Équipe No. 3

Projet Intégrateur 3
Protocole de communication

Version 1.3

Historique des révisions

Date	Version	Description	Auteur
2017-02-03	1.0	Début description des paquets	Ulric Villeneuve
2017-02-05	1.1	Description des paquets Introduction et communication client-serveur	Ulric Villeneuve Simon-Pierre Desjardins
2017-02-06	1.2	Sérialiseur-désérialiseur	Ulric Villeneuve
2017-02-08	1.3	Correction et finalisation	Simon-Pierre Desjardins, Ulric Villeneuve
2017-02-09	1.4	Ajustement du protocole en fonctions de spécificités du client léger.	Ulric Villeneuve Camille Gendreau

Table des matières

Protocole de communication

1. Introduction

Le document du protocole de communication permet d'étaler les détails sur le choix du moyen de communication client-serveur de notre application ainsi que le contenu des différents paquets utilisés. Il documente aussi le rôle du sérialiseur-désérialiseur.

2. Communication client-serveur

Tout d'abord, les communications entre les clients et le serveur peuvent être résumés assez aisément. Cela débute par un changement sur un client (par exemple un changement de vitesse du robot). Cette information sera ensuite transmise dans un paquet dédié à ce type d'information afin d'être relayé jusqu'au serveur. Le serveur va par la suite retransmettre cette information dans le même type de paquet à tous les autres clients présents afin que ceux-ci puissent observé également la modification du client en question (le changement de vitesse dans ce cas-ci). Le modèle de communication client/serveur TCP/IP sera utilisé afin de répondre aux besoins du système. De plus, le protocole internet utilisé sera celui le plus commun, c'est à dire IPv4. Ces choix sont ceux de base pour un système de communication client-serveur et notre application n'aura pas besoin de plus que cela; le nombre d'adresses supplémentaires que le protocole internet IPv6 peut fournir n'est pas nécessaire dans le cadre de ce projet. Ce modèle a également été choisi, car certains membres possédaient déjà des connaissances sur l'implémentation d'un client-serveur avec les particularités décrites plus haut.

3. Description des paquets

En général

Ci-dessous seront décrits chacun des différents formats de messages qui sont envoyés à partir des clients légers ou lourds vers le serveur, et vice versa.

En général, les paquets respectent le format suivant.

Index des bytes	[0]-[3]	[4]	[5]	[6] - [longueur+1]
Information	Longueur	Système	Commande	Données

Dans ce format, les quatres premiers bytes indiquent la longueur en bytes du message en s'incluant dans cette longueur. Le message est ensuite composé d'un byte indiquant à quel système le message s'adresse. Il peut s'agir, par exemple, du système de clavardage, du module de physique ou du système de salles de jeu. La composante "Commande" correspond au type de demande qui sera envoyée au système spécifié. Par exemple, pour le module de physique, les commandes possibles sont : Position ("p"), Vitesse ("v"), Rotation("r"), Vitesse Angulaire ("w") et Mise à Échelle ("s"). La composante "Données" contient les données complémentaires à la commande qui doivent être traitées par le système spécifié. Par exemple, dans le cas d'un changement de position, les données envoyées correspondent au ID de l'objet modifié et à la nouvelle position.

Exemple de paquet envoyé au système de physique demandant de changer la position d'un objet :

Index des bytes	[0]-[3]	[4]	[5]	[6] - [longueur-1]
Information	Longueur	Système	Commande	Données

Exemple de données du paquet	25	p	p	id(int) = 03;
				x(float) = 10.0;
				y(float) = 0.0;
				z(float) = 1.0

Dans cet exemple, le paquet est analysé par le sérialiseur-désérialiseur et est ensuite acheminé au module correspondant, soit celui de la physique en tant que changement de position d'un objet. Ensuite, la physique applique le changement, c'est-à-dire que le module assigne les trois valeurs float séparés par des ";" aux variables "x", "y" et "z" représentant la position de l'objet indiqué par "id".

Système de clavardage

Les formats de paquets concernant le système de clavardage permettent la connexion et la déconnexion d'un utilisateur à un chat, l'envoie d'un message par un utilisateur dans un canal, la création et la suppression d'un canal.

Tableau des commandes du système:

Système	**Commande**	**Données**	**Description**
c	j	utilisateur(string);canal(string)	Fait rejoindre l'utilisateur au canal de clavardage "canal" ou le crée s'il n'existe pas.
	q	utilisateur(string);canal(string)	Fait quitter l'utilisateur du canal de clavardage "canal" et supprime le canal s'il n'y reste personne.
	m	utilisateur(string);canal(string); heure(string);date(string); message(string)	Envoie de message de l'utilisateur vers le canal avec l'heure, la date et le message.
	l	utilisateurs(string)	Fournit une liste d'utilisateurs avec séparés par des virgules

Paquet envoyé lorsqu'un utilisateur quitte un canal :

Index des bytes	[0]-[1]	[2]	[3]	[4] - [longueur-1]
Information	Longueur	Système	Commande	Données
Exemple de données du paquet	13	c	q	userId(int) = 05; canalID(int) = 02

Dans cet exemple, le paquet est analysé par le sérialiseur-désérialiseur et est ensuite acheminé au module correspondant, soit celui du clavardage en tant que déconnexion d'un utilisateur à un canal. Ensuite, le système de clavardage applique le changement, c'est-à-dire qu'il retire l'utilisateur dont l'ID est "05" du canal dont l'ID est "02" et supprime celui-ci s'il ne reste plus d'utilisateur dans le canal..

On peut préciser que les paquets indiquant l'envoie d'un message par un utilisateur dans un canal, doivent contenir le message, la date et l'heure de l'envoie et l'auteur du message.

Système de sessions multi-utilisateurs

Les formats de paquets concernant le système de sessions multi-utilisateurs permettent la création ou la suppression d'une carte, la connexion et la déconnexion d'un utilisateur à une session ainsi que le chargement d'une carte lorsqu'un utilisateur s'y connecte.

Tableau des commandes du système:

Système	**Commande**	**Données**	**Description**
m	j	mapId(string); sessionType(char)	Envoyé par le client pour ouvrir ou joindre une session multi-utilisateur avec la carte identifiée par "mapId" en une session de type "sessionType".
	l	-	Envoyé par le client lorsqu'il veut quitter la session multi-utilisateur dans laquelle il se trouve.

	c	sessionType(char);
		adminUsers(string);
		mapData(string);
			Envoyé par le client pour créer une nouvelle carte sur le serveur avec le type de carte à créer ainsi que la liste d'utilisateurs avec droits d'administrateur sur cette carte. "mapData" contient une carte en json si le message sert à charger une carte déjà construite localement.

	d	mapId(string)	Envoyé par le client pour supprimer la carte du serveur.
	p	mapId(string);username(string); permissions(string)	Envoyé par un client qui tente de changer les permissions de l'utilisateur "username" sur la carte "mapId".

Exemple de paquet envoyé lorsqu'un utilisateur se joint à une salle de jeu:

Index des bytes	[0]-[1]	[2]	[3]	[4] - [longueur-1]
Information	Longueur	Système	Commande	Données
Exemple de données du paquet	8	m	c	sessionType(char) = e; adminUsers(string) = "bob"; mapData(string) = ""

Dans cet exemple, le paquet est analysé par le sérialiseur-désérialiseur et est ensuite acheminé au module correspondant, soit le système de salles de jeu en tant que création d'une carte. Cette carte est crée en mode édition tel que spécifiée par "sessionType". Ensuite, l'application offre les privilèges d'administrateurs aux utilisateurs listés, donc "bob".

Module de physique

Les formats de paquets concernant le module de physique permettent le changement de position, de vitesse, de direction, de vitesse angulaire et de mise à échelle d'un objet.

Tableau des commandes du système:

Système	**Commande**	**Données**	**Description**
p	r	id(int); x(float);y(float);z(float)	Effectue une rotation de l'objet correspondant à l'identifiant "id" d'un nombre de degré autour des axes représentés par les valeurs "x", "y" et "z".
	v	id(int); x(float);y(float);z(float)	Effectue la mise à jour du vecteur de vitesse de l'objet identifié par "id"
	p	id(int); x(float);y(float);z(float)	Effectue la mise à jour du vecteur de position de l'objet identifié par "id"
	w	id(int); x(float);y(float);z(float)	Effectue la mise à jour de la vitesse angulaire de l'objet identifié par "id"
	s	id(int); x(float);y(float);z(float)	Effectue la mise à jour de la mise à échelle de l'objet identifié par "id".

Exemple de paquet envoyé lorsqu'un utilisateur modifie la direction d'un mur.

Index des bytes	[0]-[1]	[2]	[3]	[4] - [longueur-1]
Information	Longueur	Système	Commande	Données

Exemple de données du paquet	23	p	r	id(int) = 02;
				x(float) = 90.0;
				y(float) = 0.0;
				z(float) = 0.0

Dans cet exemple, le paquet est analysé par le sérialiseur-désérialiseur et est ensuite acheminé au module correspondant, soit le module de physique en tant que changement de rotation d'un objet. Ensuite, l'application s'occupe de modifier la rotation de l'objet dont l'ID est "02" avec les nouvelles valeurs fournies.

Module d'authentification d'utilisateurs

Le module d'authentification d'utilisateurs à pour but de régir la création, l'authentification et la suppression des comptes utilisateurs sur le serveur. C'est aussi le module qui s'occupe d'accueillir la connexions d'un client au serveur.

Tableau des commandes du système:

Système	**Commande**	**Données**	**Description**
u	c	username(string)	Envoyé par un client qui tente de créer un nouveau compte utilisateur
	l	username(string)	Envoyé par un client qui tente de se logger avec le nom utilisateur "username".
	a	authenticationResult(char);	Envoyé par le serveur à un client qui tente de s'identifier. Informe le client si l'authentification est une succès ou si une erreur a eu lieu.
	m	property(string); value(string);	Envoyé par le client pour modifier une propriété de l'utilisateur.

Exemple de paquet envoyé lorsqu'un utilisateur tente de créer un nouveau compte utilisateur

Index des bytes	[0]-[1]	[2]	[3]	[4] - [longueur-1]
Information	Longueur	Système	Commande	Données
Exemple de données du paquet	11	u	c	username(string) = "JohnDoe"

Dans cet exemple, le paquet est analysé par le sérialiseur-désérialiseur et est ensuite acheminé au module correspondant, soit le module d'authentification d'utilisateurs en tant que tentative de création d'un nouvel utilisateur. Ensuite, l'application s'occupe de tenter d'ajouter le nouvel utilisateur avec le nom d'utilisateur "JohnDoe", s'il existe déjà il ne le crée pas à nouveau.

Module d'édition de carte

Le module d'édition de carte s'occupe de la création d'objets dans la carte, leur sélection et de leur suppression au besoin.

Tableau des commandes du système:

Système	**Commande**	**Données**	**Description**
e	c	type(char);x(float); y(float);z(float);id(string)	Envoyé lorsqu'un objet de type "type" est créé à la position x, y, z. Lorsque le client envoie ce message, le serveur assigne un identifiant"id" à l'objet et renvoie ce message à tous les utilisateurs comme confirmation de la création de l'objet à ce point.
	d	id(string)	Indique la supression de l'objet indiqué par "id".
	s	id(string)	Indique la sélection de l'objet

Exemple de paquet envoyé lorsqu'un utilisateur tente de créer un nouveau compte utilisateur

Index des bytes	[0]-[1]	[2]	[3]	[4] - [longueur-1]
Information	Longueur	Système	Commande	Données
Exemple de données du paquet	13	e	d	id(string) = "045fecd12"

Dans cet exemple, le paquet est analysé par le sérialiseur-désérialiseur et est ensuite acheminé au module d'édition, soit le module édition. Ensuite, le serveur supprime l'objet identifié par "id" dans sa version de la carte avant de renvoyer le message aux autres utilisateurs.

4. Sérialiseur-désérialiseur

Le sérialiseur doit détenir les règles de transformations nécessaires afin de pouvoir créer un paquet avec l'instruction qui lui est fournie. C'est donc son devoir d'arriver aux formats de paquets définis dans la section 3 à partir des commandes qui sont passées.
Tandis que le désérialiseur fait l'opération inverse, c'est-à-dire qu'il doit détenir les règles de décomposition des paquets. Cela signifie qu'il doit être en mesure de recevoir tous les paquets utilisés dans le cadre de l'application et de les convertir en instructions à exécuter.

5. Analyse des paquets et traitement de l'information

Le sérialiseur-désérialiseur a la tâche de décomposer les paquets reçus en instructions, mais une fois que ces instructions sont passées au système adéquat, c'est ce dernier qui se charge de les exécuter.
Par exemple, lorsque le sérialiseur-désérialiseur reçoit un paquet correspondant à un changement de position d'un objet, il ne fait que transmettre l'information au module de physique. Puis, celui-ci affecte les nouvelles valeurs à la position de l'objet concerné en faisant les appels nécessaire.