Projet Système d'exploitation

Sherlock 13 : Jeu réseau avec interface graphique

Quentin Deschamps

Mai 2020

Table des matières

1 Introduction

Sherlock 13 est un jeu de déduction pour 2 à 4 joueurs dans lequel les joueurs incarnent des détectives tentant de démasquer un criminel. En s'aidant des indices qu'ils ont en main et qu'ils récoltent au fur et à mesure du jeu, ils doivent innocenter les suspects jusqu'à ce qu'il n'en reste qu'un : le coupable. Vous pouvez consulter les règles du jeu sur ce **lien**.

L'objectif de ce projet est de reproduire ce jeu en réseau pour 4 joueurs avec une interface graphique en langage C. Il faut donc créer un serveur et un client ainsi qu'une interface graphique pour le client.

Ce rapport explique d'abord comment lancer le serveur et les clients. Ensuite, il montre comment utiliser l'interface graphique du client. Les extensions apportées au jeu notamment sont décrites. Enfin, nous expliquerons en détail le fonctionnement du programme, en particulier la communication entre le serveur et les clients.

2 Lancement

Cette partie décrit les différentes étapes pour lancer le jeu Sherlock 13. Le projet est composé de deux fichiers de code source : **server.c** et **sh13.c**. Ils correspondent respectivement au serveur et au client du jeu.

1. **Compilation** : la première étape consiste à compiler les deux fichiers. Pour cela, il suffit d'exécuter le fichier **cmd.sh**. En étant dans le répertoire du projet, voici la commande :

   ```
   ./cmd.sh
   ```

   Les fichiers exécutables **server** et **sh13** sont alors créés.

2. **Lancement du serveur** : il faut ensuite lancer l'exécution du fichier **server**. Le numéro de port doit être indiqué. Voici la ligne de commande :

   ```
   ./server <Port server>
   ```

3. **Lancement des clients** : une fois le serveur lancé, on démarre les clients. Le jeu a besoin de 4 clients connectés pour lancer la partie. Le fichier **sh13** doit être exécuté comme ceci :

   ```
   ./sh13 <IP address server> <Port server> <IP address client>
                        <Port client> <Name>
   ```

   Pour obtenir l'adresse IP de votre ordinateur, vous pouvez tapez la commande suivante ;

   ```
   hostname -I
   ```

   Si le serveur et les clients sont exécutés sur le même ordinateur, il suffit de spécifier `localhost` pour les adresses IP.

3 Utilisation de l'interface

Dans cette partie, nous allons montrer comment utiliser l'interface graphique du jeu. Celle-ci est réalisée à l'aide de la librairie graphique **SDL2**[1] pour le langage C.

3.1 Présentation

Voici l'interface du jeu Sherlock 13 :

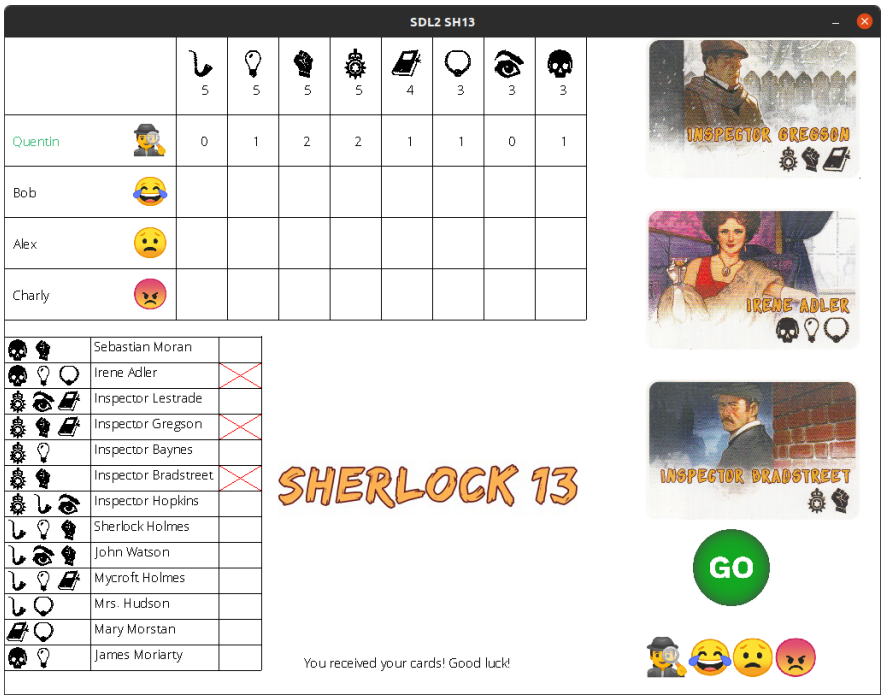

FIGURE 1 – Interface

Elle est composée de différentes parties :

— **Table des symboles** (en haut) : tableau récapitulant les symboles possédés par chaque joueur. Ce tableau est complété au fur et à mesure de la partie.

— **Cartes personnages possédées** (à droite) : ces personnages ne peuvent donc pas être le coupable.

— **Table des suspects** (à gauche) : tableau listant tous les suspects avec leurs symboles. Vous pouvez ajouter une croix pour les personnages que vous pensez ne pas être coupables en cliquant sur la case correspondante.

1. Lien de la documentation

— **Bandeau d'informations** (en bas) : permet d'obtenir des informations sur le jeu (voir la partie des extensions)
— **Bouton Go** (en bas à droite) : permet de jouer lorsque c'est son tour
— **Émojis** (en bas à droite) : permet de chambrer les adversaires (voir la partie des extensions)

3.2 Connexion

Une fois le client du jeu lancé, vous arrivez sur l'interface suivante :

FIGURE 2 – Connexion

Tout d'abord, pour vous connecter au serveur, il suffit de cliquer sur le bouton **Connect** en haut à gauche. Si l'opération a réussi, vous devez voir le message *"You are connected ! Waiting for other players..."* dans le bandeau d'informations et le nom des joueurs actuellement connectés au serveur dans la table des symboles. La partie est lancée quand 4 joueurs sont connectés.

3.3 Jeu

Lorsque la partie est lancée, la fenêtre du jeu s'apparente à l'image ci-dessous.

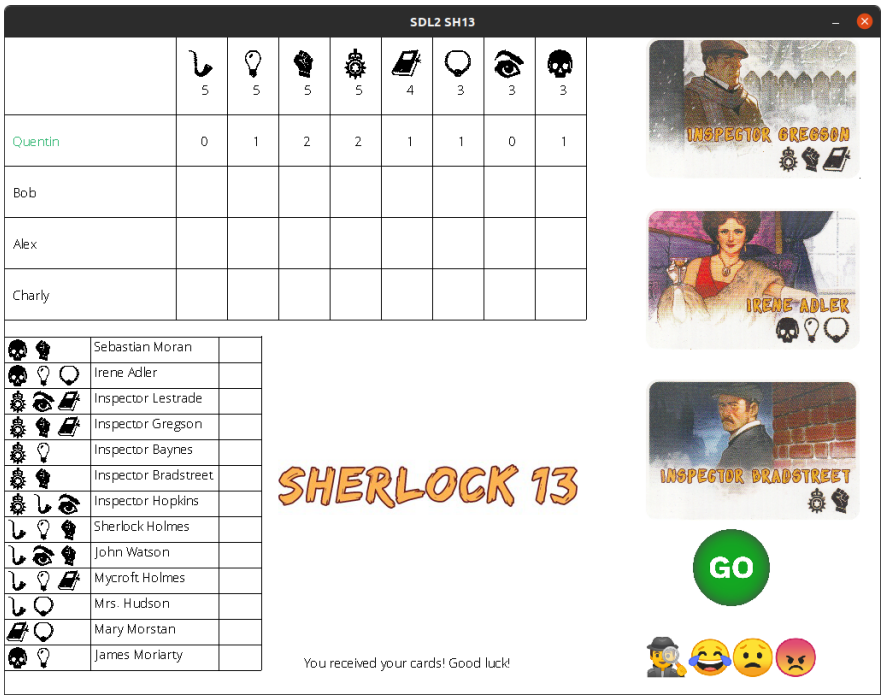

FIGURE 3 – Début de partie

Vous voyez à droite vos 3 cartes personnages. Vous pouvez aussi constater que la ligne de la table des symboles correspondant à votre nom est complétée par les symboles de vos 3 cartes.

C'est alors au joueur 0 de commencer (celui le plus haut dans le tableau). Le nom du joueur courant est coloré en vert dans la table des symboles (visible par tous les joueurs). Lorsque c'est votre tour, le bouton **Go** apparaît en bas à droite de l'écran. Ce bouton n'est visible que pour le joueur courant. Vous disposez alors de trois actions possibles [2] :

1. **Enquête 1** : choisissez un symbole en cliquant sur son image en haut de la table des symboles et demandez aux autres joueurs s'ils ont ce symbole (sans donner le nombre) en cliquant sur le bouton Go. Tous les joueurs reçoivent les réponses. Si un joueur a répondu posséder le symbole, alors une étoile apparaît dans la table à la case correspondante. Sinon, 0 est affiché.
2. **Enquête 2** : choisissez un symbole comme pour l'action précédente et un joueur en cliquant sur son nom à gauche de la table des symboles puis validez avec le bouton Go. Le joueur désigné doit alors dire le nombre exact de symboles de ce type qu'il possède. Tous les joueurs reçoivent sa

2. Nom des actions d'après la règle officielle du jeu

réponse. Le nombre exact de symboles est alors reporté dans la table à la case correspondante.

3. **Accusation** : sélectionnez un personnage en cliquant sur son nom dans la table des suspects et cliquez sur Go pour l'accuser. Deux scénarios sont alors possibles :
   — **Si vous avez trouvé le bon coupable**, alors vous avez gagné et la partie se termine. Tous les joueurs sont informés du gagnant et du coupable par le bandeau d'informations. Le nom du coupable apparaît en vert dans la table des suspects.
   — **Si le personnage désigné est innocent**, alors vous êtes éliminé de la partie. Votre nom apparaît en rouge dans la table des symboles et votre tour sera passé jusqu'à la fin de la partie. Les autres joueurs sont informés du personnage que vous avez accusé à tort avec le bandeau d'informations. La croix correspondant à ce personnage est automatiquement dessinée pour tous les joueurs dans la table des suspects.

La partie peut aussi se terminer si tous les joueurs sauf un ont raté leur accusation. Dans ce cas, le joueur restant remporte la partie. Voici un exemple de fin de partie :

FIGURE 4 – Fin de partie

On peut notamment voir que Alex a été éliminé au cours de cette partie car

son nom est en rouge. De plus, le bandeau d'informations indique que Charly a gagné la partie et que Mycroft Holmes était le coupable. Son nom apparaît en vert dans la table des suspects. De plus, le bouton *"Play Again"* est apparu en haut à gauche (voir la partie des extensions).

3.4 Extensions

Cette partie montre les différentes améliorations apportées au jeu de base.

3.4.1 Améliorations graphiques

Quelques modifications et améliorations graphiques ont été apportées au jeu. Tout d'abord, le logo du jeu Sherlock 13 a été rajouté au centre de l'écran. Par conséquent, le bouton Go a été repositionné sous les images des cartes en bas à droite. De plus, l'image de ce bouton a été modifiée (voir figure 1).

Ensuite, la coloration des noms a été ajoutée comme décrit précédemment. Le nom du joueur courant s'affiche en vert, et celui des joueurs éliminés en rouge (les autres en noir). Enfin, quand la partie est terminée, le nom du coupable est écrit en vert dans la table des personnages (voir figure 4).

Enfin, le bandeau d'informations a été rajouté en bas de l'écran. Il permet d'obtenir des informations sur le jeu lors des cas suivants, avec exemples à l'appui :

- **Arrivée sur le jeu** (message de bienvenue) :
  "Hello Quentin ! Welcome to Sherlock 13 !"
- **Connexion** :
  "You are connected ! Waiting for other players..."
- **Distribution des cartes et début d'une partie** :
  "You received your cards ! Good luck !"
- **Élimination d'un joueur** :
  "Alex is eliminated ! Irene Adler is innocent !"
- **Victoire d'un joueur** :
  "Charly wins ! The guilty person was Mycroft Holmes !"
- **Demande de rejouer d'un joueur** (voir partie suivante) :
  "Quentin wants to replay ! (1/4)"

3.4.2 Bouton Replay

À la fin de la première partie, le bouton **Replay** apparaît au même emplacement que le bouton Connect (voir figure 4). Ce bouton permet de demander une nouvelle partie. Lorsqu'un joueur clique dessus, le bouton disparaît et sa fenêtre se réinitialise. Les autres joueurs sont alors informés de la demande via le bandeau d'informations. Le nom du joueur en question ainsi que le nombre de joueurs ayant demandé une nouvelle partie sont indiqués (comme dans l'exemple

précédent). Si les quatre joueurs ont cliqué sur le bouton, alors le serveur envoie les cartes de la nouvelle partie aux joueurs et la partie commence.

3.4.3 Émojis

Lors d'un jeu, il est toujours très amusant de chambrer ses adversaires. Cela est possible pour notre jeu grâce aux émojis ! Quatre émojis ont été ajoutés sur la fenêtre en bas à droite. Ils apparaissent au début de la première partie. Si vous cliquez sur l'un d'entre eux, celui-ci apparaît à droite de votre nom dans la table des symboles. Il est visible par tous les joueurs. Vous pouvez en changer à tout moment, même si ce n'est pas votre tour, et même si la partie est terminée (voir figure 1).

4 Fonctionnement

Dans cette partie, nous allons voir le fonctionnement du programme. Nous verrons comment le serveur communique avec les clients lors des différents évènements rencontrés.

4.1 Description du serveur

Nous allons ici décrire le fichier **server.c**. Tout d'abord, les lignes 12 à 22 correspondent aux inclusions des interfaces des bibliothèques nécessaires à la mise en place du serveur.

Ensuite, les lignes 24 à 28 définissent la structure **client** qui est composée de trois champs : l'adresse IP, le numéro de port et le nom de l'utilisateur. On peut voir que le tableau **tcpClients** est crée, contenant 4 instances de cette structure. Ce tableau contient donc les informations utiles pour les 4 clients. Celui-ci sera modifié à chaque nouvelle connexion d'un client.

4.1.1 Variables globales

Les lignes 29 à 50 définissent différentes variables globales :
- **nbClients** : nombre de clients connectés au serveur. La partie peut démarrer quand ce nombre vaut 4.
- **fsmServer** : définit l'état du serveur. Deux états possibles : 0 pour "attente de connexions", 1 pour "jeu" (quand 4 joueurs connectés).
- **deck** : liste des 13 cartes (représentées par des entiers)
- **tableCartes** : représente la table des symboles. Tableau de 4x8 où les 4 lignes correspondent aux 4 joueurs et les 8 colonnes aux 8 symboles.
- **nomcartes** : noms des suspects
- **joueurCourant** : numéro du joueur courant

— **eliminated** : indique les joueur éliminés. Tableau de 4 entiers correspondant aux 4 joueurs : 1 si éliminé, 0 sinon.
— **nbPlayersRemaining** : nombre de joueurs non éliminés
— **nbReplayPlayers** : nombre de joueurs demandant à rejouer une partie

4.1.2 Procédures et fonctions

Les lignes 52 à 245 définissent plusieurs procédures et fonctions utiles pour le jeu :

— **error** : gère les erreurs du programme
— **melangerDeck** : mélange le deck. La technique utilisée est la suivante : on échange deux éléments pris aux hasard dans le tableau **deck** et on répète cette opération 1000 fois.
— **createTable** : initialise le tableau des symboles. Cette fonction parcourt le tableau **deck** et remplit les cases du tableau **tableCartes**. Chaque joueur possède 3 cartes du tableau **deck** :
  — Le joueur 0 possède les cartes d'indice 0, 1, 2
  — Le joueur 1 possède les cartes d'indice 3, 4, 5
  — Le joueur 2 possède les cartes d'indice 6, 7, 8
  — Le joueur 3 possède les cartes d'indice 9, 10, 11
  — Le coupable est la carte d'indice 12
— **printDeck** : affiche le paquet des cartes, correspondant au tableau **deck**
— **printClients** : affiche les informations sur les clients, correspondant au tableau **tcpClients**
— **findClientByName** : retourne à partir du nom du joueur l'indice correspondant (son numéro)
— **sendMessageToClient** : envoie un message à un client. Cette procédure prend en argument l'adresse IP et le numéro de port du client ainsi que le message. Elle crée une *socket* (ligne 194), puis la connecte au serveur (ligne 205) et envoie le message (ligne 211) avant de fermer la *socket* (ligne 213).
— **broadcastMessage** : envoie un message à tous les clients, en répétant la procédure précédente pour chacun des clients
— **nextPlayer** : retourne le numéro du prochain joueur qui doit jouer. Cette fonction prend en argument le numéro du joueur courant. Le prochain joueur ne doit pas être éliminé pour pouvoir jouer (ligne 229).
— **initGame** : initialise les données pour une partie. Cette procédure :
  — Mélange le deck (ligne 238)
  — Crée le tableau des symboles (ligne 239)
  — Initialise le joueur courant (0), le nombre de joueurs non éliminés (4) et le tableau des éliminés (tous à 0)

4.1.3 Main

Le *main* du fichier **server.c** se déroule comme suit :

1. Initialisation de *rand* pour le mélange du deck (ligne 248)

2. Initialisation de différentes variables. À noter :
   — **buffer** : chaîne de caractères contenant les messages reçus par le serveur
   — **reply** : chaîne de caractères contenant les messages envoyés par le serveur
3. Vérifie si le numéro de port est donné lors du lancement du serveur (ligne 262)
4. Création de la *socket* (ligne 266)
5. Associe la socket au port indiqué (ligne 274)
6. Écoute les connexions (ligne 276)
7. Initialise une partie (ligne 279)
8. Entre dans la boucle infinie et attend la connexion d'un client (ligne 288)

À ce moment là, le programme se bloque jusqu'à ce qu'un client ce connecte. Lorsqu'un client arrive, le serveur l'accepte (si c'est possible). Le serveur va alors attendre de recevoir des messages de la part du client.

Lorsque le nombre de clients est inférieur à 4, alors le serveur doit attendre des connexions avant de lancer la partie. Son état défini par la variable **fsmServer** vaut 0. Quand le nombre de clients est égal à 4, le serveur est en état de jeu : **fsmServer** vaut 1. Le serveur répond alors aux messages qui sont envoyés par les clients. Les messages sont décrits dans la suite de ce rapport.

4.2 Description du client

Nous allons maintenant décrire le fichier **sh13.c**. Tout d'abord, on peut voir pour les lignes 12 à 22 l'inclusion des interfaces des bibliothèques utiles pour le client. Les lignes 12 à 14 correspondent à des parties de la bibliothèque graphique **SDL2**. Celle-ci est utilisée pour l'interface graphique du jeu.

4.2.1 Variables globales

Les lignes 24 à 69 définissent différentes variables globales :
— **tread_server_tcp_id** : identifiant du thead serveur tcp
— **gbuffer** : chaîne de caractères contenant les messages reçus de la part du serveur
— **gServerIpAddress** et **gServerPort** : adresse IP et numéro de port du serveur
— **gClientIpAddress** et **gClientPort** : adresse IP et numéro de port du client
— **gName** : nom du joueur
— **gNames** : noms des 4 joueurs
— **gId** : id du joueur (son numéro)

— **joueurSel**, **objetSel**, **guiltSel** : numéros respectifs du joueur, du symbole et du suspect sélectionnés. Ces variables valent -1 si aucun n'est sélectionné.
— **guiltGuess** : tableau gérant les croix à droite des suspects
— **tableCartes** : représente le tableau des symboles
— **b** : tableau des 3 cartes possédées
— **goEnabled**, **connectEnabled**, **replayEnabled**, **emojiEnabled** : autorisent l'affichage respectif des boutons Go, Connect, Replay et des émojis.
— **emojiPlayers** : tableau contenant l'émoji actuel de chacun des 4 joueurs
— **guilty** : numéro du coupable (connu à la fin de la partie)
— **info** : chaîne de caractères contenant le message affiché par le bandeau d'informations
— **nbobjets** : nombre total de chaque symbole
— **nbnoms** : noms des suspects
— **synchro** : indique la réception d'un message de la part du serveur
Les autres variables sont identiques à celles du serveur.

4.2.2 Procédures et fonctions

Sur les lignes 71 à 189, plusieurs procédures et fonctions sont définies :
— **fn_serveur_tcp** : gère le thread du client avec le serveur tcp. Cette procédure crée une *socket* (ligne 77), l'associe au port (ligne 87), écoute les connexions (ligne 91) et accepte les connexions dans sa boucle infinie (ligne 94).
— **sendMessageToServer** : envoie un message au serveur
— **initEmoji** : initialise le tableau des émojis (-1 pour aucun émoji)
— **initGame** : initialise une partie
— **resetSel** : réinitialise les variables de sélections
— **initReplay** : initialise la demande de rejouer

4.2.3 Main

Le *main* du fichier **sh13.c** se déroule comme suit :

1. Initialisation de différentes variables :
   — **quit** : vaut 1 si la fenêtre est quittée, 0 sinon
   — **mx** et **my** : position de la souris (l'origine du repère correspond au coin supérieur gauche)
2. Vérifie si les informations nécessaires ont été données lors du lancement (voir partie lancement) (ligne 201)
3. Initialisation de la fenêtre et de la police d'écriture (lignes 212-213)
4. Création de la fenêtre (lignes 215 à 219)
5. Création du *renderer*, permettant d'actualiser la fenêtre de jeu (ligne 221)
6. Création des couleurs (lignes 224 à 226) et des surfaces pour les images (lignes 229 à 262)

7. Initialisations pour le jeu (lignes 264 à 280)
8. Création des textures (lignes 283 à 297) et de la police (ligne 300)
9. Création du thread serveur tcp (lignes 304 à 306)
10. Initialisation du bandeau des informations (ligne 309)
11. Entre dans la boucle principale (ligne 311)

Une fois le programme dans cette boucle, celui-ci surveille si des évènements se produisent et agit en conséquence. Il peut recevoir deux types d'évènement :

1. **Évènement graphique** : quand un évènement graphique arrive, le programme passe le test ligne 312 et regarde la nature de l'évènement. Si c'est la fermeture de la fenêtre, alors le programme sort de la boucle, ferme la fenêtre et s'arrête. Si l'utilisateur a cliqué avec la souris sur un endroit de la fenêtre, alors le programme étudie si cet endroit correspond à une action. Si c'est le cas, le programme actualise les variables de sélections ou envoie un message au serveur.
2. **Évènement réseau** : pour les évènements de ce type, c'est le test ligne 389 que le programme passe, avec la variable **synchro**. À ce moment là, le programme étudie le premier caractère du message envoyé par le serveur. Cela correspond à l'identifiant du message. Il agit ensuite en conséquence en mettant à jour des variables décrites précédemment.

Après avoir traité un évènement, le programme dessine la fenêtre et l'actualise (lignes 451 à 852). Vous pouvez regarder les commentaires sur le programme pour voir chaque partie de la fenêtre.

Une fois la fenêtre quittée, les textures et surfaces sont détruites (lignes 869 à 878).

4.3 Messages

Comme vu précédemment, des messages sont échangés entre le serveur et les clients. Ces messages sont des chaînes de caractères. Chaque type de message est identifié par une lettre, correspondant au premier caractère de la chaîne.

Dans cette partie, les messages envoyés du serveur vers les clients et des clients vers le serveur sont listés. Vous trouverez enfin les échanges qui ont lieu avec ces messages.

4.3.1 Messages d'un client vers le serveur

Le message est écrit par le client dans la variable **sendBuffer** avec la fonction **sprintf**, puis est envoyé au serveur par la procédure **sendMessageToServer**.

Pour décoder le message, le serveur utilise la fonction **sscanf** sur la variable **buffer** qui contient le message reçu. Voici ci-dessous la liste des messages envoyés par un client vers le serveur.

Type	Données	Utilisation
C : Connexion	Adresse IP, port, nom	Connexion avec le serveur
R : Replay	ID	Demande de rejouer
H : Head	ID, n° émoji	Envoie un émoji
G : Guilt	ID, n° suspect	Fait une accusation
O : Others	ID, n° symbole	Action d'enquête 1
S : Solo	ID, n° symbole	Action d'enquête 2

TABLE 1 – Messages d'un client vers le serveur

Notation : ID correspond au numéro d'un joueur.

4.3.2 Messages du serveur vers les clients

Le message est écrit par le serveur dans la variable **reply**, puis est envoyé à un client par la procédure **sendMessageToClient**. Pour envoyer un message à tous les clients, le serveur utilise **broadcastMessage**.

Pour décoder le message, les clients utilise la fonction **sscanf** sur la variable **gbuffer** qui contient le message reçu. Voici ci-dessous la liste des messages envoyés par le serveur vers les clients.

Type	Données	Information envoyée
I : ID	ID	Identifiant d'un joueur
L : List	Noms des joueurs (4)	Liste des noms des joueurs
D : Deck	N° suspects (3)	Les 3 cartes d'un joueur
M	ID	Numéro du joueur courant
V : Value	ID, n° symbole, valeur	Valeur de la table des symboles
E : Eliminated	ID, n° suspect	ID de l'éliminé et l'innocent
W : Winner	ID, n° suspect	ID du gagnant et le coupable
H : Head	ID, n° émoji	Un émoji
R : Replay	ID	Demande de replay

TABLE 2 – Messages du serveur vers les clients

4.3.3 Échanges

Enfin, nous allons voir les échanges de messages qui ont lieu lors des différents évènements possibles.

1. **Connexion** : lorsqu'un client se connecte au serveur en appuyant sur le bouton Connect, il envoie un message de type C au serveur pour lui communiquer ses informations. Le serveur répond alors en envoyant à ce client son numéro par un message de type I. Il envoie ensuite à tous les joueurs connectés la liste des joueurs par un message de type L pour qu'ils mettent à jour les noms des joueurs.

2. **Lancement de partie** : si le client qui vient de se connecter monte le nombre de clients connectés à 4, alors la partie peut commencer. Le serveur envoie à tous les joueurs leurs cartes par un message de type D, et envoie la ligne de la table des symboles qui leur correspond avec des messages de type V. Le serveur envoie enfin à tout le monde le numéro du joueur courant (0) à tous les joueurs. L'affichage graphique des clients se met alors à jour.
3. **Émoji** : lorsqu'un joueur sélectionne un émoji, un message de type H est envoyé au serveur. Celui-ci transfère alors ce message à tous les joueurs pour qu'ils voient tous l'émoji.
4. **Action Enquête** : lorsqu'un joueur réalise l'action "Enquête 1" ou "Enquête 2", un message de type O ou de type S respectivement est envoyé au serveur. Celui-ci renvoie les réponses associées à la demande à tous les joueurs par des messages de type V. La table des symboles des joueurs est alors mise à jour. Le serveur envoie enfin le nouveau numéro du joueur courant par un message de type M.
5. **Accusation** : lorsqu'un joueur fait une accusation, il envoie un message de type G au serveur. Celui-ci regarde alors si le suspect envoyé est bien coupable. Si c'est le cas, le serveur envoie aux clients un message de type W. En revanche, si le suspect est innocent, alors le serveur envoie un message de type E à tout le monde qui indique le joueur éliminé et le suspect innocent. Dans le cas où il n'y a plus qu'un seul joueur non éliminé, le serveur envoie un message de type W comme précédemment. Si ce n'est pas le cas, alors la partie continue et le serveur envoie le nouveau joueur courant par un message de type M.
6. **Replay** : lorsqu'une partie est terminée, si un joueur appuie sur le bouton Replay, il envoie un message de type R au serveur. Celui-ci comptabilise cette demande en incrémentant le compteur de joueurs voulant rejouer et transfère le message aux autres joueurs. Si les 4 joueurs veulent rejouer, alors le serveur refait de nouveau la procédure de lancement de partie.

Vous pouvez regarder en détail le programme et ses commentaires pour voir les variables mise à jour par tous ces messages.

Table des figures

Liste des tableaux