Air Hockey Circus

Protocole de communication

Version 3.0

Historique des révisions

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Auteur** |
| 2013-01-23 | 1 | Rédaction initiale du document | Simon Delisle  François Pierre Doray |
| 2013-04-07 | 2 | Mise à jour du document | Simon Delisle |
| 2013-07-07 | 3 | Mise à jour du de la section iOS | François Pierre Doray |
|  |  |  |  |
|  |  |  |  |

Table des matières

1. Introduction 4

2. Communication entre les clients lourds et le serveur 4

2.1 Authentification 4

2.2 Créer une partie 4

2.3 Afficher une liste des parties en attente de joueurs 5

2.4 Rejoindre une partie 5

2.5 Déroulement d’une partie 6

2.6 Déroulement d’un tournoi 6

2.7 Clavardage 6

2.8 Modification du profil 6

3. Communication et description des paquets client léger-serveur 8

3.1 Authentification 8

3.2 Publication d’une table sur le serveur 8

Protocole de communication

# Introduction

Ce document décrit en détails les méthodes qui seront utilisées pour la communication entre le serveur et les clients lourds ainsi qu’entre le serveur et les clients légers. Dans les deux cas, on explique d’abord le protocole général. Puis, on offre une description détaillée des paquets qui seront échangés.

# Communication entre les clients lourds et le serveur

Le serveur aura un port dédié port écouter les connexions des clients lourds. Les clients lourds utiliseront un socket TCP pour initier une connexion sur ce port lorsqu’ils en auront besoin. Pour communiquer, le client et le serveur utiliseront principalement des objets sérialisés à l’aide d’un sérialiseur binaire du framework .NET. Plus précisément, les messages échangés auront le format suivant :

* Entier indiquant la taille de l’objet sérialisé (4 octets)
* Objet sérialisé (taille variable)

Durant le jeu, les données de position des objets seront échangés avec un protocole différent afin d’obtenir de meilleures performances. Une connexion UDP dédiée à cette tâche sera créée à l’aide d’un nouveau socket.

Les sections suivantes décrivent les classes des objets sérialisés échangés entre le client et le serveur ainsi que le contenu des paquets transitant sur la connexion UDP dédiée durant le jeu.

## Authentification

Classe MessageAuthentification, Client 🡪 Serveur

|  |  |
| --- | --- |
| String | Nom d’utilisateur |
| String | Mot de passe encrypté |
| Bool | Indique si l’utilisateur souhaite que l’on se souvienne de lui. |

Classe MessageResultatAuthentification, Serveur 🡪 Client

|  |  |
| --- | --- |
| Bool | Indique si les informations sont valides |
| String | Clé de connexion à utiliser pour restaurer la connexion en cas de défaillance |
| String | Clé pour la fonctionnalité « Se souvenir de moi » |

## Créer une partie

Un joueur peut créer une nouvelle partie.

Classe MessageCreerPartie, Client 🡪 Serveur

|  |  |
| --- | --- |
| Partie | Informations de la partie (nom, table de jeu, nombre de joueurs, type de partie) |

Classe MessageResultatCreerPartie, Serveur 🡪 Client

|  |  |
| --- | --- |
| Int | Numéro de la partie créée. |

La partie est alors publiée sur le serveur du jeu afin que des joueurs puissent la rejoindre. Le joueur ayant créé la partie peut en tout temps modifier les informations de la partie.

Classe MessageModifierPartie, Client 🡪 Serveur

|  |  |
| --- | --- |
| Partie | Informations de la partie (numéro, nom, table de jeu, nombre de joueurs) |

Le joueur ayant créé la partie peut débuter le jeu lorsqu’un nombre suffisant de joueurs est prêt :

Classe MessageDemandeDebuterPartie, Client 🡪 Serveur

|  |  |
| --- | --- |
| Int | Numéro de la partie |

## Afficher une liste des parties en attente de joueurs

Un joueur peut afficher une liste des parties en attente de joueurs.

Classe MessageAccederLobby, Client 🡪 Serveur

|  |
| --- |
| *Classe vide.* |

Classe MessageActualiserLobby, Serveur 🡪 Client

|  |  |
| --- | --- |
| Partie[] | Informations des parties en attente de joueurs. |

## Rejoindre une partie

Un joueur peut rejoindre une partie en attente de joueurs.

Classe MessageRejoindrePartie, Client 🡪 Serveur

|  |  |
| --- | --- |
| String | Nom de la partie à rejoindre. |
| TypeDePartie | Permet de savoir de quel type de partie le client veut rejoindre (partie rapide ou tournoi). Ce champ est utilisé pour savoir quelle partie aléatoire choisir. |

Le message suivant est envoyé au client lorsqu’il rejoint une partie. Ce message est aussi envoyé à tous les joueurs de la partie chaque fois que ses propriétés sont modifiées où que de nouveaux joueurs la rejoignent.

Classe MessageDetailsPartie, Serveur 🡪 Client

|  |  |
| --- | --- |
| Partie | Détails de la partie |
| ProfilJoueur[] | Profils des joueurs associés à la partie |
| AvatarJoueur[] | Avatars des joueurs associés à la partie |
| Bool[] | Les joueurs qui sont prêt à commencer la partie |

Lorsque le créateur de la partie a indiqué au serveur qu’il était prêt à débuter la partie, tous les joueurs reçoivent ce message :

Classe MessageDebuterPartie, Serveur 🡪 Client

|  |  |
| --- | --- |
| Int | Numéro de port UDP à utiliser pour le déroulement de la partie |
| String | Description de la table de jeu au format XML. |
| Int | Nombre de joueur dans la partie |
| String | Le design du maillet |
| Int | Pointage pour synchroniser les spectateurs |
| Int | Jauge de tir puissant pour synchroniser les spectateurs |
| Bool | Indique si la partie utilise le souffle |

## Déroulement d’une partie

Des paquets UDP sont échangés en continu entre le serveur et les clients durant toute la durée de la partie. Ces paquets sont échangés sur une nouvelle connexion créée avec le port dédié transmis par le serveur au client via le message MessageDebuterPartie.

Serveur 🡪 Client

|  |  |
| --- | --- |
| Double [] | Position des maillets |
| Double [] | Destination des maillets |
| Double | Position de la rondelle |
| Enum Identifiant du son à jouer | Jouer les différents sons |

Client 🡪 Serveur

|  |  |
| --- | --- |
| Double | Destination du maillet |

Ce message est envoyé sur la connexion TCP principale chaque fois qu’un but est compté :

Classe MessagePointagePartie, Serveur 🡪 Client

|  |  |
| --- | --- |
| Int | Pointage de l’équipe 1 |
| Int | Pointage de l’équipe 2 |

Ce message est envoyé sur la connexion TCP principale chaque fois qu’un client change la musique :

Classe MessageChangerMusique, Client 🡪 Serveur

|  |  |
| --- | --- |
| String | Nom de la chanson |
| Int | Index de la chanson |

## Déroulement d’un tournoi

Un tournoi est créé en suivant la même procédure que pour créer une partie rapide. Au début d’une partie d’un tournoi, les joueurs impliqués reçoivent le message DebuterPartie décrit à la section 2.4. Les messages échangés au cours de la partie sont les mêmes que ceux décrits à la section 2.5. À la fin d’une partie d’un tournoi, les joueurs reçoivent le message suivant afin de pouvoir afficher le classement :

Classe MessageSuiviTournoi, Serveur 🡪 Client

|  |  |
| --- | --- |
| Partie[] | Résultats des parties du tournoi |

## Clavardage

En mode jeu en ligne, l’utilisateur peut envoyer et recevoir des messages de clavardage.

Classe MessageEnvoiChat, Client 🡪 Serveur

|  |  |
| --- | --- |
| String | Texte du message |

Classe MessageReceptionChat, Client 🡪 Serveur

|  |  |
| --- | --- |
| String | Texte du message |
| String | Nom de l’expéditeur |

## Modification du profil

Classe MessageObtenirImagesAvatar, Client 🡪 Serveur

|  |  |
| --- | --- |
| Vide |  |

Classe MessageResultatObtenirImagesAvatar, Serveur 🡪 Client

|  |  |
| --- | --- |
| ImageAvatar[] | Toutes les images d’avatar disponible dans la base de données |

Classe MessageObtenirImagesMaillet, Client 🡪 Serveur

|  |  |
| --- | --- |
| Vide |  |

Classe MessageResultatObtenirImagesMaillet, Serveur 🡪 Client

|  |  |
| --- | --- |
| ImageMaillet[] | Toutes les images de maillet disponible dans la base de données |

Classe MessageObtenirProfil, Client 🡪 Serveur

|  |  |
| --- | --- |
| String | Nom d’utilisateur du profil demandé |

Classe MessageResultatObtenirProfil, Serveur 🡪 Client

|  |  |
| --- | --- |
| Profil | Informations du profil demandé |
| Avatar | Information sur l’avatar de l’utilisateur |
| String | Design du maillet de l’utilisateur |

Le client peut modifier son profil et sauvegarder ses modifications sur le serveur :

Classe MessageSauvegarderProfil, Client 🡪 Serveur

|  |  |
| --- | --- |
| Profil | Profil modifié |
| Avatar | Avatar modifié |
| String | Maillet modifié |

Classe MessageResultatSauvegarderProfil, Serveur 🡪 Client

|  |  |
| --- | --- |
| Bool | Indique si le profil a été sauvegardé correctement. |
| String | Message d’erreur en cas d’erreur. |

# Communication et description des paquets client léger-serveur

Le serveur et les clients légers communiquent via une connexion TCP. Le serveur a un port dédié pour recevoir les nouvelles connexions provenant des clients légers.

Chaque message débute par un entier (4 octets) indiquant le type du message. Puis, les données du message sont transmises en suivant les règles suivantes pour chaque type :

* String : 4 octets indiquant la taille de la chaîne, suivie de la chaîne encodée en Unicode.
* Bool : 1 octet
* Int : 4 octets
* Image : 4 octets indiquant la taille de l’image, suivie du contenu de l’image au format PNG.

## Authentification

Client 🡪 Serveur

|  |  |
| --- | --- |
| Type de paquet (4 octets) | Authentification (valeur : 1) |
| Bool | Indique si l’utilisateur souhaite se connecter anonymement. Dans ce cas, les champs nom d’utilisateur et mot de passe seront vides. |
| String | Nom d’utilisateur |
| String | Mot de passe encrypté |

Serveur 🡪 Client

|  |  |
| --- | --- |
| Type de paquet (4 octets) | Résultat de l’authentification (valeur : 2) |
| Bool | Indique si l’authentification a réussi. |
| String | Message d’erreur, s’il y a lieu. |

## Publication d’une table sur le serveur

Client 🡪 Serveur

|  |  |
| --- | --- |
| Type de paquet (4 octets) | Publication de table (valeur : 7) |
| String | Nom de la table |
| String | Code XML de la table |
| Byte | Vignette d’aperçu de la table. |

Serveur 🡪 Client

|  |  |
| --- | --- |
| Type de paquet (4 octets) | Résultat de la publication de table (valeur : 8) |
| Bool | Indique si la table a été publiée correctement |
| String | Message d’erreur si applicable. Vide sinon. |