GEN Projet

Protocole d'échange client - serveur

Introduction

Ce document décrit la structure des messages échangés par l'application client et l'application serveur. Les applications utiliseront des sockets TCP pour les communications critiques nécessitant une garantie de livraison (inscription ou connexion utilisateur, connexion à une salle d'attente, etc.) Pour les communications de jeu, lors de la course, ou les performances doivent être très rapides, les applications échangeront des datagrammes UDP. Les communications TCP se feront au moyen de requêtes et d'échanges d'objets JSON sérialisés, et les communications UDP seront constitués simplements d'envoi d'objets JSON.

Fonctionnalités

Protocole d'échange TCP

Le protocole d'échange TCP est spécifié dans le tableau ci-dessous. Comme expliqué dans le chapitre relatant de la conception du projet dans le rapport avec lequel est fourni ce document, les échanges entre le client et le serveur ne sont pas immédiats, mais nécessitent d'abord un traitement de la part du serveur. Ainsi la plupart des réponses du serveur spécifie le contexte dans lequel la ou les réponses suivantes s'inscrivent en répétant la requête effectuée auparavant par le client.

Utilisateur standard

Inscription utilisateur

Le client désire inscrire un nouvel utilisateur

Client	**Serveur**	**Description**
REGISTER_USER		Le client envoie la commande qui prévient le serveur qu'il désire inscrire un nouvel utilisateur
Envoi de données		Le client envoie son username sur une ligne, puis son mot de passe sur la ligne d'après.
	REGISTER_USER	
	SUCCESS	Le serveur envoie une commande de confirmation de l'inscription de l'utilisateur.
	ERROR	Le serveur envoie une commande informant le client d'une erreur lors de l'inscription. Il envoie ensuite une ligne décrivant la nature de l'erreur.

Connexion utilisateur

Le client désire connecter un utilisateur

Client	**Serveur**	**Description**
CONNECT_USER		Le client envoie la commande qui prévient le serveur qu'il désire connecter un utilisateur
Envoi de données		Le client envoie son username sur une ligne, puis son mot de passe sur la ligne d'après.
	CONNECT_USER	
	SUCCESS ERROR	Le serveur envoie une commande de confirmation de la connexion de l'utilisateur suivie d'un entier décrivant les droits d'accès de l'utilisateur. Le serveur envoie une commande informant le client d'une erreur lors de la connexion. Il envoie ensuite une ligne décrivant la source de l'erreur.

Déconnexion utilisateur

Le client se déconnecte du serveur

Client	**Serveur**	**Description**
DISCONNECT_USER		Le client indique au serveur qu'il se déconnecte du serveur.
		Le serveur ferme la connexion TCP.

Récupération des scores

Le client désire se connecter à une salle d'attente

Client	**Serveur**	**Description**

GET_SCORES		Le client indique au serveur qu'il désire connaître la liste des scores sur le serveur.
Envoi des données		Le client envoie le nom d'utilisateur éventuel dont il veut voir les scores. Il envoie une ligne vide si il désire recevoir tous les scores
	GET_SCORES	
	Envoi des informations	Le serveur renvoie une liste de scores correspondant à la demande du client sous la forme d'un objet JSON représentant une liste d'objets de type TCPScoreMessage. Exemple: `[{"id":1,"raceName":"Fire Dome","position":1,"time":607,"date":"2017-04-23","username":"Valomat"}]`

Récupérer la liste des salles d'attente

Le client désire connaître la liste des salles disponibles

Client	**Serveur**	**Description**
LIST_ROOMS		Le client envoie la commande qui prévient le serveur qu'il désire connaître la liste des salle
	LIST_ROOMS	
	Envoi des informations	Le serveur envoie les informations de salle sous la forme d'un JSON représentant une liste d'objets de type TCPRoomMessage. Exemple: `[{"name":"Funny room","id":"1"},{"name":"Competition room","id":"2"},{"name":"OkeyDokey room","id":"3"}]`

Connexion à une salle d'attente

Le client désire se connecter à une salle d'attente

Client	Serveur	Description
JOIN_ROOM		Le client indique au serveur qu'il désire se connecter à une salle d'attente spécifique.
Envoi des données		Le client envoie l'identifiant de la salle concernée
	JOIN_ROOM	
	SUCCESS	Le serveur notifie le client de la réussite de la connexion.
	ERROR	Le serveur notifie le client d'erreurs éventuelles. Il envoie ensuite une ligne décrivant la nature de l'erreur.

Déconnexion d'une salle d'attente

Le client se déconnecte d'une salle d'attente

Client	Serveur	Description
QUIT_ROOM		Le client indique au serveur qu'il quitte de la salle d'attente courante.
	QUIT_ROOM	
	SUCCESS	

Attente dans une salle d'attente

Lors de l'attente dans une salle d'attente, le serveur envoie cette commande lorsque l'état de la salle change (départ, venue d'un joueur).

Client	Serveur	Description
	ROOM_INFOS	Le serveur indique qu'il va envoyer les informations de la salle où l'utilisateur est connecté.

	Envoi des données	Le serveur envoie les informations sous la forme d'un objet de type TCPRoomInfoMessage. Cet objet contient en tout cas la liste des joueurs présents dans la salle, sous la forme d'objets de type TCPPlayerInfoMessage. Exemple: `{"name":"Funny room","players":[{"username":"player","state":"Waiting"}],"id":"0"}`

Attente dans une salle d'attente

Lors de l'attente dans une salle, un joueur peut également explicitement demander les informations de la salle.

Client	**Serveur**	**Description**
ROOM_INFOS		Le client demande les informations de la salle courante au serveur.
	ROOM_INFOS	Le serveur indique qu'il va envoyer les informations de la salle où l'utilisateur est connecté.
	Envoi des données	Le serveur envoie les informations sous la forme d'un objet de type TCPRoomInfoMessage. Cet objet contient en tout cas la liste des joueurs présents dans la salle, sous la forme d'objets de type TCPPlayerInfoMessage. Exemple: `{"name":"Funny room","players":[{"username":"player","state":"Waiting"}],"id":"0"}`

Notification qu'un joueur est prêt

Lors de l'attente dans la salle d'attente, le joueur indique qu'il est prêt quand il a fini de boire son café. Le client envoie alors la notification au serveur.

Client	Serveur	Description
USER_READY		Le client notifie le serveur que le joueur est prêt à joueur.

Lancement d'une partie

Au lancement de la partie, le serveur indique au joueur que la partie est lancée. A partir de là et jusqu'à la fin de la course, les informations seront échangés au travers de paquets UDP.

Client	Serveur	Description
	RACE_START	Le serveur envoie le signal de départ à tous les joueurs
	Envoi des données	Le serveur envoie le numéro de port sur lequel la connexion UDP peut être établie.

Fin d'une partie

A la fin d'une course, le serveur notifie les clients en leur indiquant la fin de la course.

Client	Serveur	Description
	RACE_END	Le serveur envoie le signal de fin de course à tous les clients.
	Envoi des données	Le serveur envoie les scores finaux aux client sous la forme d'un objet JSON représentant une liste d'objets de type TCPScoreMessage. Exemple: `[{"id":1,"raceName":"Fire Dome","position":1,"time":607,"date":"2017-04-23","username":"Valomat"}]`

Notification de déconnexion

En cas de déconnexion forcée d'une salle (expulsion, suppression de la salle), le serveur envoie une commande de notification au joueur.

Client	Serveur	Description
	ROOM_DISCONNECTION	Le serveur indique au client qu'il a été déconnecté d'une salle. Le serveur revient à un état après connexion utilisateur.

Compte à rebours de début de course

Le serveur notifie un client du compte à rebours du début de course

Client	Serveur	Description
	COUNTDOWN	
	Envoi de données	Le serveur envoie un entier représentant l'état du compte à rebours.

Notification qu'un joueur a atteint la ligne d'arrivée

Le client notifie le serveur qu'il a atteint la ligne d'arrivée

Client	Serveur	Description
USER_FINISHED		

Utilisateur administrateur

Création d'une salle d'attente

Le client désire créer une salle d'attente

Client	Serveur	Description
CREATE_ROOM		Le client indique au serveur qu'il désire créer une salle d'attente.
Envoi des données		Le client envoie le nom de la salle.

Suppression d'une salle d'attente

Le client désire supprimer une salle d'attente

Client	**Serveur**	**Description**
DELETE_ROOM		Le client indique au serveur qu'il désire supprimer une salle d'attente
Envoi des données		Le client envoie l'identifiant de la salle.

Expulsion d'un joueur

Le client désire expulser un joueur d'une salle

Client	**Serveur**	**Description**
BAN_USER		Le client indique au serveur qu'il désire expulser un utilisateur
Envoi des données		Le client envoie un identificateur d'un joueur.
	BAN_USER	
	SUCCESS ERROR	Le serveur notifie le client de la réussite de l'expulsion Le serveur notifie le client d'une erreur. Il envoie ensuite une ligne décrivant l'erreur.

Récupérer la liste des utilisateurs

Le client désire récupérer la liste des utilisateurs

Client	**Serveur**	**Description**
GET_USERS		Le client indique qu'il désire récupérer la liste des utilisateurs
Envoi des données		Le client envoie un nom d'utilisateur ainsi qu'un nouveau statut de privilèges.
	GET_USERS	

	Envoi de données ERROR	Le serveur envoie la liste d'utilisateurs sous la forme d'un objet JSON sérialisé. Le serveur notifie le client d'une erreur. Il envoie ensuite une ligne décrivant l'erreur

Modification des droits d'un utilisateur

Le client désire modifier les droits d'un utilisateur

Client	**Serveur**	**Description**
USER_RIGHTS		Le client indique qu'il désire modifier les droits d'un client.
Envoi des données		Le client envoie un nom d'utilisateur ainsi qu'un nouveau statut de privilèges.
	USER_RIGHTS	
	SUCCESS ERROR	Le serveur notifie le client de la réussite de la modification. Le serveur notifie le client d'une erreur. Il envoie ensuite une ligne décrivant l'erreur

Protocole d'échange UDP

Le protocole d'échange UDP est utilisé pour échanger des informations entre le serveur et les clients lors de la phase de jeu de la course. Les paquets seront échangés pour communiquer des informations concernant la partie en cours.

Il est constitué des deux classes UDPPlayerMessage et UDPRaceMessage dont les instances sont sérialisées en JSON avant d'être envoyées par paquet UDP.

Les tableaux suivant représentent les attributs qui sont ainsi transmis entre le serveur et le client.

```
UDPPlayerMessage
Pour communiquer la position d'un joueur au serveur
```

```
float posx;
float posY;
float velX;
float velY;
int color;
```

UDPRaceMessage Pour communiquer l'état de la course au client
String raceName; private long time; List<UDPPlayerMessage> players;

Ces objets sont donc sérialisés et envoyés sur des sockets UDP.