RType

Abstract

Ce document spécifie le protocole régissant les communications entre le serveur et les client du jeu RType.

Table des matières

Introduction

L'implémentation du netcode dans un jeu multijoueur nécessite de prendre en compte des considérations de sécurité. Ce document propose un protocole visant à garantir l'absence de tricherie par les joueurs. Ainsi il met en place un principe de streaming : le serveur détient l'état du jeu et notifie les différents clients des évènements qu'ils doivent répliquer localement. A l'inverse, les clients transmettent les interactions du joueur au serveur.

> Selon ce principe, le client ne fait qu'afficher diligemment les informations transmise par le serveur et transmet les touches pressé par le joueur. Aucune autre logique de gameplay ne doit y être présente.

Protocole

Création d'une nouvelle instance de Jeu

Les étapes suivantes décrivent les interactions entre un client souhaitant créer une nouvelle instance de jeu, sélectionner le niveau qu'il souhaite jouer et attendre d'autres joueurs.

Connexion

1. Se connecter vers l'endpoint du serveur sur le port 7654
2. Attendre le payload `WELCOME`.
3. Envoyer le payload `CREATE_INST|[PARTITION]`
4. Attendre le payload `EVENT|INST_CREATED|[UUID]`.
5. [Ecouter les `EVENT|INST_PLAYER_JOINED|[UUID]`]

Le `UUID` renvoyé par le serveur dans l'événement `INST_CREATED` correspond à l'identifiant de l'instance hébergé sur le serveur. C'est cet identifiant qui permettra à d'autres clients de rejoindre la partie.

> Le serveur va maintenant attendre d'autres connexions de client souhaitant rejoindre la partie. A tout moment, le client initiateur de l'instance peut commencer la partie, avec ou sans autre joueur. Cf. Commencer la partie.

Les `UUID`s renvoyés par le serveur dans les événements `INST_PLAYER_JOINED` sont les identifiant des autres joueurs prêt à jouer dans l'instance précédemment créé.

Rejoindre une partie

1. Se connecter vers l'endpoint du serveur sur le port 7654
2. Attendre le payload `WELCOME`.
3. Envoyer `JOIN_INST|[UUID]`
4. Attendre le payload `EVENT|INST_JOINED|[UUID];[UUID];[...]`.
5. [Ecouter les `EVENT|INST_PLAYER_JOINED|[UUID]`]

Les `UUID`s renvoyés par le serveur dans l'événement `INST_JOINED` sont les identifiant des joueurs déjà présent dans l'instance rejointe.

Les `UUID`s renvoyé par le serveur dans les événements `INST_PLAYER_JOINED` sont les identifiant des autres joueurs prêt à jouer dans l'instance précédemment rejointe.

Commencer la partie

Partie concernant le client initiateur de l'instance

1. Envoyer le payload `INST_START`
2. Attendre le payload `EVENT|INST_STARTING|[TIME]`
3. Ecouter les `EVENT|GAME_EVENT|[...]` et executer les actions.
4. Attendre `[TIME]`.
5. Ecouter les actions utilisateurs et envoyer des payloads `INPUT|[...]`

Partie concernant les autres joueurs ayant rejoint l'instance

1. Ecouter le `EVENT|INST_STARTING|[TIME]` et executer les actions.
2. Ecouter le `EVENT|GAME_EVENT|[...]` et executer les actions.
3. Attendre `[TIME]`.
4. Afficher le jeu et les entités reçus.
5. Ecouter les actions utilisateurs et envoyer des payloads `INPUT|[...]`

Après avoir demandé au serveur de commencer la partie, il est nécessaire de synchroniser les horloges de tous les joueurs. Chaque client va recevoir un événement de type `INST_STARTING` incluant un temps T (exprimé en millisecondes depuis Epoch) qui correspond au temps 0 de la partie. C'est à ce moment que la partie commencera, garantissant que les temps des clients et du serveur soit synchronisés.

> L'écoute des évènements commence avant le temps 0. Le serveur va directement commencer à envoyer des commandes concernant les Entités présentent dans la partie. Ces commandes seront daté relativement au temps 0 de la partie. Ces entités correspondent aux ennemis, aux autres clients, au background, etc. C'est au client de garder en mémoire ces informations et de les afficher correctement au moment demandé par le serveur.

Les payloads `INPUT`

Les payload `INPUT` sont envoyé par un client au serveur lorsque que le joueur interagit avec le RType.

> Exemple : `INPUT|UP` est envoyé à chaque frame par le client tant que le joueur presse la touche `UP` de son clavier.

Si le joueur appuie simultanément sur plusieurs touche, les clients devra faire une agrégation des évènements et les ajouter au payload.

> Exemple : `INPUT|UP|LEFT|FIRE`.

Les inputs

- `UP` : Touche "haut".
- `DOWN` : Touche "bas".
- `RIGHT` : Touche "droite".
- `LEFT` : Touche "gauche".
- `FIRE` : Touche "tirer".

Le binding des touches est laissé à la discretion de l'implémentation du client.

Les payload `GAME_EVENT`

Les payloads `GAME_EVENT` peuvent décrire :

- La création d'une nouvelle entité

- Des variations de propriétés d'entités dans le temps.
- La destruction d'une entité
- Le fin de la partie

Nouvelles entités

Format : `NEW_ENTITY|[TYPE]|[UUID]`

Le `TYPE` de l'entité permet au Client de charger les informations relatives à cette entité dans le fichier du niveau que le joueur a sélectionné. Notamment :

- Son sprite, si lieu
- Sa hitbox, si lieu
- etc.

Une entité nouvellement créée n'est pas affiché à l'écran et n'existe pas encore dans le jeu.

Variations de propriétés dans le temps

Translation dans l'espace

Format : `TEMPORAL_COORD|[UUID]|[TIME1]|X1;Y1|[TIME2]|X2;Y2`

Cet événement demande au client d'enregistrer une translation de coordonnée de l'entité d'identifiant `UUID`. Cette translation doit commencé aux coordonnées `X1;Y1` à temps `TIME1` vers les coordonnées `X2;Y2`. L'entité doit avoir atteint ces coordonnées au `TIME2`.

Destruction d'une entité

Format : `ENTITY_DESTROYED|[UUID]`

À la réception de ce payload, cette entitée doit disparaître du client, éventuellement en jouant une animation de destruction. Le client ne recevra plus aucun événement la concernant.

Fin de la partie

Format : `GAME_OVER|[RESULT]`.