T6

Projet intégrateur LOG3900
Protocole de communication

Version 1.4

Historique des révisions

Date	Version	Description	Auteur
2019-02-07	1.0	Première rédaction du document	Kelvin Tran
2019-02-07	1.1	Révision du document, apport de corrections mineures	Freddy Sossa
2019-02-08	1.2	Ajout de type de requêtes à la section 3.6	Kelvin Tran
2019-02-08	1.3	Correction de la mise en page et mise à jour de la table des matières	Kelvin Tran
2019-02-08	1.4	Révision finale du document	Amal Metsahel

Table des matières

Protocole de communication

1. Introduction

Ce document contient les détails sur le protocole de communication qui sera utilisé dans le projet Poly Paint Pro. Ce document est divisé en 2 sections principales. La section 2 contient les informations sur le moyen de communication utilisé entre le client et le serveur. La section 3 contient la forme du contenu des différents paquets qui seront envoyés.

2. Communication client-serveur

La communication client-serveur se fait à travers une connexion socket TCP/IP. Avant de pouvoir échanger des données entre le client et le serveur, le client tente en premier de se connecter à un socket TCP au serveur. Le serveur lui répond et lui accorde un accès si les toutes les données entrées sont valides. Toutes les données sont sérialisées en JSON, donc en string, avant d'être transmises de client à serveur et vice-versa. L'encodage et le décodage se font autant du côté serveur que du côté client.

Après une connexion TCP établie, le serveur tente à priori d'identifier et d'authentifier le client qui s'est connecté. Si le client ne peut pas s'authentifier, le serveur transmet un message d'erreur au client et ce dernier ferme la connexion TCP au serveur.

Si l'authentification est réussie, la connexion TCP est maintenue et divers types de paquets peuvent être envoyés continuellement entre le client et le serveur dépendamment de l'état du client.

3. Description des paquets

Toutes les données des paquets envoyées du client au serveur ou du serveur au client sont de format JSON. Un modèle du JSON a été défini afin d'assurer une complémentarité et une efficacité lors des communications entre un client léger et un client lourd. Ce modèle JSON contient un champ nommé "type" qui indique le type de paquet envoyé. En effet, ce champ permet au destinataire de savoir quel type de données il reçoit et de les traiter d'une façon adéquate.

Voici les différents types de paquets et leur valeur pour le champ "type" :

AUTH = 0
MESSAGE = 1
ERROR = 2
EVENT = 3
OPERATION = 4
REQUEST = 5
RESPONSE = 6

3.1 Paquet de type AUTH

Le paquet de type AUTH est utilisé pour s'authentifier au serveur. Il est envoyé après l'établissement de la connexion TCP. Voici la forme du contenu de ce paquet:

```
{
	type: 0,
	id: "id de l'utilisateur",
	password: "mot de passe de l'utilisateur"
}
```

3.2 Paquet de type MESSAGE

Le paquet de type MESSAGE est utilisé pour envoyer des messages du client au serveur pour les sessions de clavardage, ainsi que pour diffuser les messages provenant du serveur aux autres clients. Il représente en soi un message. Voici la forme du contenu de ce paquet:

```
{
	type: 1,
	sender: "id de l'utilisateur qui a écrit le message",
	channel: "id du canal de clavardage",
	content: "contenu du message",
	timestamp: "date et temps de création du message en format ISO 8601"
}
```

3.3 Paquet de type ERROR

Le paquet de type ERROR est utilisé pour signaler une erreur quelconque lors de la communication. Il peut être utilisé du client vers le serveur, et vis-versa. Voici la forme du contenu de ce paquet:

```
{
	type: 2,
	error: "id de l'erreur",
	message: "message d'erreur",
}
```

3.4 Paquet de type EVENT

Le paquet de type EVENT est utilisé pour informer un ou plusieurs partis de l'occurrence d'un événement non opérationnel. Cet événement peut être autant du côté client que du côté serveur. Voici la forme du contenu de ce paquet:

```
{
	type: 3,
	name: "type d'événement",
	context: "contexte de l'événement (ex: id du canal de clavardage)"
	source: "id de la source de l'événement (ex: id de l'utilisateur)",
	timestamp: "date et temps du lancement de l'événement en format ISO 8601"
}
```

3.5 Paquet de type OPERATION

Le paquet de type OPERATION est utilisé pour transmettre l'information sur les opérations effectuées sur un canevas. Elle représente toute opération d'édition ou de modification effectuée au cours d'une session de dessin. Voici la forme du contenu de ce paquet:

```
{
```

```
        type: 4,
        operation: (type d'opération),
        source: "id de la source de l'opération",
        target: "id du canevas cible de l'opération",
        ...
        (autres champs variant selon l'opération)
}
```

Voici les différents types d'opérations possibles:

```
CREATE = 0
DELETE = 1
MODIFY = 2
SELECT = 3
RESET = 4
```

3.5.1 Opération CREATE

L'opération CREATE permet la création d'un objet quelconque. Voici la forme du contenu du paquet:

```
{
        type: 4,
        operation: 0,
        source: "id de la source de l'opération",
        target: "id du canevas cible de l'opération",
        object: {
                id: "id de l'objet créé"
                type: "type de l'objet",
                position: {
                        x: (position sur le plan x),
                        y: (position sur le plan y),
                        z: (position sur le plan z)
                }
        }
}
```

3.5.2 Opération DELETE

L'opération DELETE permet de supprimer un objet quelconque. Voici la forme du contenu du paquet:

```
{
        type: 4,
        operation: 1,
        source: "id de la source de l'opération",
        target: "id du canevas cible de l'opération",
        object: "id de l'objet à supprimer"
}
```

3.5.3 Opération MODIFY

L'opération MODIFY permet de modifier les propriétés d'un objet quelconque. Voici la forme du contenu du paquet:

```
{
	type: 4,
	operation: 2,
	source: "id de la source de l'opération",
	target: "id du canevas cible de l'opération",
	object: {
		id: "id de l'objet"
		type: "type de l'objet",
		position: {
			x: (position sur le plan x),
			y: (position sur le plan y),
			z: (position sur le plan z)
		},
		rotation: (rotation de l'objet),
		scale: (facteur de taille de l'objet),
		connections: [(objets connectés), ...]
	}
}
```

Les champs dans la section propriétés peuvent être nuls s'il n'y a pas de modification à effectuer sur cette propriété.

3.5.4 Opération SELECT

L'opération SELECT permet de sélectionner ou de désélectionner un objet quelconque. Voici la forme du contenu du paquet:

```
{
	type: 4,
	operation: 3,
	source: "id de la source de l'opération",
	target: "id du canevas cible de l'opération",
	object: "id de l'objet à sélectionner"
}
```

3.5.4 Opération RESET

L'opération RESET permet de réinitialiser le canevas au complet. Voici la forme du contenu du paquet:

```
{
	type: 4,
	operation: 4,
	source: "id de la source de l'opération",
	target: "id du canevas à réinitialiser",
}
```

3.6 Paquet de type REQUEST

Le paquet de type REQUEST est utilisé pour faire une requête de données. Le parti qui reçoit ce paquet REQUEST doit répondre avec un paquet de type RESPONSE. Voici la forme de ce paquet:

```
{
	type: 5,
	request: "type de requête"
}
```

Voici les différents types de requêtes possibles :

3.6.1 Requête "publicCanvases"

Requête d'une liste de tous canevas publique

3.6.2 Requête "canvasInfo-[Canvas ID]"

Requête des informations sur un canevas spécifique

3.6.3 Requête "canvas-[Canvas ID]"

Requête des données d'un canevas spécifique (données du fichier lui-même)

3.6.4 Requête "publicChats"

Requête d'une liste de tous les canaux de clavardage publics

3.6.5 Requête "chat-[Chat ID]"

Requête des informations sur un canal de clavardage spécifique

3.6.6 Requête "session-[ID]"

Requête des informations sur une session d'édition spécifique

3.7 Paquet de type RESPONSE

Le paquet de type RESPONSE est utilisé pour envoyer des données face à une requête de données. Voici la forme de ce paquet:

```
{
	type: 6,
	request: "type de requête auquel on répond"
	data: {
		(données demandées dépendant de la requête)
	}
}
```