LOG8715- Architecture de jeu vidéo

Rapport TP3 :

Gestion de latence

Présenté à

Samuel Bellomo

&

Keven Chaussé

Réalisé par

Équipe 4

Emmanuel Doré-Walsh	1954113
Alex Hua	1994253
Vincent Beiglig	2161304

Prédiction

Pour la prédiction, nous avons permis au client de déterminer la position de son carré et des cercles par lui-même, et ce sans modifier la transmission des inputs au serveur pour que celui-ci soit en mesure de déterminer la position réelle du joueur. Cela permet au joueur de jouer sans se rendre compte que sa latence est très élevée, car ses inputs se traduisent en mouvement instantanément et le déplacement des cercles se fait de manière non saccadée.

Pour ce faire, les inputs du joueur sont transmis au client avant d'être utilisés pour calculer la position résultante du joueur du point de vue du client. Cette position est ajoutée à un registre de position. Ensuite, si cette position est à l'extérieur des bornes du jeu, un ajustement de la position est fait. Un transform est appliqué au carré en utilisant la dernière position ajoutée au registre.

Pour les cercles, leur position est prédite côté client à partir de leur vitesse et de leur position lors du premier tick. Pour terminer, l'effet de stun est appliqué côté client au moment où le joueur appuie sur espace. On utilise un timer local pour éviter la désynchronisation avec le serveur. Pour simuler le stun, les transforms cesse d'être appliqué tant que le timer n'est pas écoulé.

Réconciliation

Pour la réconciliation, seule la position du jeu est synchronisée avec le serveur, puisque le joueur n'interagit pas avec les cercles. Leur position réelle n'est pas importante, l'important est que leur déplacement ne laisse pas entrevoir la latence élevée. Pour réconcilier la position prédite avec celle établie par le serveur, un registre contenant toutes les positions calculées est mis en relation avec un registre de ticks. Lorsqu'un input est entré, celui-ci est envoyé au serveur avec le numéro du tick courant côté client par le biais d'un ServerRPC. De plus, l'input est utilisé pour calculer une position qui est ajoutée au registre. À cause de la latence, le serveur traite l'input alors que le client a déjà effectué plusieurs ticks.

Pour que le client sache à quel tick est associée la position reçue par le client de la part du serveur, le serveur envoie le numéro du tick reçu avec l'input par le biais de deux Network Variables. Du côté client, on retrouve la position prédite à l'aide du numéro de tick reçu et du registre de tick. La position prédite est ensuite soustraite de la position réelle et cette différence est ajoutée à toutes les positions prédites (si cette différence est assez significative) pour effectuer la réconciliation avant d'utiliser la dernière position du registre pour positionner le joueur côté client. (Voir la Figure 1)

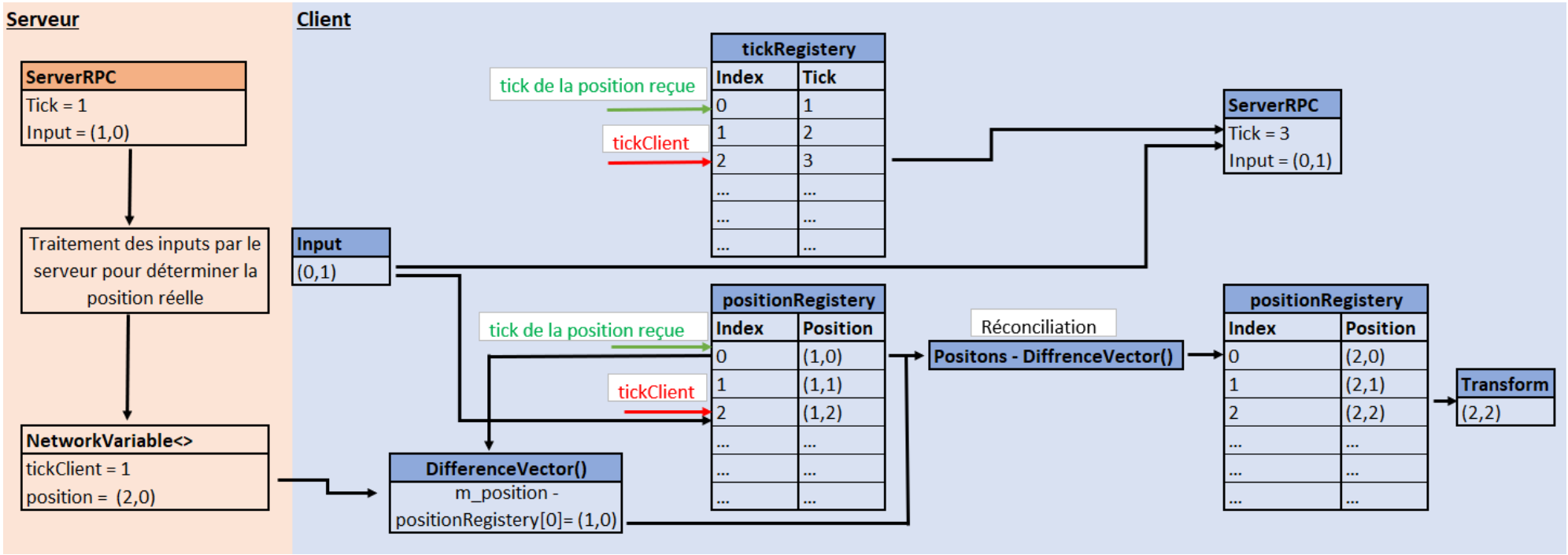

Figure 1 : schéma simplifié de la réconciliation entre la position déterminé par le server et la position prédite

L’intérêt de cette manière de faire est que la réconciliation est que l’on peut choisir une taille de registre de manière à rendre le jeu jouable avec une latence voulue : plus le registre est grand, plus l’on peut faire avec une grande latence. De manière générale à partir d’une certaine latence, un joueur sera appelé à régler ses problèmes de connexion et donc la taille de notre registre a été fixée pour fonctionner avec un RRT d’approximativement 2000 ms. Pour terminer, notre implémentation a montré une lacune, car le DiffrenceVector() est souvent mis à (0,0), alors que la position réelle et la position prédite d’un tick client ne sont pas les mêmes. Cela est sûrement dû à une erreur de logique de synchronisation que nous n’avons pas trouvée.