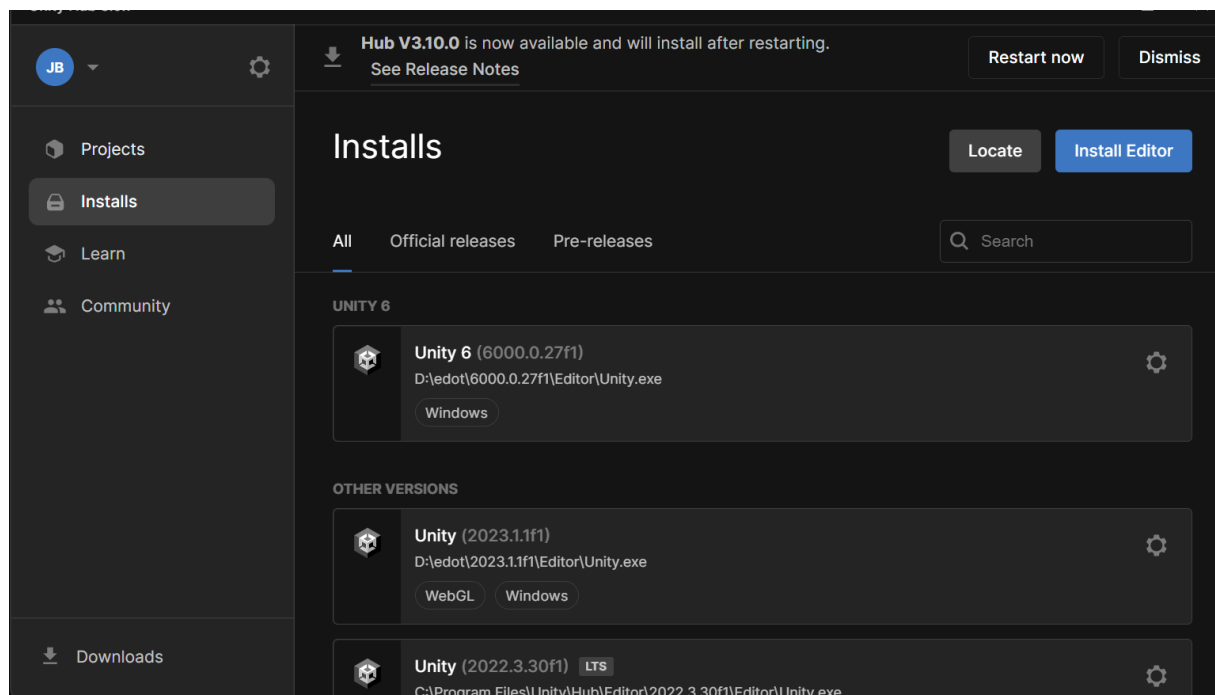
Le fichier en version PDF sera aussi rendu lors du rendu final.

8.4 PROCÉDURE D'INSTALLATION

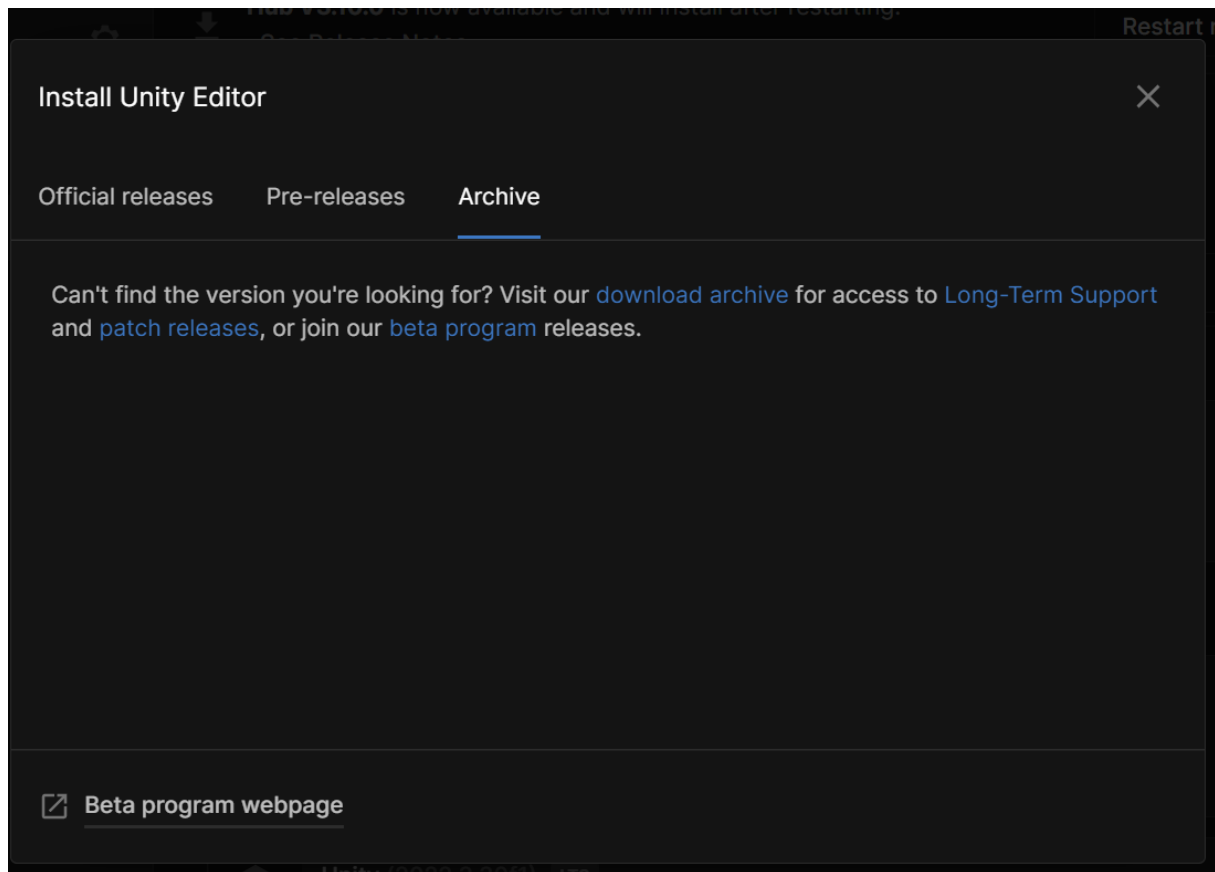
8.4.1 INSTALLATION D'UNITY

Unity utilise sa propre plateforme d'installation, qu'ils ont nommée « Unity Hub ». Vous pouvez la télécharger ici : <https://unity.com/download>.

Après avoir installé Unity Hub, il faut installer la bonne version pour le projet.



Pour ça, allez dans Installs, et cliquez sur install Editor.



Allez ensuite dans archive et cliquer sur le lien Download Archive . Cela vous redirigera vers le site de Unity, qui contient la version utilisée par le projet.

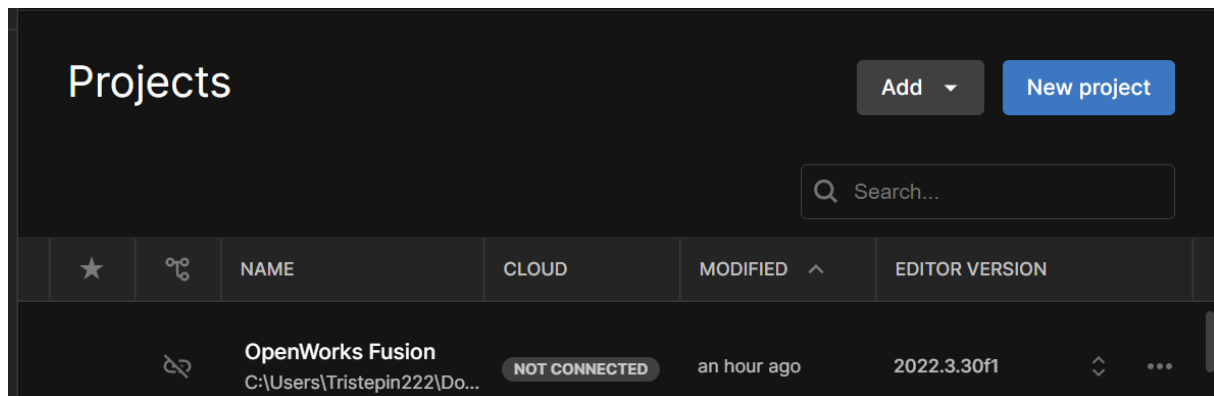
To learn more about the release versions, visit [Unity Releases](#).

Unity 6 2023 **2022** 2021 2020 2019 2018 2017 Unity 5

All versions **LTS (Default)** Tech Stream

Version	Release date	Release notes	Hub installation	Downloads
2022.3.54f1	4 déc. 2024	Read	INSTALL →	See all
2022.3.53f1	19 nov. 2024	Read	INSTALL →	See all
2022.3.52f1	6 nov. 2024	Read	INSTALL →	See all
2022.3.51f1	22 oct. 2024	Read	INSTALL →	See all
2022.3.50f1	9 oct. 2024	Read	INSTALL →	See all

Assurez-vous que vous êtes bien sous l'onglet "2022" et "LTS", vous trouverez la version 2022.3.30f1. Cliquez sur "Install" et suivez la procédure d'installation, sans ajouter de modules. L'installation peut prendre du temps, je vous conseille donc d'installer VSCode en même temps. Dès que Unity est installé, vous pouvez ajouter le projet à Unity Hub.



Vous trouverez l'option pour ajouter un projet depuis votre disque sous "Add".

Une fois que le projet est ajouté, vous pouvez lancer le projet.

8.4.1.1 INSTALLATION DES DIVERS LIBRAIRIES

Unity téléchargera les bibliothèques nécessaires automatiquement, et pour QuaD, ModTool et ModIO, elles sont directement incluses dans le projet.

8.4.2 INSTALLATION DE VSCODE

Pour installer VSCode, allez sur ce lien : <https://code.visualstudio.com/download> et installer l'extension PlantUml : <https://marketplace.visualstudio.com/items?itemName=jebbs.plantuml>.

8.4.3 INSTALLATION DE STEAM

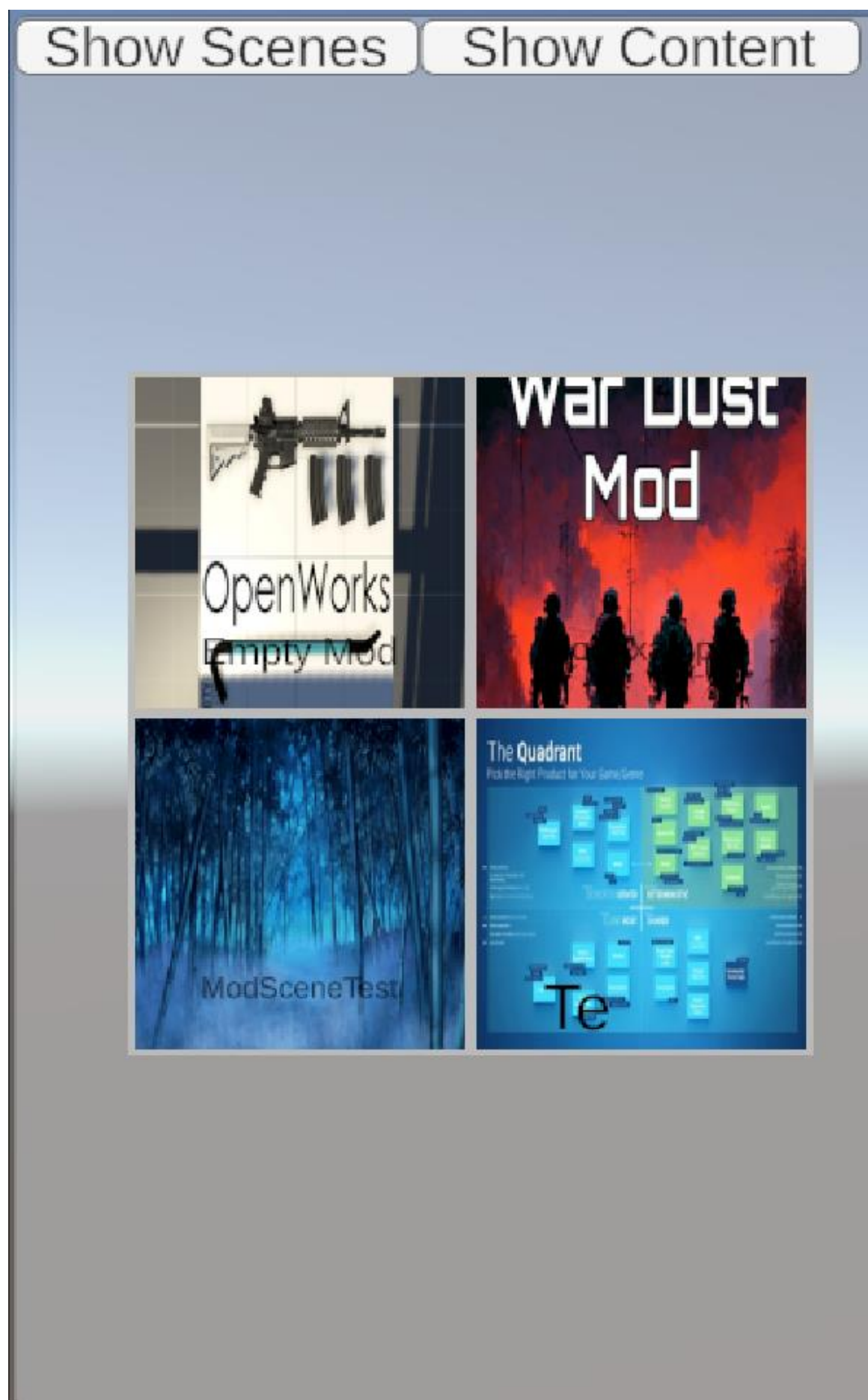
Steam est nécessaire pour le projet. Vous pouvez l'installer ici : <https://store.steampowered.com/>.

Connectez-vous ou créez un compte. Assurez-vous que Steam est bien ouvert lorsque vous essayez de tester le projet.

8.5 MANUEL D'UTILISATION

8.5.1 SAMPLE SCENE

Le projet inclut une scène d'exemple qui montre une liste de contenu prêt à être téléchargé. Vous pouvez cliquer sur les images pour télécharger le contenu additionnel.



Vous pouvez aussi cliquer sur les boutons en haut pour changer le type de contenu additionnel.

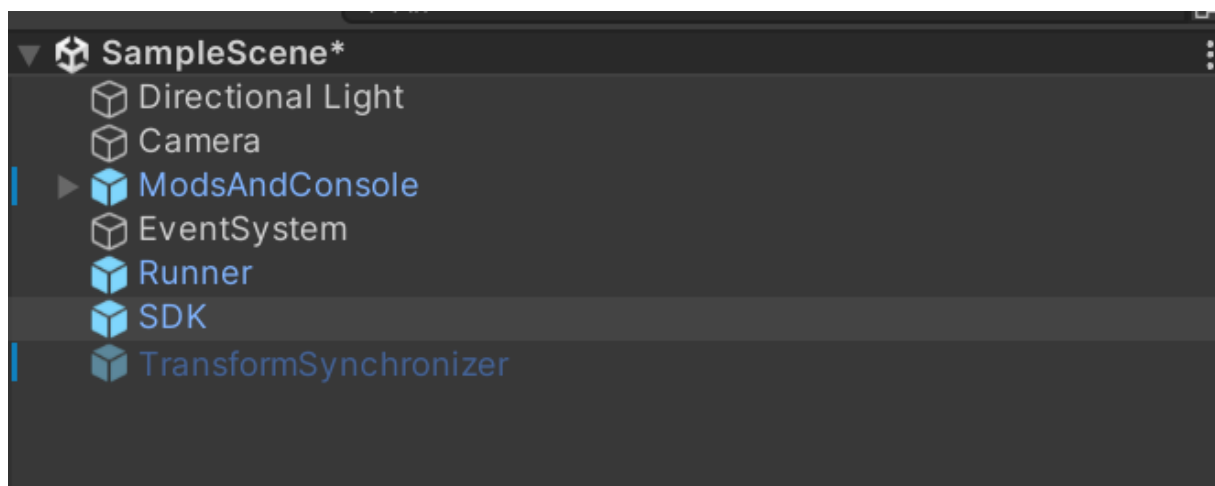
8.5.2 TESTS

Les tests se trouvent dans la scène MockServer, sous le dossier Mock. Les tests peuvent être lancés en ayant la scène active et en appuyant sur ENTER.

8.5.3 MISE EN PLACE D'UNE SCÈNE SIMPLE

Comme pour la scène d'exemple, je vais vous guider sur la façon de mettre en place une scène fonctionnelle qui utilise ce projet.

8.5.3.1 STRUCTURE



Le minimum requis pour une scène est d'avoir le prefab SDK et Runner dans la scène. Ces prefabs sont disponibles sous OpenWorks/Prefabs.

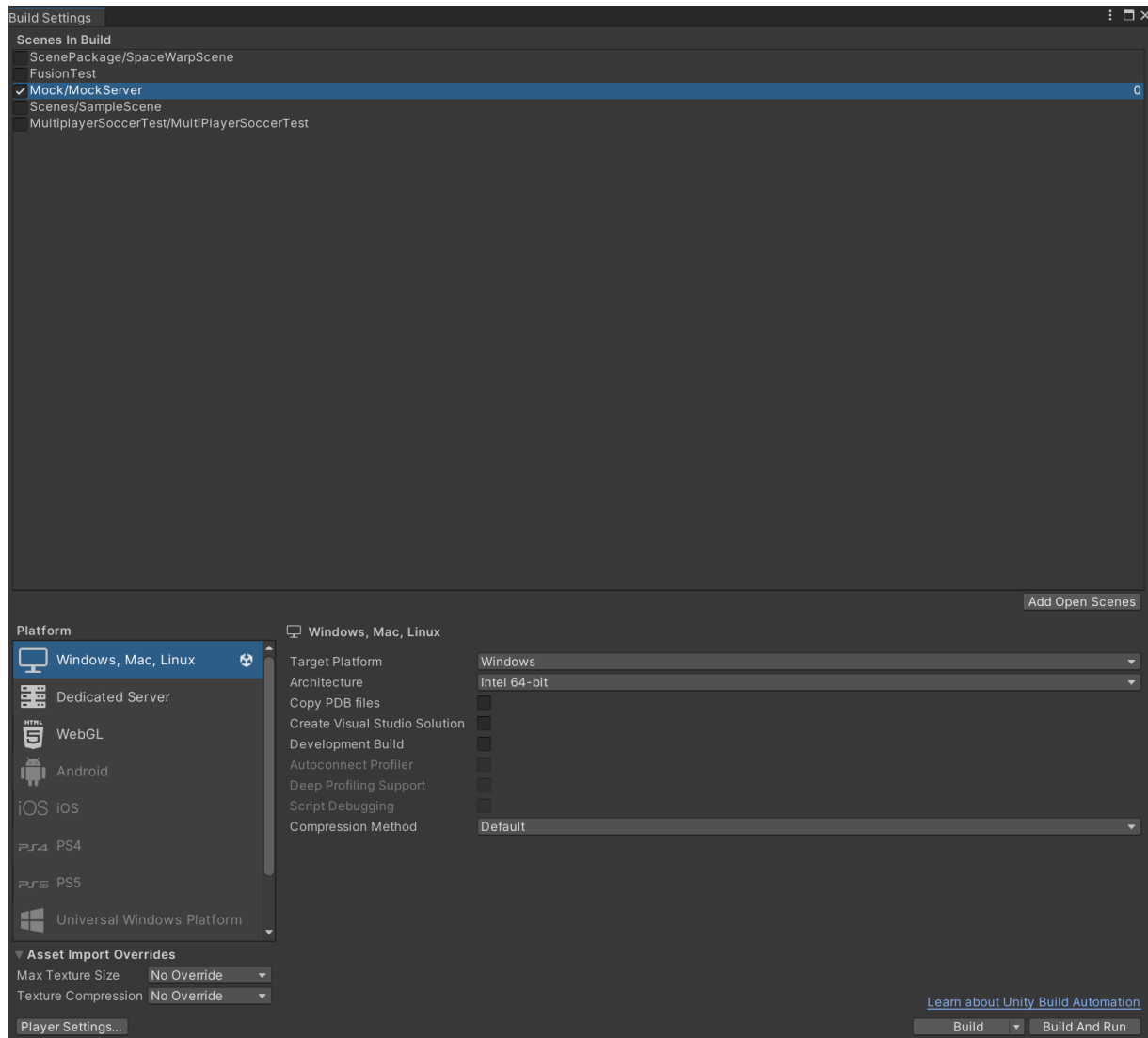
Ajoutez simplement ces prefabs dans la scène. Ils sont déjà correctement configurés.

8.5.3.2 LISTE DE CONTENU ADDITIONNEL

Si vous voulez également avoir accès à la liste de contenu additionnel, vous pouvez ajouter le prefab ModsAndConsole à la scène. Ce prefab est également configuré correctement, donc il n'y a pas besoin de modifier des valeurs.

8.5.4 BUILD

Avec Unity, vous pouvez générer le projet. Cela peut prendre un peu de temps, alors assurez-vous que la scène MockServer est cochée avant de cliquer sur Build.



Cliquez sur Build, puis choisissez un emplacement sur votre disque pour sauvegarder le projet généré.

Ensuite, copiez le fichier steam_appid.txt, situé à la racine du projet, et collez-le à côté de l'exécutable ASW.exe.

Il se peut que le build échoue, vous pouvez simplement refaire la même manipulation.

8.5.5 CONNECTION ENTRE CLIENT ET SERVEUR

Vous pouvez, soit par le build, ou Unity, lancer un serveur, si vous êtes sur Unity, ouvrez la scène MockServer, attendez quelques secondes et appuyez sur ENTER.

Pour le client, ouvrez ASW.exe, et automatiquement vous êtes connectez en tant que client, et vous pouvez voir les mêmes choses que sur le serveur, transform synchronisé, effets, etc...

8.6 SUPPORTS D'ARCHIVAGE DU PROJET

Un lien sera disponible pour le téléchargement du dossier complet.

Le dossier contient plusieurs dossiers, voici le contenu de chacun :

- Projet Unity : l'entièreté du projet Unity, sauf les libraires
- Schémas
- Journal de travail
- Rapport de travail de diplôme

8.7 TABLE DES ILLUSTRATIONS

1 Exemple d'un petit projet avec UGC	17
2 Liste de contenu sur UGC	18
3 Politique des prix de Unity pour UGC	19
4 Présentation de Workshop sur leur documentation	19
5 Page d'accueil de ModIO.....	20