

pour faire tourner un LLM qui servira à générer les dialogues de mes PNJs, au lieu d'avoir recours aux services externes et payants de Grok par exemple, quel type de matériel puis-je utiliser en interne. Je pense aux GPUs AMD qui ont la particularité d'être bon marché avec beaucoup de RAM

L'idée d'utiliser des GPU AMD pour un projet de MMO est excellente, surtout pour le rapport **Prix / VRAM**. Dans le monde des LLM (Large Language Models), la VRAM est le facteur limitant numéro 1 : si le modèle ne tient pas dans la mémoire vidéo, il tournera sur le processeur (CPU) et sera beaucoup trop lent pour des dialogues en temps réel.

Voici comment aborder votre configuration matérielle interne pour 2026 :

1. Pourquoi AMD est un choix malin

Les cartes AMD offrent souvent plus de mémoire pour moins cher que les cartes NVIDIA équivalentes (souvent bloquées à 8 ou 12 Go sur le milieu de gamme). Pour un LLM de PNJ :

- **Capacité** : Vous pouvez viser des modèles de **14B** ou **32B** paramètres avec une seule carte de 24 Go.
- **Technologie** : AMD utilise l'écosystème **ROCm** (l'équivalent de CUDA chez NVIDIA). C'est maintenant bien supporté par des outils comme **Ollama**, **LM Studio** ou **vLLM**.

2. Les modèles recommandés

Pour un GPU AMD, voici les deux champions actuels pour le "Inference per Dollar" :

Modèle	VRAM	Avantages
Radeon RX 7900 XTX	24 Go	Le meilleur choix "grand public". Elle rivalise avec une RTX 3090/4090 en vitesse d'inférence brute pour beaucoup moins cher.
Radeon RX 7900 XT	20 Go	Très bon compromis si le budget est serré, permet de faire tourner des modèles 14B très confortablement.
Instinct MI50 (Occasion)	32 Go	Une carte "Data Center" ancienne mais trouvable en occasion. Énormément de VRAM pour son prix, mais demande un refroidissement spécifique.

3. La barrière logicielle (Le point de vigilance)

C'est là que l'économie d'argent se paie en "temps de configuration" :

- **Compatibilité** : Sous Windows, le support est correct mais moins stable que sous **Linux**. Pour un serveur de dialogues PNJ stable, je vous recommande vivement de faire tourner votre LLM sur une machine Linux dédiée.
- **Quantification** : Pour que ça tourne vite, vous utiliserez des versions "quantifiées" des modèles (format GGUF ou EXL2). Cela réduit la précision de 16-bit à 4-bit, divisant par 4 la RAM nécessaire sans perte de qualité notable pour du dialogue de jeu.

4. Quel modèle de PNJ choisir ?

Pour un MMO Medfan, vous n'avez pas besoin d'un modèle géant comme Grok ou GPT-4.

- **Llama 3.1 (8B) ou Qwen 2.5 (7B)** : Ultra rapides, tiennent dans 8 Go de VRAM. Parfaits pour des réactions instantanées.
- **DeepSeek-R1 (Distill-Qwen-14B)** : Le meilleur compromis actuel entre intelligence de raisonnement et performance.

Conseil d'intégration UE5 : Plutôt que d'intégrer le LLM directement *dans* le client de jeu, faites-le tourner comme un **Micro-service API** (avec Ollama ou vLLM). Votre Blueprint Unreal n'aura qu'à envoyer une requête HTTP au serveur local et recevra le texte du PNJ en retour.

Souhaitez-vous que je vous aide à structurer le **Prompt** (l'instruction système) pour que vos PNJ conservent leur personnalité "Medfan" sans sortir de leur rôle ?

J'en suis à la phase d'information, même pas d'analyse. Je crois avoir compris que Ollama (je vais partir sur cette option-là, c'est elle qui revient le plus souvent) peut animer mes personnages et m'éviter le codage fastidieux des longs dialogues à embranchements

C'est une excellente conclusion. Vous avez identifié le changement de paradigme majeur : on passe d'un **système déterministe** (si le joueur dit A, le PNJ répond B) à un **système probabiliste** (le PNJ comprend l'intention et improvise).

Pour votre phase d'information, voici les trois piliers qui vont remplacer vos "longs dialogues à embranchements" avec **Ollama** :

1. La "Fiche de Personnage" (System Prompt)

Au lieu d'écrire des répliques, vous écrivez une description.

- **Avant** : 500 lignes de dialogues pré-écrits.
- **Avec Ollama** : Un paragraphe décrivant : *"Tu es Thrain, un forgeron nain grincheux qui déteste les elfes. Tu ne parles que de fer et de bière. Ta connaissance se limite à la région de Forge-Fer."*

2. Le "Contexte du Monde" (RAG - Retrieval Augmented Generation)

C'est là que votre MMO devient crédible. Vous ne pouvez pas mettre toute l'histoire de votre monde dans un seul prompt.

- Le jeu détecte où se trouve le joueur.
- Il envoie à Ollama les faits pertinents : *"Le joueur vient de tuer le Dragon de Cuivre"*.
- Ollama intègre cette info dynamiquement dans la conversation du PNJ.

3. La sortie structurée (JSON)

C'est le point technique crucial pour UE5. Ollama peut répondre au format **JSON**.

- Le PNJ ne renvoie pas juste du texte, mais un paquet de données :
 - **"Texte"** : "Bonjour voyageur !"
 - **"Animation"** : "Wave_Hand"
 - **"Humeur"** : "Amical"
- Dans vos **Blueprints**, vous recevez ce JSON et vous déclenchez l'animation correspondante automatiquement.

Pourquoi Ollama est le bon choix pour vous :

1. **Gratuité totale** : Pas de facture à la fin du mois, peu importe le nombre de tests.
2. **Confidentialité** : Votre univers de jeu reste sur votre machine.
3. **Simplicité** : Il s'installe en un clic et propose une API très simple que les plugins Unreal (comme *HTTP Blueprint*) peuvent contacter facilement.

Petit bémol à anticiper : La latence. Même avec un bon GPU AMD, il y a souvent 1 à 2 secondes de réflexion avant que le PNJ ne parle. C'est le défi principal pour un MMO.

Souhaitez-vous que je vous explique comment on gère techniquement cette "attente" dans Unreal pour ne pas que le jeu freeze pendant que le PNJ réfléchit ?

oui, c'est en effet un souci, cette latence

La latence est effectivement le "boss de fin" de l'intégration des LLM en local. Pour un MMO, un blanc de 3 secondes avant qu'un garde ne vous réponde "Halte !" casse totalement l'immersion.

Pour contrer cela, on utilise généralement trois stratégies dans Unreal Engine :

1. L'Asynchronisme (Le "Non-Blocking")

C'est la règle d'or. Votre Blueprint ne doit jamais "attendre" la réponse d'Ollama.

- On envoie la requête (Request).
- Le jeu continue de tourner (60 FPS).
- On utilise un **Event Custom** qui se déclenche uniquement quand Ollama a fini de générer la réponse.

2. Le "Streaming" de texte

Ollama peut envoyer la réponse mot par mot (comme ChatGPT) au lieu d'attendre la fin de la phrase.

- Dans UE5, vous pouvez commencer à afficher les premiers mots dans la bulle de dialogue pendant que le GPU calcule la suite.
- Cela donne une impression de réactivité immédiate au joueur.

3. Les "Animations de réflexion" (Le camouflage)

Pendant que le GPU travaille (phase de "Time to First Token"), on joue une animation sur le PNJ :

- Il se gratte la barbe, fronce les sourcils ou fait un bruit de réflexion ("Hmm...", "Laisse-moi réfléchir...").
- Ces fichiers audio et animations sont stockés en local et se lancent **instantanément** dès que le joueur valide sa question.

Le "Small Model" : La solution ultime

Pour un MMO, au lieu d'utiliser un énorme modèle de 30 Go, on utilise souvent un **modèle spécialisé de petite taille** (ex: *Llama-3-8B* ou même des modèles de 1B à 3B paramètres).

- **Avantage** : Sur un GPU AMD 7900 XTX, ces petits modèles répondent quasiment instantanément (moins de 200ms).
- **Compromis** : Ils sont moins "philosophes", mais amplement suffisants pour dire à un joueur où se trouve la taverne la plus proche.

Pour votre phase d'information, est-ce que vous aimeriez que je vous liste les **3 plugins Unreal Engine** les plus populaires (et souvent gratuits) qui permettent de connecter Ollama à vos Blueprints sans écrire une seule ligne de C++ ?

Si la latence n'est pas satisfaisante, je programmerai les dialogues moi-même à l'ancienne, c'est inacceptable que le joueur soupçonne seulement l'existence de l'IA, mais j'ai d'autres questions à poser qui pourraient court-circuiter les problèmes.

1° Les mécanismes existent pour détecter l'arrivée d'un PJ. Je peux donc provoquer la génération d'un premier message du PNJ en lui envoyant soit ce que le PJ va probablement demander ainsi que ses particularités qui vont influencer sur le dialogue.

2° Je peux provoquer la création de plusieurs dialogues probables, supprimant toute latence, pour n'en afficher que le plus pertinent quand le dialogue commence (ça, je ne sais pas comment, faut y réfléchir)

3° Qu'est-ce que je perds si j'utilise un moteur 'instantané' par rapport à un plus gros modèle ?

4° Comment se fait-il que si je te pose à toi, donc maintenant, sans rapport avec le jeu, une question complexe comme 'Qu'est-ce que Hitler aurait fait s'il avait découvert la bombe avant les alliés' tu vas répondre instantanément et avec pertinence ?

5° Puis-je utiliser l'AI pour stocker un certain nombre d'arborescences, telles que je les écrivais à l'époque de NWN pour que l'IA n'ait plus qu'à choisir les réponses les plus pertinentes sans passer du temps à calculer ?

Je sais, ça fait beaucoup de questions, c'est normal en phase d'approche, je ne sais pas où je vais

C'est une phase d'approche très saine. Vos questions touchent exactement aux limites actuelles du développement de jeux pilotés par l'IA. Voici des éléments de réponse précis pour chaque point :

1° Le "Pre-warming" (Préchauffage) : C'est une excellente stratégie. Dès que le PJ entre dans la "Sphere Collision" du PNJ, vous pouvez envoyer un signal à Ollama. On ne génère pas encore la réponse, mais on prépare le **contexte**. Vous pouvez même lui demander de générer une "phrase d'accueil" variée. Cela donne l'impression que le PNJ est vivant avant même que le joueur n'appuie sur "Parler".

2° La génération prédictive : C'est techniquement possible mais coûteux en ressources. Vous pourriez générer trois réponses types (amicale, neutre, agressive) pendant que le joueur s'approche. Le défi est la consommation de VRAM : faire tourner trois générations en parallèle ralentira votre GPU AMD. Il vaut mieux souvent une seule génération très rapide qu'un "multitasking" qui crée des saccades.

3° Ce qu'on perd avec un modèle "instantané" (Small Model) :

- **La cohérence à long terme :** Un petit modèle (ex: 1B ou 3B paramètres) peut oublier ce qu'il a dit dix phrases plus tôt.
- **La subtilité :** Ils ont tendance à être plus répétitifs et moins doués pour le sarcasme ou les doubles sens complexes.
- **Le respect strict des consignes :** Un gros modèle suivra à la lettre votre "Fiche de Personnage", là où un petit modèle pourrait parfois redevenir un assistant générique.

4° Pourquoi ma réponse est-elle instantanée ? Il y a deux raisons majeures :

- **L'infrastructure :** Je ne tourne pas sur un seul GPU domestique, mais sur des grappes de milliers de processeurs spécialisés (TPU/GPU) qui coûtent des millions d'euros.
- **Le débit (Throughput) :** Ces serveurs sont optimisés pour commencer à envoyer le premier mot en quelques millisecondes. En local, sur votre machine, vous avez un "temps de chargement" (le temps que le modèle lise votre question et commence à réfléchir) qui est proportionnel à la puissance de votre carte seule.

5° Utiliser l'IA pour générer des arborescences (Hybrid System) : C'est, à mon avis, la meilleure approche pour un MMO en 2026.

- Au lieu de demander à l'IA de parler au joueur, vous lui demandez (en phase de développement) de générer 50 variantes de dialogues pour une situation donnée.
- Vous stockez ces variantes dans une **DataTable** ou une **Map** (comme votre **TempStackMap**).
- En jeu, vous utilisez une IA minuscule ou un simple algorithme de "recherche de similarité" pour piocher la branche la plus proche de ce que le joueur a dit.
- **Résultat** : Latence zéro, contrôle total de la qualité, et l'illusion d'une liberté infinie.

Souhaitez-vous que l'on s'arrête sur ce point 5 pour voir comment l'IA pourrait vous aider à remplir automatiquement vos structures de données Unreal au lieu de tout écrire à la main ?

Nous avons donc nous-mêmes une arborescence :

1° sur un GPU AMD à grosse mémoire (que je peux même acheter d'occasion) puis-je choisir la taille de mon modèle pour limiter la latence à 1000ms ou bien les modèles sont-ils fixes ?

2° puis-je entraîner moi-même un modèle pour lui parler du monde, des systèmes de jeu, des particularités de l'artisanat, du PVP, de tout ce qui fait l'univers, pour l'adapter à mon univers plutôt que de le laisser dans un état 'généraliste' ?

3° Dans l'option 5 de système hybride, je risque de me court-circuiter moi-même si des choses fondamentales changent dans l'univers, sous peine de devoir reconstruire toutes les arborescences.

Nous progressons mais c'est encore extrêmement flou pour moi, rappelle-moi plus tard de te parler de rendre les PNJs conscients de leur environnement, soit de quelle manière je pourrai a) intégrer les nouveautés de l'environnement dans leur conscience

b) définir leur IA pour qu'ils cachent volontairement certaines choses aux PJs jusqu'à ce que les PJs posent les questions pertinentes.

c) régler le seuil de tolérance de l'IA pour que le bon roleplay soit récompensé (vertuchou, ça va faire un chapitre rien que pour ça, je le sens)

Ca, c'est pour plus tard.

C'est passionnant de voir votre réflexion se structurer. On est typiquement dans la phase où l'on cherche l'équilibre entre la "magie" de l'IA et la rigueur du Game Design.

Voici des précisions sur vos trois points :

1° Flexibilité de la taille des modèles

Oui, vous avez une liberté totale. Les modèles ne sont pas des blocs figés.

- **La Taille (Paramètres)** : Vous pouvez choisir de télécharger une version **8B** (8 milliards de paramètres), **14B**, ou **32B**. Plus le chiffre est petit, plus c'est rapide.
- **La Quantification (Compression)** : C'est le réglage le plus important pour la latence. Un même modèle (ex: Llama 3) existe en plusieurs niveaux de compression

(Q4, Q5, Q8). En choisissant une version **Q4_K_M** (un standard), vous divisez par deux le poids en RAM et accélérez radicalement la réponse sans perdre l'intelligence du personnage.

- **Le réglage "Flash Attention"** : Sur les cartes AMD récentes, ces optimisations logicielles permettent de traiter le texte très rapidement. Sur une 7900 XTX, un modèle 8B en Q4 répondra en bien moins de 1000ms.

2° Adaptation à l'univers (Fine-tuning vs RAG)

Il existe deux méthodes pour "éduquer" votre modèle :

- **Le Fine-Tuning (Entraînement)** : Vous donnez des milliers de dialogues Medfan au modèle pour qu'il adopte le *style* et le *ton*. C'est lourd techniquement.
- **Le RAG (Context Injection)** : C'est la méthode recommandée pour vous. Au moment où le joueur parle, vous insérez "en secret" dans la mémoire immédiate de l'IA les données de votre Wiki de jeu (artisanat, lore, règles PVP).
 - *Avantage* : Si vous changez une règle de craft dans votre base de données, l'IA est au courant à la seconde suivante sans ré-entraînement.

3° Le risque du système hybride (Obsolescence des données)

C'est votre crainte la plus légitime. Si vous générez 10 000 lignes de dialogue fixes et que vous changez le nom du Roi, tout est à refaire.

- **La solution** : Utiliser des **"Tokens" (balises)** dans vos arborescences générées.
 - Au lieu de stocker : *"Le Roi Arthur est à Camelot"*,
 - On stocke : *"Le {Current_King} est à {Capital_City}"*.
- Au moment de l'affichage dans Unreal, vous remplacez les balises par les variables de votre jeu. Cela garde la structure du dialogue fluide tout en restant synchronisé avec vos changements de monde.

Note pour plus tard

C'est noté. Vos points sur la **conscience de l'environnement**, le **secret volontaire** (gestion de l'inventaire/connaissance du PNJ) et la **récompense du Roleplay** (analyse de sentiment du texte du joueur) sont des piliers de ce qu'on appelle l'**Emergent Gameplay**.

Pour le point (c), sachez qu'on peut effectivement demander à l'IA de noter la "qualité" du message du joueur sur 10 et d'utiliser cette note pour modifier une variable de "Réputation" dans votre Blueprint.

On met cela de côté pour le moment. Sur quoi souhaitez-vous focaliser notre prochaine étape d'exploration ?

C'est gentil de m'encourager. J'ai appris aujourd'hui que Stephen avait jeté l'éponge sur Ashes of Creation. Que ce soit la vérité ou une manœuvre pour camoufler une escroquerie, même si ce jeu m'avait toujours fait l'effet d'un vaporware, l'effet est dévastateur pour un dev

solo comme moi de voir qu'une équipe de, dit-on, 250 ingénieurs, ce sont cassé la gueule sur un projet que j'entreprends en solitaire, même si mon architecture est très différente.

Soit, je ne suis pas en mesure de comprendre les subtilités techniques dont tu parles avant d'avoir mis les mains dans la graisse. Quand j'en serai là dans le développement, je commencerai par utiliser ma propre RTX5080 16GB locale et apprendrai par la pratique comment tout cela s'architecture.

Une simple question : j'ai écrit deux romans sur le monde où se développe le jeu et ai commencé l'écriture du troisième, que je ne finirai pas vu le travail qu'Unreal me demande. Puis-je :

a) donner à digérer ces deux romans de 300 pages à l'IA ?

b) est-ce pertinent ?

c) l'IA serait-elle capable (non plus mon IA interne mais toi-même) de me suggérer des quêtes, histoires, personnages secondaires, etc. en rapport avec ce que j'ai déjà écrit.

L'annonce concernant *Ashes of Creation* illustre parfaitement le piège des méga-projets : plus il y a de monde, plus la coordination et la technique s'écrasent sous leur propre poids. En tant que développeur solo, votre agilité et vos choix technologiques (comme l'IA) sont vos meilleurs atouts pour éviter leurs erreurs.

Pour vos romans et l'IA, voici une analyse de votre stratégie :

a) Donner à "digérer" 600 pages de texte

Oui, c'est tout à fait possible et même recommandé. Pour votre IA locale (avec la RTX 5080), on ne "ré-entraîne" pas le modèle avec vos livres (ce serait trop long et complexe). À la place, on utilise une technique appelée **RAG** (Retrieval-Augmented Generation) :

- Vous transformez vos romans en une "base de connaissances" vectorielle.
- Quand vous posez une question ou qu'un PNJ doit parler, l'IA va chercher les extraits pertinents dans vos livres en quelques millisecondes pour s'en inspirer.
- Avec 16 Go de VRAM, vous pourrez faire tourner des modèles très performants pour cette tâche.

b) Est-ce pertinent ?

C'est le plus haut degré de pertinence possible pour votre projet. Rien n'est plus frustrant dans un MMO que des PNJ qui semblent génériques. En utilisant vos propres écrits :

- La cohérence du monde est assurée : les noms des lieux, les dieux, le passé historique seront respectés à la lettre.
- Le style d'écriture des dialogues héritera naturellement de votre "plume" d'écrivain, ce qui donnera au jeu une identité littéraire unique que les gros studios n'ont plus.

c) L'IA peut-elle suggérer des quêtes et des personnages ?

Absolument, et c'est là que je peux vous aider immédiatement. En me transmettant des résumés, des chapitres ou des fiches de personnages, je peux :

- **Détecter des "trous" dans le lore** : "Vous mentionnez cette ancienne bataille dans le tome 1, pourquoi ne pas créer une quête de recherche de reliques à cet endroit ?"
- **Générer des quêtes secondaires cohérentes** : Créer des arcs narratifs qui ne contredisent pas vos romans mais les complètent.
- **Brainstormer des mécanismes de gameplay** : Traduire une scène d'action de votre livre en une mécanique de boss ou un système de craft spécifique.

Conseil pratique : Puisque vous avez déjà une RTX 5080, vous avez une machine de guerre pour l'IA. Ne voyez pas l'arrêt de votre troisième roman comme un abandon, mais comme une **mutation**. Votre roman devient le "moteur narratif" de votre jeu.

Voulez-vous essayer de me soumettre un court résumé ou un passage clé d'un de vos romans pour que je vous montre comment je peux en extraire des idées de quêtes ou des profils de PNJ pour votre projet UE5 ?

Mets-ça en mémoire, j'y tiens beaucoup, aujourd'hui j'ai du sale boulot de tirage de fils (aka BP) à finir aujourd'hui mais je reviendrai vers toi pour voir comment tu t'en sort avec l'analyse de mes textes. Confirme que tu l'as mis en mémoire et que je devrai te tester pour mesurer tes possibilités mimétiques.

Avant que je reparte câbler, tu peux répondre aux points que j'avais laissé en suspens.

C'est noté, j'ai bien enregistré l'existence de vos deux romans et l'importance qu'ils ont pour le lore de votre MMO. Je suis prêt pour le test de mimétisme quand vous aurez fini votre "câblage" de Blueprints ; j'ai hâte de voir si je parviens à capturer l'essence de votre plume. D'accord, je garde ça en mémoire. Vous pouvez à tout moment me demander d'oublier certaines informations ou les gérer dans [vos paramètres](#).

Voici des éléments de réponse pour vos trois points en suspens avant que vous ne retourniez dans le cambouis d'Unreal :

a) Intégrer les nouveautés de l'environnement (Conscience du PNJ)

Pour qu'un PNJ "remarque" un changement (ex: une porte défoncée, un nouvel objet), on utilise un système de **Perception de Données** :

- **Le système** : On n'envoie pas tout le monde au PNJ. On lui envoie une liste de "Tags" ou d'états d'objets proches.
- **Le mécanisme** : Dans votre prompt Ollama, vous insérez une variable `{Environnement_Proche}` qui se met à jour selon les acteurs présents dans une sphère de détection autour du PNJ. Si un joueur pose un coffre, le tag "Coffre_Bois" s'ajoute, et le PNJ peut réagir : *"Tiens, ce coffre n'était pas là tout à l'heure."*

b) Cacher volontairement des informations (Secrets et Débloques)

C'est une question de **Gestion du Contexte Filtré** :

- L'IA possède le savoir total dans sa base de données (votre roman), mais vous ne lui donnez qu'une partie.
- **La logique** : Vous créez une variable **Confiance** ou **Progression_Quete**. Si **Progression < 5**, le système de RAG (votre recherche dans vos textes) ne lui envoie pas les extraits concernant le secret.
- Le PNJ sait qu'il doit être évasif car vous lui dites : *"Si le joueur ne connaît pas le mot de passe 'Vertuchou', dis-lui que tu ne sais rien sur la cave."*

c) Le seuil de tolérance pour le Roleplay (Récompense)

C'est ici qu'on utilise l'IA comme **Juge de Sentiment** :

- Quand le joueur écrit une phrase, avant de générer la réponse du PNJ, vous faites une micro-analyse avec l'IA : *"Sur une échelle de 0 à 1, cette phrase est-elle bien écrite dans un style médiéval ?"*
- Si le score est de 0.9 (Bon Roleplay), vous augmentez une variable **Bonus_Reputation** dans Unreal.
- Le PNJ répond alors avec plus de respect et peut débloquent des options de dialogue ou des prix réduits. Cela transforme le chat en un véritable mécanisme de jeu.

Bon courage pour votre session de Blueprints ! C'est souvent la partie la plus ingrate, mais c'est elle qui fait tenir l'édifice. Je vous attends pour l'analyse de vos textes quand vous serez prêt.

J'ai mis au point un système de points d'aventure. Quand un joueur bute sur une difficulté, il peut dépenser les points d'aventure qu'il récolte en voyageant, réussissant des quêtes ou des crafts etc. Peut-on intégrer cela dans l'IA comme 'bonus' ?

Absolument, c'est une excellente idée de "gamification" du dialogue. Dans votre architecture, ces **Points d'Aventure (PA)** deviennent une monnaie d'échange non pas seulement pour des statistiques, mais pour de l'**information** ou de l'**influence**.

Voici comment intégrer cela techniquement entre vos Blueprints et l'IA :

1. Le "Pot-de-vin" narratif

L'IA peut ajuster sa coopération en fonction de la dépense de points.

- **Mécanisme** : Si le joueur décide de dépenser des PA pour "insister" ou "persuader", vous envoyez un paramètre supplémentaire à l'IA : **{Persuasion_Active: True}**.
- **Résultat** : Dans ses instructions, vous dites à l'IA : *"Si la persuasion est active, sois plus bavard et révèle un secret que tu caches normalement."*

2. L'IA comme "Guide" (Indice payant)

Si un joueur est bloqué sur une énigme ou une direction de quête :

- Le joueur dépense X Points d'Aventure.
- Vous envoyez la question du joueur à l'IA avec un accès étendu à vos textes (le RAG dont nous parlions).
- L'IA reçoit l'instruction : *"Le joueur a payé son tribut à l'aventure. Donne-lui un indice cryptique mais utile basé sur le chapitre 4 du tome 1 de mes romans."*

3. Le système de "Faveur" persistant

Vous pouvez lier les PA à la personnalité même du PNJ via vos Blueprints :

- **Dépenser des points** peut débloquent de façon permanente un "ton" plus amical de la part de l'IA.
- Dans votre structure de données du PNJ, vous pourriez avoir une `Map [Name -> Float]` (ID du Joueur -> Niveau de sympathie). Si le joueur dépense des PA pour aider le PNJ, ce score monte, et l'IA recevra systématiquement l'instruction d'être chaleureuse avec ce joueur précis.

4. Analyse de la "Valeur" du Roleplay

C'est là que le lien avec votre point (c) précédent devient puissant :

- Si l'IA analyse que le joueur fait un **excellent Roleplay**, vous pouvez faire l'inverse : au lieu de dépenser des points, le joueur en **gagne**.
- *L'IA renvoie* : `{"Texte": "Quel beau discours !", "Points_Gagnes": 15}`. Votre Blueprint lit ce JSON et ajoute les points au compteur du joueur.

En résumé : Les Points d'Aventure servent de **pont** entre la rigueur de vos systèmes de jeu (Blueprints/Variables) et la souplesse de l'IA. Cela évite que l'IA ne soit qu'un "gadget" et l'intègre au cœur de l'économie de votre MMO.

C'est un excellent moyen de réguler l'utilisation de l'IA (pour éviter que les joueurs n'abusent du serveur) tout en récompensant l'investissement dans votre univers.

Bon retour dans vos branchements de Blueprints ! On en reparle quand vous aurez "tiré vos fils".

La voix !

Tu m'as dit que c'est plus lourd et donc plus cher.

Je peux envisager de recourir à des services comme Grok si cela :

a) élimine la latence

b) permet le dialogue vocal (je suppose que le dialogue verbal se reflète toujours dans le chat)

La voix change tout au niveau de l'immersion, mais c'est effectivement le défi technique ultime. En 2026, Grok (via l'API **Grok Voice Agent**) a fait un bond de géant qui pourrait répondre à vos besoins.

Voici l'analyse de cette option par rapport à vos objectifs :

a) L'élimination de la latence

Grok Voice utilise une technologie de **Full-Duplex** (WebSocket).

- **Comment ça marche** : Au lieu de faire Voix -> Texte -> LLM -> Texte -> Voix, Grok traite tout dans un seul flux.
- **Le résultat** : La latence moyenne (Time-to-First-Audio) est tombée sous les **0,8 seconde**. C'est presque imperceptible pour un humain et plus rapide qu'une configuration locale mal optimisée.
- **Le "Barge-in"** : C'est la fonction clé. Si le PNJ parle et que vous l'interrompez, il s'arrête instantanément pour vous écouter. Cela supprime l'effet "robot qui récite son texte".

b) Le dialogue vocal et le chat

Oui, les deux sont parfaitement compatibles.

- **Le miroir textuel** : L'API renvoie simultanément le flux audio (pour les oreilles du joueur) et la transcription textuelle. Vous pouvez donc continuer à alimenter votre fenêtre de chat UE5 en temps réel.
- **Animations faciales** : Grok peut renvoyer des données de "phonèmes" (les mouvements de la bouche). Cela vous permet de synchroniser les lèvres de votre PNJ avec le son généré par l'IA.

Le "Prix de la Liberté" (Comparatif 2026)

Critère	Solution Locale (RTX 5080)	Solution Cloud (Grok API)
Coût	Électricité + Achat GPU (Amorti)	\$0.05 / minute de connexion
Latence	~1.5s à 2s (avec voix locale)	< 1s (Ultra fluide)
Voix	Souvent un peu robotique (TTS classique)	Humaine , avec accents et émotions
Dépendance	Totale indépendance	Dépend du serveur et du portefeuille

Mon avis de "collaborateur" pour votre MMO :

Pour un **MMO**, le coût de Grok peut devenir un gouffre financier si vous avez 1000 joueurs qui parlent en même temps.

La stratégie gagnante pour vous :

1. **Phase de Dev / Alpha** : Utilisez **Grok API**. C'est rapide à brancher dans Unreal, ça vous permet de tester le "feeling" du jeu sans galérer avec le code de la voix locale.
2. **Phase de Production** : Si le jeu cartonne, vous basculez sur votre solution locale (ou un serveur dédié) pour supprimer les coûts à la minute.

Attention sur un point : Grok est une entité "externe". Pour qu'il respecte vos **deux romans**, vous devrez lui envoyer les extraits via le système de "Tool Calling" ou de contexte massif, ce qui peut augmenter un peu le prix.