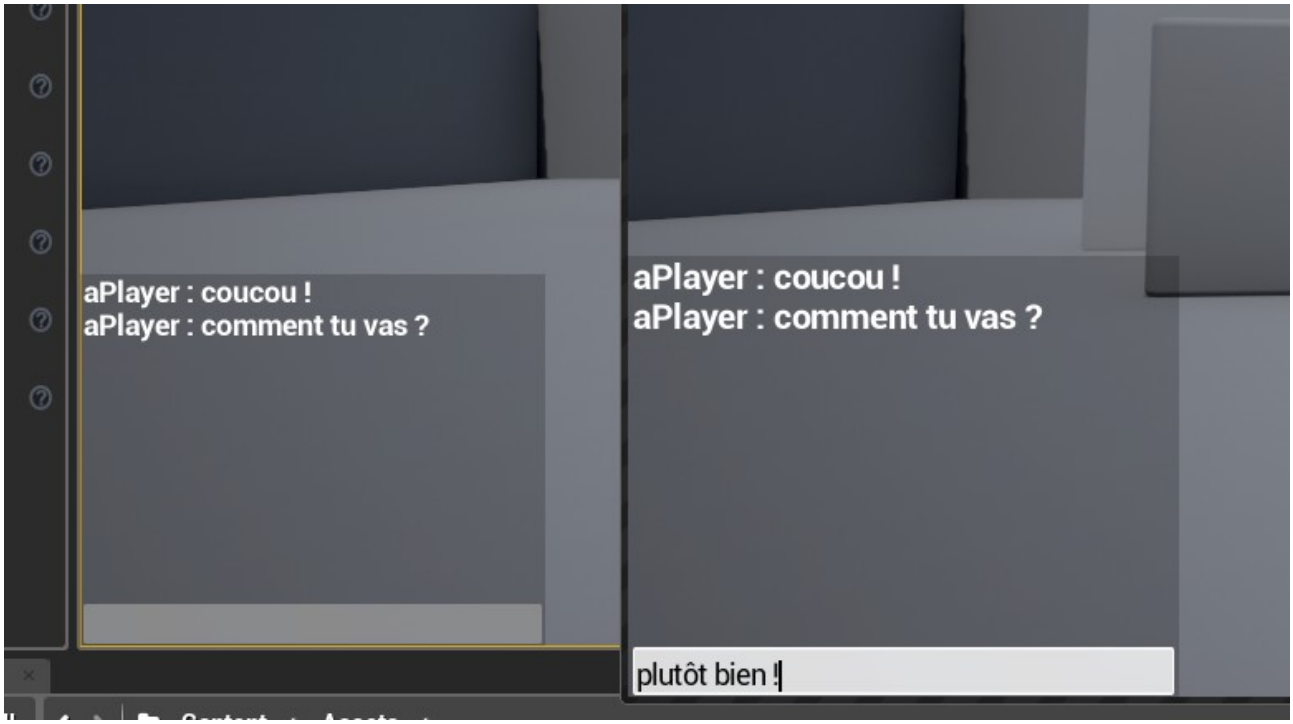
Introduction

(NB : J'espère que ce tutoriel vous sera utile et qu'il sera compréhensible par tous. Si vous avez des questions n'hésitez pas à m'envoyer un message sur Facebook « Antoine Gargasson » ou par mail antoine.gargasson@gmail.com)

(NB2 : Ce tuto est réalisé sur la version 4.10.4 de UE4)

Dans ce tutoriel, nous allons créer un chat en Blueprint pour du multijoueur.

Voici le résultat final :



Deux joueurs ou plus peuvent s'envoyer des messages par le réseau, on peut voir ici que les deux joueurs se répondent et ont une vue sur le joueur qui parle. (Dans l'exemple les deux joueurs ont le même nom.)

Préparation

Nous allons utiliser le réseau pour réaliser le chat. Il est important de comprendre comment le réseau fonctionne sur UE4 afin de bien l'utiliser. Pour ce tutoriel vous n'aurez pas besoin de connaissances particulières, juste de la bonne version d'Ue4 et de votre cerveau !

Unreal gère le réseau de deux façons différentes : Vous pouvez choisir de faire d'un de vos joueurs le serveur et les autres joueurs seront des clients. Vous pouvez sinon choisir de faire un serveur dédié qui va gérer votre jeu et que tous vos joueurs soient des clients. C'est ce deuxième choix que nous allons prendre.

Il faut savoir que vous allez coder votre jeu différemment en fonction de la façon de gérer le réseau que vous aurez choisie.

Quand vous souhaitez appeler une fonction et que vous faites du réseau, vous devez vous poser la question suivante : " Cette fonction doit-elle être exécuter côté serveur, côté client ou en local ? "

En effet, certaines fonctions doivent être appelées sur le serveur pour mettre à jour des variables (fonctions "server"), d'autres doivent être envoyées à tous les clients pour qu'ils se mettent à jour (fonction "multicast" et "owner") et enfin certaines fonctions peuvent rester en local (fonction "non répliquée").

Le mot **répliqué** va souvent revenir dans ce tuto, il signifie que votre fonction ou variable est envoyé et utilisable en réseau.

Pour ce tuto, j'ai choisi de prendre un seul type de personnage, vous pouvez adapter ensuite ce tuto pour plusieurs types de joueurs. Je pars d'un projet Third Person blueprint de base sans le starter content. Sachez que par défaut les Pawn et Characters sont répliqués sur le réseau. Ainsi si vous lancez votre projet 3^{e} personne une fois créé, vous pourrez déjà voir vos personnages bouger.

Pour lancer votre jeu en multijoueur c'est très simple, cliquez sur la petite flèche à côté du "Play" en haut à droite et changez les options suivante :

Number of Players : 2
Run Dedicated Server : coché

Si vous faites Play maintenant, Unreal va vous ouvrir une fenêtre et vous allez pouvoir bouger vos deux personnages.

Le visuel

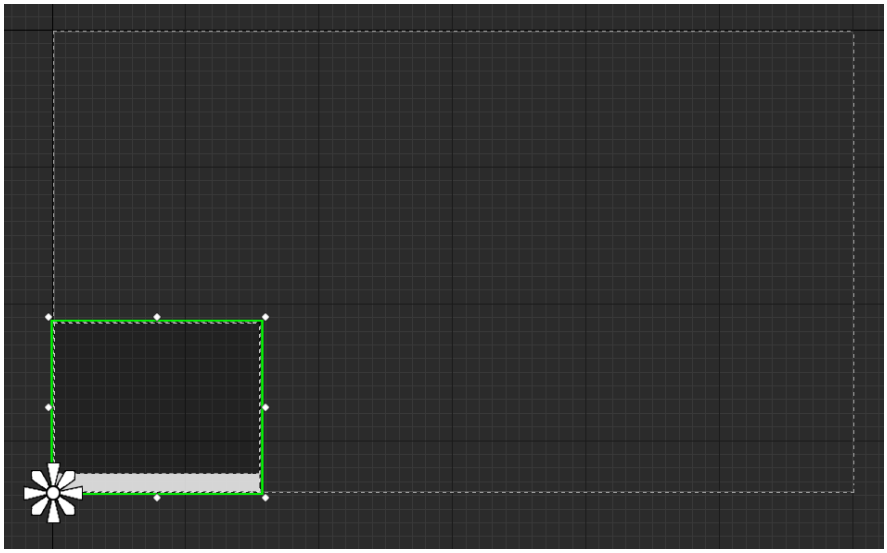
Pour commencer nous allons créer un peu **d'UI** (user interface) à l'aide de **l'UMG** (Unreal Motion Graphics) proposé par Unreal.

Créez un dossier **Assets** dans le **Content** et dans ce dossier créez **trois Widget Blueprints** (Clique droit → User Interface). Nommez les **UMG_HUD**, **UMG_ChatBox** et **UMG_TextChatBox**.

Commencez par **ouvrir UMG_TextChatBox**. Il va être le plus simple à faire. **Supprimez** le **Canvas Panel** dans la **Hierarchy** et ajoutez un **Text**. Ce texte va être dans la chat box et c'est ici que vous allez pouvoir gérer votre texte comme bon vous semble (taille, couleur, etc.).
Dans notre cas, on va se contenter de **cocher Auto Wrap Text** qui va faire un retour à la ligne quand on aura atteint de bord de la box. N'oubliez pas de **cocher Is Variable** dans le panel Details, une fois le texte sélectionné, afin de pouvoir modifier votre texte ultérieurement. Je l'ai également renommé **TextBox**.

Ouvrez ensuite **UMG_ChatBox**. Nous allons ici créer la chat Box en elle même. Pour commencer, **supprimez** le **Canvas Panel** pour pouvoir gérer sa taille par la suite. **Ajoutez** une **Border**. **Changez** sa **Brush Color** comme bon vous semble. J'ai choisi de la mettre en **noir** avec un **alpha** à **0,2**. Dans la **border**, mettez un **Vertical Panel** pour gérer les éléments les uns au dessus des autres. Vous pouvez alors **ajouter** une **Scroll Box** et un **Text Box** dans ce Vertical Panel.
Sur la **Scroll Box**, **changez** la **Size** en **Fill** au lieu de Auto et **cochez Is Variable**.
Sur la **Text Box**, allez dans Style, déroulez Style. Changez le **Padding Top** à **10** et la **taille** de la **Font** à **20**. **Décochez** le **Is Enabled**.
Votre Chat Box est visuellement prête !

Ouvrez ensuite **UMG_HUD**. Laissez le Canvas Panel et **ajoutez** un **UMG Chat Box** (User Created). J'ai choisi de mettre **l'ancre** en **Bas** à **Gauche**. Modifiez **l'Alignment** à **0 en x** et **1 en Y**. Pour la **taille** j'ai opté pour **500 en Size X** et **400 en Size Y**. Mettez les **positions** en **X et en Y à 0**.



Le code

Abordons la partie la plus compliquée : Le code.

Tout d'abord nous allons nous charger de transmettre le message à tous les autres joueurs.
Ensuite nous allons créer les fonctions sur la Chat Box pour entrer le texte et l'envoyer au joueur qui l'a écrit.

Commençons par préparer quelques blueprints qui vous seront utiles :

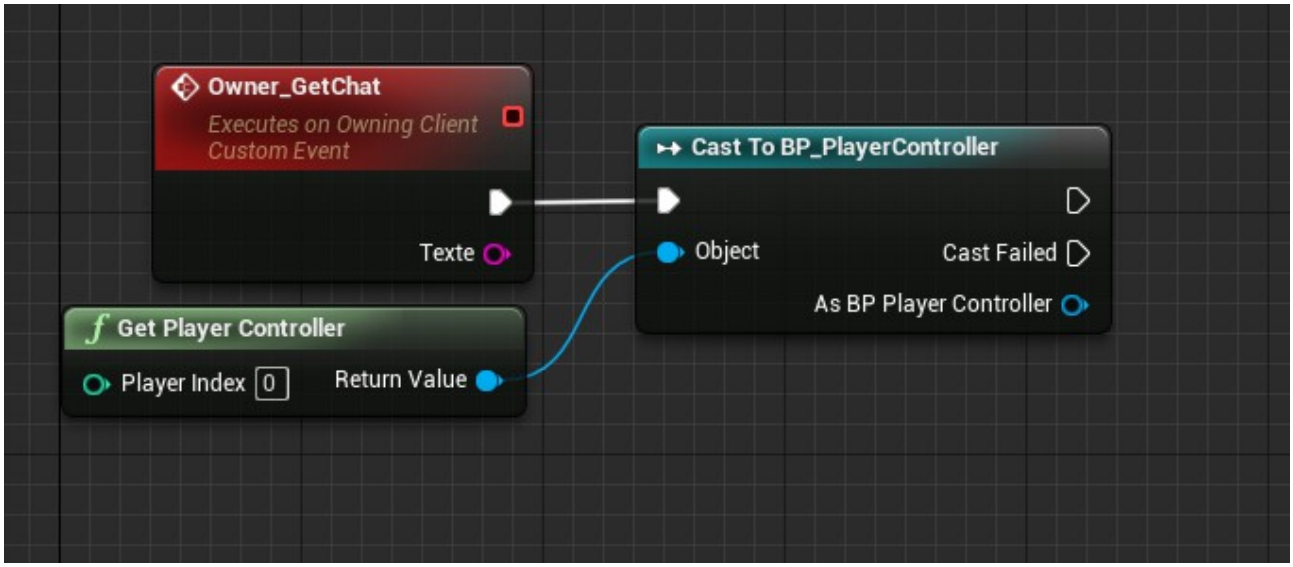
Créer un nouveau **Player Controller** (Clique droit → Blueprint → Player Controller) et **nommez** le **BP_PlayerController**. Ensuite, **créez** un nouveau **Player State** (Clique droit → Blueprint → All Classes → Player State) et **nommez** le **BP_PlayerState**.
Pour pouvoir utiliser ces deux blueprints, **créez** un nouveau **Game Mode** (Clique droit → Blueprint → Game Mode) et **appelez** le **BP_GameMode**.

Ouvrez votre **BP_GameMode**, **changez** le **Player Controller** et le **Player State** par ceux que vous venez de créer.
Rendez-vous dans Edit → Project Settings → Maps & Modes et mettez votre **BP_GameMode** dans comme **Default GameMode**.

La phase de préparation est finie, commençons le code !

Commencez par **créer** un **custom Event** dans le **BP_PlayerState**. Je l'ai appelé **Owner_GetChat**. Par convention, le met le type de fonction devant quand je fais du réseau (Owner, Server, Multicast ou rien pour le local).
Changez son **Replicates** en **Run on owning Client**. Vous pouvez le modifier dans le panel Details.
En **Input**, ajoutez un **String** et changez son nom en "Texte".

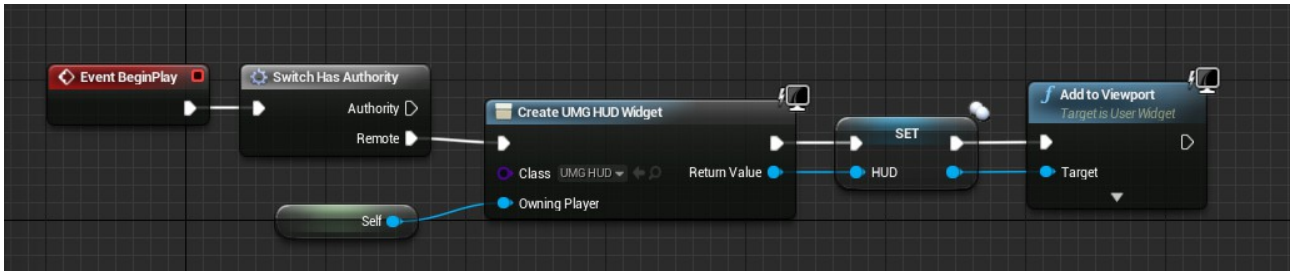
A la suite de cette fonction, ajoutez un Get Player Controller et castez le en BP_PlayerController.
Vous pouvez laisser ce blueprint de côté pour le moment.



Continuons avec le **BP_PlayerController** !

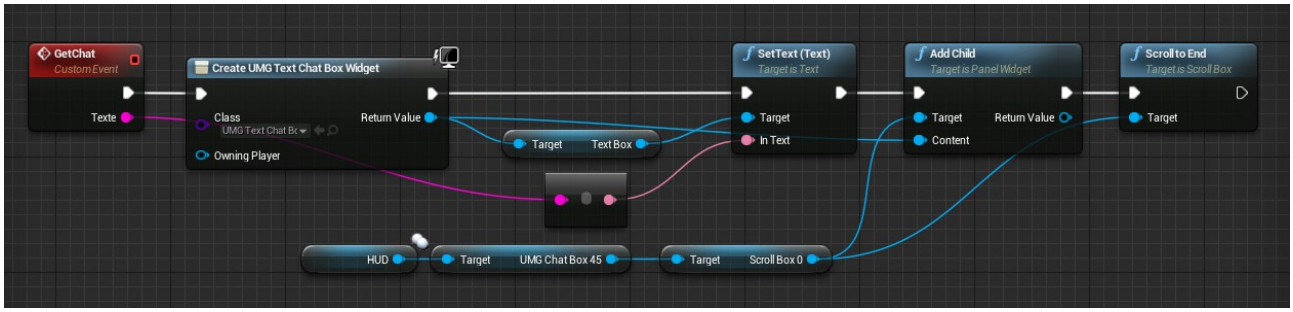
C'est ici que beaucoup de choses vont se produire.
Commençons par **ajouter** quelques **variables** :
"HUD" de type UMG_HUD, oubliez pas de le répliquer.
"InChat" de type booléen, non répliqué.

Continuons avec le **Begin Play**. **Créez** un **Switch Has Authority**. Ce node va vérifier le **rôle** du **player Controller** (Client ou Serveur). Partant du **Remote** (client) **ajoutez** un **Create Widget**. **Changez** la **class** par **UMG_HUD**, **ajoutez** un **Set HUD** à la sortie du Create Widget et enfin un **Add To Viewport**.
Vos joueur ont désormais un HUD chacun !



Créons ensuite un **Custom Event** "**GetChat**". Cette fonction va **s'exécuter en local** et ajoutera du texte reçu dans la Scroll Box. **Ajoutez** un **paramètre String** et renommez le "Texte".
Branchez un **Create Widget** de type **UMG_TextChatBox**. Exécutez ensuite un **SetText** sur ce widget et **branchez** en entrée le **Texte en paramètre** de la fonction. Faites ensuite un **Add Child** sur la **Scroll Box** avec comme **entrée** le **Widget**, cela va l'ajouter à la scroll box et finissez pas un **Scroll to end** sur la **scroll box** afin de faire défiler les messages.

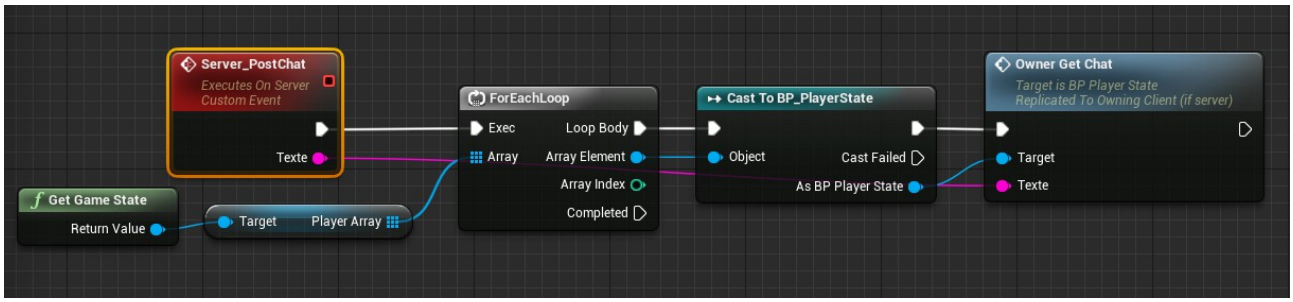
Vos messages peuvent désormais être ajoutés dans la chat box !



Nous allons maintenant passer à l'envoi sur le serveur d'une fonction. Elle va se charger d'envoyer à tous les joueurs le message.

Créez un **Custom Event**, nommez le **Server_PostChat** et mettez le en **Run on Server** pour la partie **Replicates**. **Ajoutez** un **Input** de **String**.
Récupérez le **Game State** qui est l'état du jeu courant et **récupérer** le **player Array** dessus. C'est le tableau des joueurs sur la partie (référencés par leur Player State). Nous allons exécuter un **ForEachLoop** sur le **tableau** Player Array. **Castez** ensuite le **Array Element** en **BP_PlayerState** dans le **Loop Body**. Sur ce BP_PlayerState, **utilisez** la fonction **Owner Get Chat** en oubliant pas de **brancher** le **String** en entrée.

Votre message est envoyé à tous les joueurs !

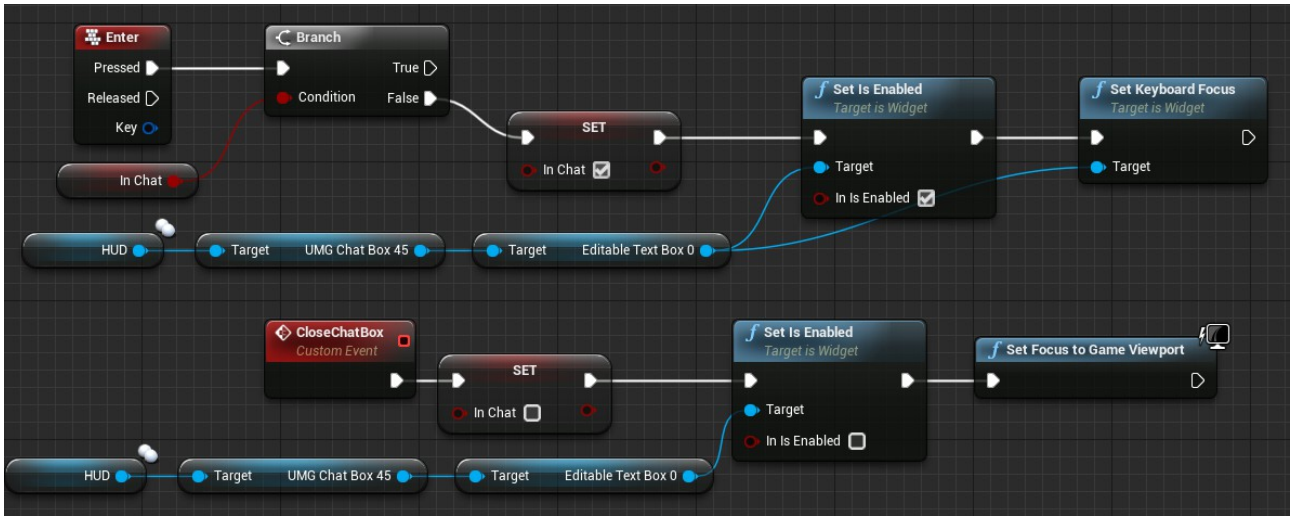


Nous allons à présent, binder la touche entrée sur l'ouverture de la text box et faire une fonction pour fermer celle-ci.

Récupérer l'input **Enter**, **créez** une **branch** avec en **entrée** la variable **In Chat**. Si c'est **False**, faites un **Set in Chat** à **true**, un **Set Is Enabled** sur la **text Box** et un **Set Keyboard Focus** sur la **Text Box**.

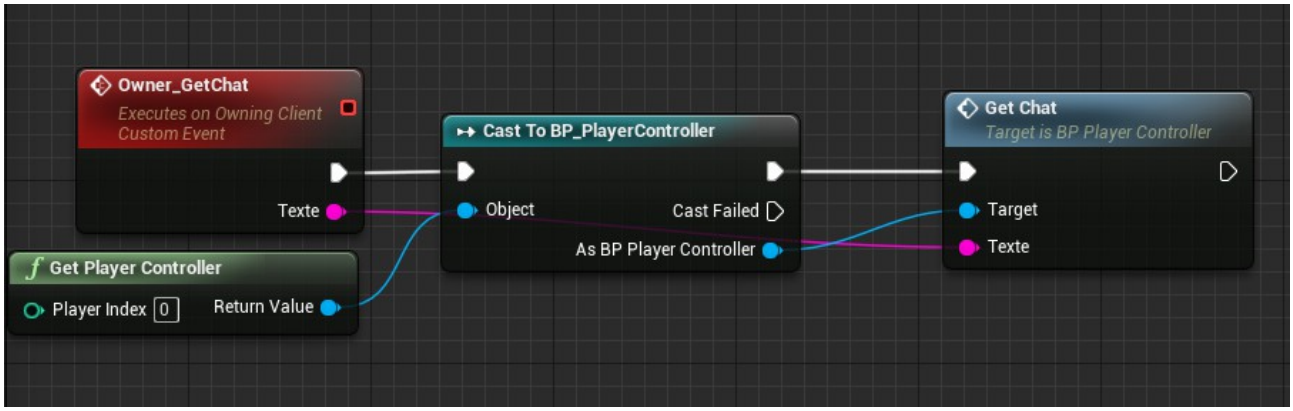
Créez un nouveau **Custom Event** "CloseChatBox". Exécutez un **Set in Chat** à **false** et un **Set Is Enabled** à **false** sur la **text box**. **Rajoutez** un **SetFocusToGameViewport** pour que votre joueur puisse bouger sans avoir besoin de cliquer à nouveau.

Vous pouvez à présent ouvrir et fermer votre fenêtre de discussion ! :D



C'est tout pour le Player Controller !

Un petit saut par le **Player State** où vous pouvez **ajouter** la fonction **Get Chat** et **relier** le texte en **entrée**.

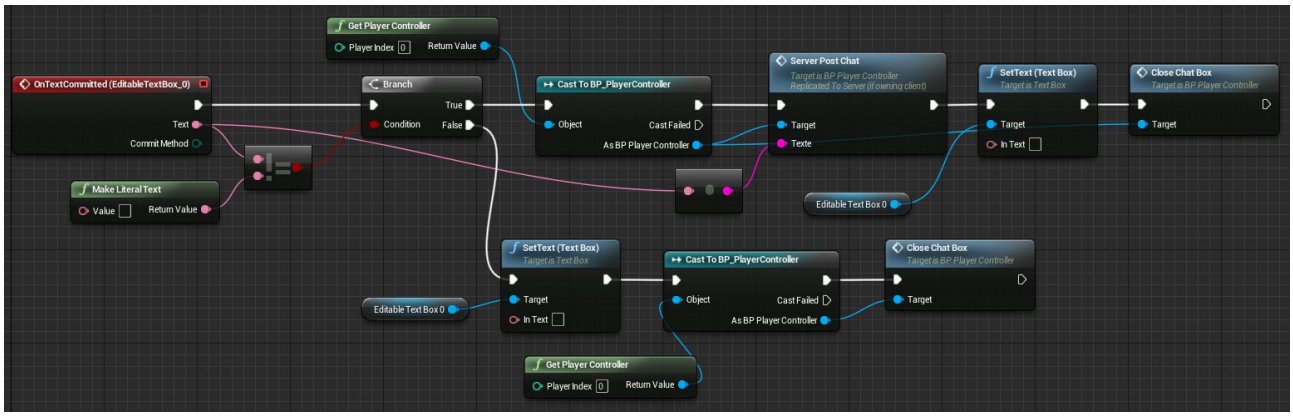


Il ne nous reste plus qu'à traiter le texte et à l'envoyer à au serveur quand on appuie sur entrée !

Ouvrez l'**UMG_ChatBox**. Sélectionnez votre **TextBox** dans la **Hiérarchy** et **cliquez** sur le bouton **+** à côté de **OnTextCommitted** dans le panel **Details**. Cette fonction s'exécute quand vous avez le focus sur le text box et que vous faites entrée.

Vérifiez que le **texte** en sortie est **différent** de **vide**.
Si c'est **true**, **récupérez** le **player controller**, **castez** le en **BP_PlayerController**, **exécuter** le **ServerPostChat** en branchant **l'entrée** et faites un **Settext** sur le **text Box** en laissant le champ **vide**. Finissez par **appeler** le **CloseChatBox** sur le **player controller**.
Si c'est **false**, faites un **set text** sur le textbox et laissez le **champ vide**, **récupérez** le **player controller**, **castez** le et **appelez** un **CloseChatBox**.

Votre texte est envoyé et réinitialisé ! :D



Vous pouvez lancer et tester votre chat. Faites entrée, tapez votre message et entrée encore. Le message devrait apparaître sur tous les chats. Sinon revérifiez que vous n'avez rien loupé comme les Replicates ou les type d'Event (server, owner, local).

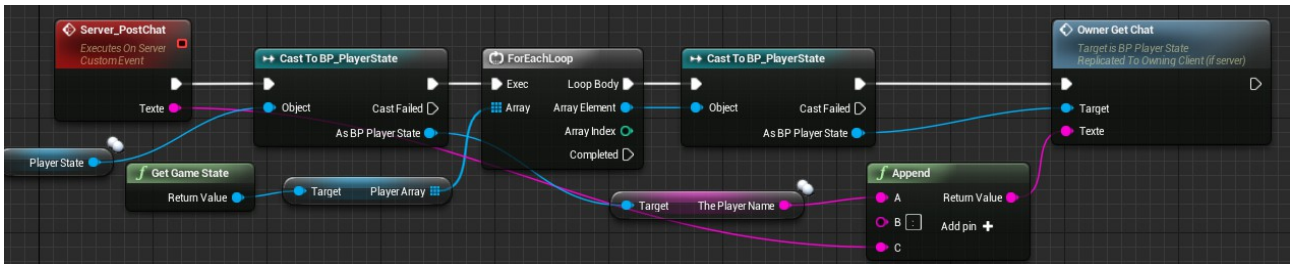
Si ça fonctionne, félicitation vous venez d'économiser 30 € (si si allez vérifier sur le store) !!! :D

Bonus

Si vous souhaitez que le **nom** du **joueur apparaisse** sur le chat, **rendez-vous** sur le **BP_PlayerState**.

Ajoutez une **variable** de type **String** et **nommez**-la **thePlayerName**. Mettez une valeur par défaut, j'ai mis "aPlayer". N'oubliez pas de la mettre en **Replicated**.

Retournez sur le **BP_PlayerController**, sur la **fonction Server_PostChat**.
Rajoutez au début de la fonction un **get player State** et **castez** le en **BP_PlayerState**. En **entrée** du **OwnerGetChat**, faites un **Append** et rajoutez un pin. En **A** mettez le **thePlayerName** récupéré sur le player state, en **B** rajoutez **" : "** et en **C branchez** le **texte** du **message**.



Si vous testez, le nom du joueur sera bien mis à côté. Les deux joueurs auront le même nom.

Je vous laisse le soin de gérer côté serveur le changement de nom via un widget blueprint ou tout autre méthode qui vous paraîtrait intéressante ;)

C'est tout pour aujourd'hui ! J'espère que ce tuto vous sera utile, qu'il vous aura permis de comprendre certaines choses concernant le réseau sur Unreal.
Je m'excuse d'avance pour les fautes qui pourraient traîner ici ou là (elles résistent les bougres!)

Pour toute éventuelles questions, remarques, détails ou demandes, n'hésitez pas à prendre contact avec moi ! (Cf : début du tuto)