# 2

# Découverte d'UNET

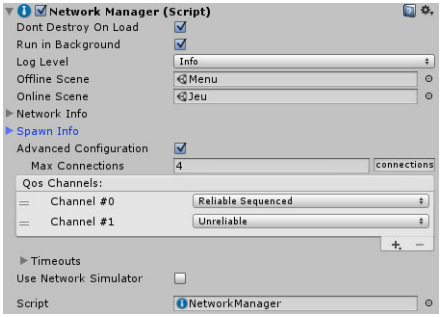
Avant l'arrivée d'UNET, les développeurs Unity devaient tout coder de A à Z lorsqu'il s'agissait de développer un jeu en réseau. Ils devaient créer des scripts pour gérer le serveur, la connexion des clients, la synchronisation des éléments sur les différents clients, etc. UNET est un système qui nous facilite grandement les choses et permet de mettre en place très rapidement un jeu multijoueur. La liaison client/serveur est automatique, de même que la synchronisation des éléments, etc.

Unity a voulu frapper fort en proposant un outil très puissant permettant à tous les développeurs d'inclure un mode en ligne dans leurs jeux. Bien sûr, il y aura quand même du code à écrire mais ce travail sera beaucoup moins laborieux qu'auparavant. Dans ce chapitre, je vais vous présenter quelques-unes de ces fonctionnalités. Nous compléterons cette liste au fur et à mesure des chapitres.

## 2.1. Le Network Manager

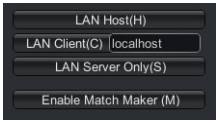
Bien que tous les composants d'UNET soient importants, le Network Manager constitue l'élément central d'un jeu multijoueur. C'est le gestionnaire de réseau qui va gérer toutes les connexions clients/host et tous les paramètres du serveur. Il se charge, entre autres, de connecter les clients au serveur via un port et une adresse IP, de créer les prefabs des joueurs lorsque ceux-ci se connectent et de gérer les déconnexions. Lorsqu'un objet est instancié, il permettra son instanciation sur tous les clients connectés au jeu. Voilà à quoi ressemble ce composant :

*Figure 2.1 : Aperçu du Network Manager*




Nous utiliserons également un autre composant associé, il s'agit du Network Manager HUD, en d'autres mots l'interface utilisateur du Network Manager. Ce petit composant très simple permet de faire apparaître à l'écran un menu élémentaire pour créer un serveur ou se connecter à un serveur :

*Figure 2.2 : Le Network Manager HUD*




Nous y recourrons dans ce livre pour faire des tests rapides et vérifier que la mise en réseau se fait bien.

***Attention*** *› Ce composant ne doit pas être utilisé en production, il ne sert qu'à tester votre projet. Pour la version finale de notre jeu, nous développerons notre propre menu avec les seules fonctions qui nous intéressent et une interface personnalisée.*

## 2.2. Le Network Identity

Là aussi, il s'agit d'un composant extrêmement important car c'est lui qui va donner une identité, donc une existence, aux objets qui doivent être visibles sur tout le réseau comme par exemple les personnages joueurs ou les projectiles. Pour qu'un objet puisse être visible ou instancié sur le réseau, il doit posséder ce composant, qui lui attribuera un numéro unique.

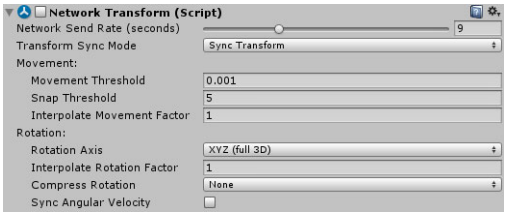
Au-delà de ces propriétés essentielles, le Network Identity offre des possibilités très puissantes comme par exemple récupérer l'identifiant unique d'un objet, savoir si le joueur en cours est client ou serveur, connaître les permissions selon le principe d'autorité ou encore de savoir quels objets peuvent voir tel ou tel autre objet. Nous examinerons cela plus en détails au cours des chapitres suivants car nous aurons besoin de savoir, par exemple, qui a tiré le projectile et qui a été touché par ce dernier pour faire le calcul des points.

## 2.3. Le Network Transform

Comme son homologue le Transform, le Network Transform permet de gérer des informations relatives à la position, la taille ou la rotation d'un objet de jeu. Cependant, le Network Transform a la capacité de synchroniser les mouvements d'un GameObject à travers le réseau, c'est-à-dire qu'il est capable de transmettre la position d'un objet à tous les autres clients connectés au réseau.

Les propriétés de ce composant permettent de gérer le mode de synchronisation ou encore la quantité d'informations envoyées sur le réseau afin d'alléger le flux de données. Voilà à quoi il ressemble :

***Figure 2.3 :*** *Le Network Transform*




***Note ›*** *Bien que ce composant soit configurable, il n'est pas suffisamment optimisé dans certains cas. Nous verrons dans la suite de ce livre comment en programmer nous-mêmes les fonctionnalités tout en optimisant la synchronisation de la position des objets afin de réduire les ralentissements et les bugs.*

## 2.4. Quelques propriétés bien utiles

Outre les composants précités, UNET apporte une panoplie de nouvelles fonctions très utiles pour programmer des jeux en réseau. Nous aurons l'occasion de les mettre en pratique mais voici déjà une petite liste des principales d'entre elles :

**`isServer`**

Facile à comprendre, cette fonction permet de tester si l'objet sur lequel est attaché le script tourne sur le serveur.

**`isClient`**

Il s'agit de la fonction inverse et celle-ci permet de savoir si l'objet tourne sur un client.

**`isLocalPlayer`**

Cette fonction, très utile, permet de savoir si l'objet est un objet du joueur local.

**`netId`**

Correspond à l'identifiant unique d'un objet sur le réseau.

**`Spawn`**

C'est la fonction permettant d'instancier un objet sur le réseau afin qu'il soit visible par tous les clients.

UNET apporte également de nouvelles façons de déclarer des variables par exemple le mot clé `[SyncVar]` permet de synchroniser automatiquement la valeur d'une variable sur le réseau. Le mot clé `hook`, quant à lui, permet d'exécuter une fonction lorsqu'une variable en `[SyncVar]` est modifiée. Cela permet de faire un traitement spécial si besoin. D'autre part, certains mots clés permettent d'exécuter des fonctions d'une façon bien précise sur le réseau. Par exemple :

**`[Server]`**

Une fonction précédée du mot `[Server]` ne s'exécutera que sur le serveur.

**[Client]**

Une fonction précédée du mot `[Client]` ne s'exécutera que sur le client.

**[Command]**

Permet à un client de demander au serveur d'exécuter une fonction.

**[ClientRpc]**

Permet d'exécuter une fonction sur tous les clients.

La liste donnée ci-dessus n'est pas complète, les propriétés de UNET étant nombreuses, mais elle présente les fonctionnalités les plus couramment utilisées. Avec ces dernières vous serez en mesure de coder 90 % des fonctionnalités liées au réseau. Nous verrons dans la suite de ce livre d'autres propriétés dont nous aurons besoin pour optimiser notre jeu.

Dans ce chapitre, vous avez pris connaissance des principaux composants et fonctions de UNET, notamment :

- le Network Manager ;
- le Network Identity ;
- le Network Transform ;
- quelques fonctions permettant de localiser les objets sur le réseau ;
- de nouveaux attributs proposés par UNET.

# 3

# Mise en réseau

Notre travail nous a permis de créer un prototype de jeu très simple mais fonctionnel. En effet, nous pouvons nous déplacer dans le niveau et nous pouvons tirer avec notre arme. Même si notre jeu peut être conservé ainsi, nous allons ajouter une couche réseau afin de le rendre beaucoup plus intéressant.

Nous allons voir maintenant comment ajouter un mode multijoueur en ligne à un jeu déjà existant. Nous partirons du jeu 2D que nous avons développé dans le module III. Concevoir un jeu 2D. Si vous êtes déjà à l'aise avec la conception de ce type de jeu, vous n'avez pas besoin de le lire pour suivre ce qui suit. Il vous suffit de télécharger les sources de l'exemple.

Nous allons avoir besoin d'ajouter de nouveaux composants qui nous permettront de synchroniser en temps réel toutes les informations (position des personnages, des projectiles...) sur le réseau. Comme nous l'avons vu au chapitre Notions fondamentales du réseau, nous allons mettre en place un système d'hôte. Le premier joueur à lancer le jeu aura le rôle de serveur. Ce joueur pourra donner son IP à ses adversaires afin que ceux-ci puissent se connecter au jeu.

***Note ›*** *Dans notre cas, le joueur qui aura le rôle de serveur aura besoin de récupérer son IP et de la donner à ses adversaires. Cette étape n'est pas automatique. Pour automatiser ce processus, il faut souscrire à un service payant auprès de Unity. Dans la suite de ce livre, vous apprendrez à communiquer avec un serveur. Vous serez en mesure de créer un petit système qui récupèrera l'IP de l'hôte, qui stockera cette IP en ligne afin qu'automatiquement les autres joueurs puissent la récupérer. Cela vous permettra d'automatiser cette étape de connexion à l'hôte.*

Dans le cadre de notre exemple, nous limiterons le nombre de joueurs maximum à quatre mais cela pourra être adapté par la suite. Une fois connectés, les joueurs pourront se tirer dessus.

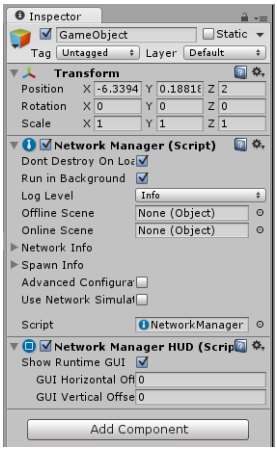
Nous allons utiliser un Network Manager pour gérer la création du serveur et la connexion des clients. Pour le mettre en place, nous avons besoin d'une *online scene*, le jeu, et d'une *offline scene*, le menu principal. Avant toute chose, nous devons donc créer un menu.

## 3.1. Création du menu principal

Dans un premier temps, nous allons créer un menu extrêmement simple qui ne nous servira qu'à lancer le jeu. Créez une nouvelle scène (Ctrl+N) et enregistrez-la (Ctrl+S) sous le nom Menu. Vous pouvez laisser cette scène vide ou y placer une image pour la décorer.

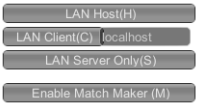
Une fois la scène prête, vous devez créer un objet vide (GAMEOBJECT/CREATEEMPTY) et lui ajouter le composant Network Manager (COMPONENT/NETWORK/NETWORKMANAGER). Ajoutez également le Network Manager HUD (COMPONENT/NETWORK/NETWORKMANAGERHUD) pour générer une interface utilisateur minimaliste.

***Figure 3.1 :*** *Les composants de UNET*




Sans rien faire de plus, ces deux composants sont suffisants pour gérer une simple mise en réseau. Si vous lancez le jeu, vous devriez avoir l'interface visuelle suivante sur votre menu :

***Figure 3.2 :*** *Le Network Manager HUD*



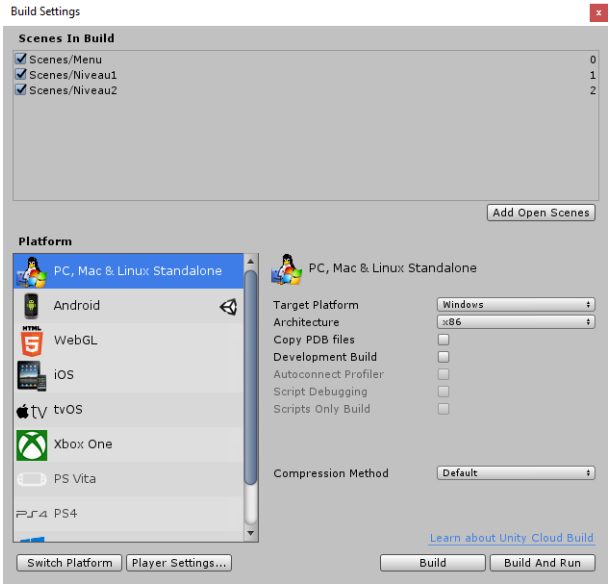

Le premier bouton LAN HOST permet de créer un serveur. Le deuxième LAN CLIENT permet de se connecter à un serveur en spécifiant une IP dans le champ de texte à droite du

bouton. Le troisième permet de créer un serveur sans être client (utile lorsque vous voulez créer une partie sur un ordinateur sans que l'on puisse jouer à partir de cet ordinateur). Le dernier, enfin, permet de créer et de rejoindre des parties en ligne en passant par les serveurs de Unity. Nous n'utiliserons pas cette fonctionnalité qui est payante.

## 3.2. Configuration du Network Manager

Pour que le Network Manager puisse utiliser les scènes du jeu, elles doivent être déclarées dans les BUILD SETTINGS. Allez donc dans FILE/BUILD SETTINGS et ajoutez-y les quelques scènes qui constituent votre jeu.

***Figure 3.3 :*** *Ajout des scènes au BUILD SETTINGS*




Une fois les scènes dans les BUILD SETTINGS, vous pouvez les ajouter au Network Manager pour indiquer laquelle est la *online scene* et laquelle est la *offline scene*. Pour cela, il vous suffit de les glisser/déposer dans la variable de l'inspector. Pour notre part, nous choisissons le menu comme scène *offline* et le niveau 1 comme scène *online*.

Vous devez aussi spécifier quel est le prefab du personnage joueur qui doit être instancié lors de la connexion d'un joueur. Ce prefab doit être placé dans la variable `Player`