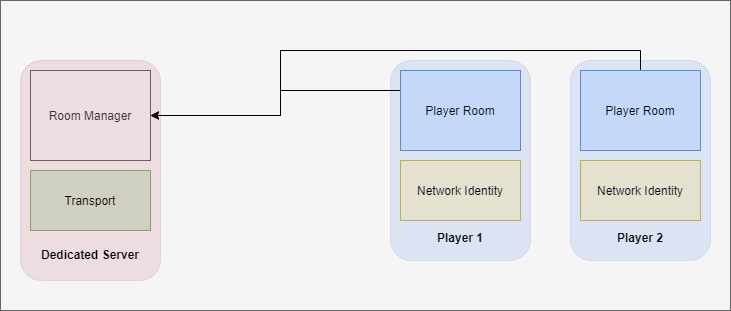
Document d’analyse Networking - Snow

# Mirror

## Présentation

Mirror est un plugin pour Unity3D qui permet de gérer la partie réseau d’un jeu. Il permet d’avoir dans un seul code la gestion des clients et du serveur, ainsi que des classes de base à partir desquelles on peut créer ses propres classes pour avoir un comportement personnalisé. Il permet de gérer à la fois un mode P2P ou un mode Serveur Dédié.



## Fonctionnement

Le Room Manager sert à gérer

## Room

### Room Player

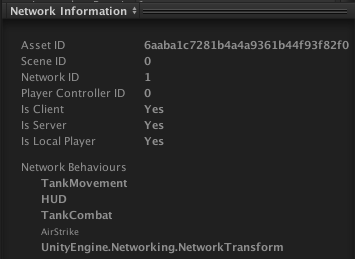
Le Room Player permet de gérer les informations du joueur de manière globale (pseudo, stats, xp…). Il se crée lors de la connexion avec le Room Manager.

### Room Manager

Le Room Manager gère l’ensemble de la Room. Tout comme le Room Player il fonctionne sur des boucles d’appel (OnServerSceneChange) qui permettent la personnalisation des comportements des classes de base de Mirror. Il gère les Room Player et communique avec eux. Il permet de générer les Room Player et Player à partir de prefabs. Il permet également de gérer les changements de scènes automatiquement.

## Synchro

### Network Identity

Le component Network Identity permet d’identifier les GameObjects sur le réseau. Il gère leur identifiant réseau unique, netId, qui permet au serveur de prendre connaissance de ces objets et de pouvoir y accéder ainsi que leurs autres propriétés réseau. Le component Network Identity est utilisé de 2 manières différentes sur le réseau, lors de l’instanciation de GameObject en ligne ou bien sur la création de GameObject à partir d’une scène.

Dans Mirror le serveur est par défaut autoritaire, c’est-à-dire que c’est à lui d’instancier les objets sur le réseau. Pour instancier un objet sur le serveur, celui-ci a besoin d’accéder au Network Identity de cet objet pour lui attribuer son netId. La méthode NetworkServer.Spawn permet d’instancier un objet sur le serveur et de le rendre visible sur l’ensemble des clients.

Il est également possible de gérer l’instanciation des objets réseaux depuis les scènes. Lorsque des GameOjects avec un NetworkIdentity sont présents sur une scène sauvegardée ils sont désactivés lors du lancement du serveur. Lorsqu’un client se connecte au serveur celui-ci lui renvoi les informations concernant uniquement les objets activés au client. Ainsi les problèmes de synchronisation des objets réseaux lors de l’apparition d’un joueur sont évités.

## Attributs

Mirror met à disposition des attributs à ajouter sous la forme de tag au-dessus des méthodes. Ces attributs permettent de définir des comportements particuliers pour ces méthodes entre les différents clients et serveur.

### [ServerCallBack] - [Server]

Les attributs [ServerCallBack] et [Server] permettent de bloquer l’utilisation de la méthode concernée si elle est appelée sur un client. [Server]enverra un warning si un client tente d’y accéder alors que [ServerCallBack] ne renverra rien. Ces tags permettent de gérer des comportements serveur.

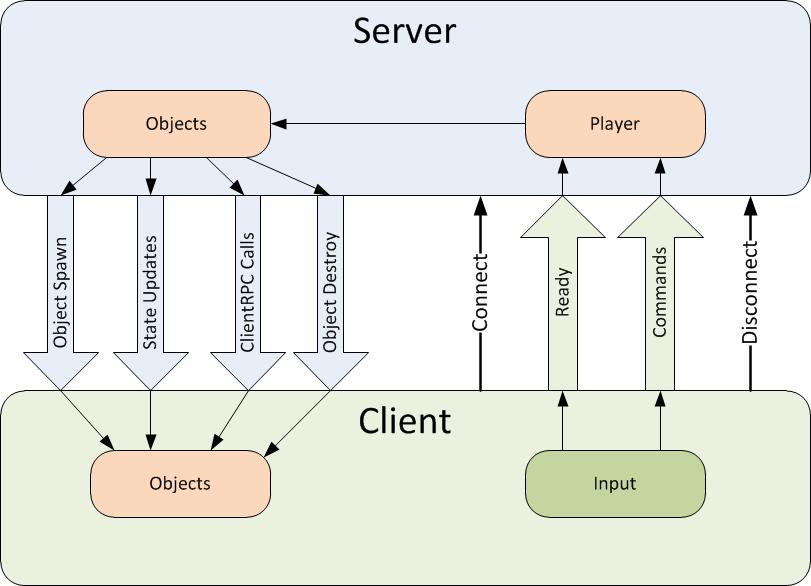
[Command]

Cet attribut permet d’appeler depuis un client une méthode sur le serveur. Il permet de gérer les actions qui doivent avoir lieu sur le serveur et non pas sur le client (instancier un objet serveur par exemple). La méthode qui possède l’attribut [Command] doit commencer par Cmd et ne pas être Static pour être reconnue en tant que commande.

[ClientRpc] - [TargetRpc]

L’attribut [ClientRpc] ajouté à une méthode permet à un objet présent sur le serveur d’appeler une méthode sur l’objet client à travers un Remote Procedure Call. Il permet de gérer des comportements qui doivent avoir lieu sur les clients et pas sur le serveur (animations)

L’attribut [TargetRpc] fonctionne de la même manière que le [ClientRpc] à la différence qu’il permet depuis d’atteindre un objet sur un client précis à partir de sa connexion.

[SyncVar]: ce tag permet, sur des variables, de définir qu’elles doivent être partagées sur toutes les instances de l’objet et donc être mises à jour régulièrement.

## Gestion des scènes

Les scènes sont gérées par le Room Manager qui prend les scènes en entrée comme des prefabs. Il se compose d’une RoomScene qui est la scène de room, d’une OfflineScene qui correspond à la scène de démarrage du client avant de se connecter au serveur et de la GameplayScene qui est la scène dans laquelle se déroule le jeu. Le Game Manager permet de changer de scènes de manière dynamique entre les 3 trois scènes qu’il gère.

On peut également changer de scène en accédant au Server Manager qui possède une méthode prévue à cet effet.

## Transport

Mirror permet de choisir le transport sur les couches de bas niveau. Ils s’intègrent par des component indépendants attachés au Network Manager. Mirror utilise par défaut Telepathy qui communique en TCP.

## Intégration

L’intégration des différents modules avec Mirror est bien supportée pour le New Input System ainsi que pour les animations qui sont gérées côté serveur puis mise à jour sur les clients à l’aide de RPC