RAPPORT PRÉLIMINAIRE – PROJET SR

PAR EDDY DRUET, CLÉMENT GILI, GROUPE TP 02
LICENCE 3 - INFORMATIQUE

STRUCTURE GENERALE

Le projet sera constitué de trois processus distincts (trois exécutables différents) qui communiqueront via une connexion socket avec le serveur de jeu.

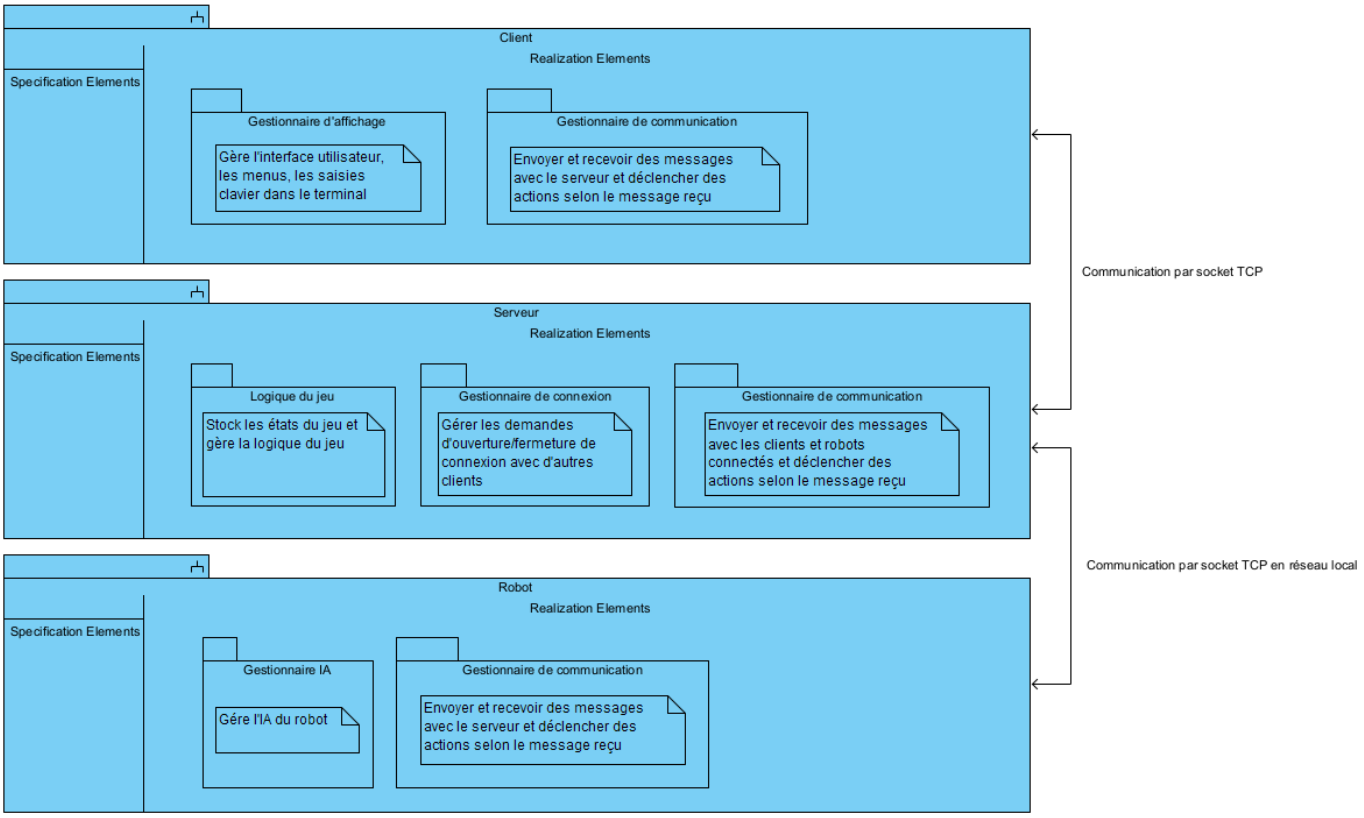

Figure 1 : Architecture du projet et des trois programmes.

Le(s) processus « robot » sera/seront automatiquement exécuté(s) en arrière-plan par le serveur au moment du lancement de la partie, le serveur fournissant l'adresse IP et le port de connexion en paramètres passés au programme « robot ». Les processus « robot » se terminent lorsque le serveur est fermé, ou que la partie est terminée.

RÉALISATION DES FONCTIONNALITÉS

Ci-dessous, les méthodes de résolution des fonctionnalités demandées avec les explications associées.

Fonctionnalité	Méthode de résolution	Explication
Multijoueur	Socket.	Permet de jouer à distance, pas seulement sur la même machine, en réseau avec connexion par adresse IP.
Langage	C uniquement.	Awk n'a pas pour rôle ce type d'application, et ne permet pas de le faire. Shell est trop fastidieux et peu pratique pour réaliser une grosse application par rapport au C.
Gestionnaire de jeu (serveur)	Processus en langage C communiquant avec les clients et robots connectés.	Application et processus distinct intermédiaire entre les joueurs et gère la logique du jeu.

Joueur humain (client)	Processus en langage C communiquant avec le serveur par connexion socket.	Application et processus distinct du « serveur » et du « joueur robot » pour bien séparer les deux côtés pour une question de maintenabilité, de souplesse et de simplicité de code.
Joueur robot	Processus en langage C communiquant avec le serveur par connexion socket.	Application et processus distinct automatiquement démarré et fermé par le serveur en début et fin de partie. Cycle de vie contrôlé par le processus « serveur ». La communication est également par socket pour avoir un système de communication inter-processus universel, unique et constant, ce qui permettra d'éviter d'augmenter la complexité du code pour rien.
Communication entre processus	Envoi de messages par connexion socket.	Le format d'un message sera celui-ci : - Émetteur du message (client, serveur, robot) ; - Type du message (rôle du message) ; - Aucun, un ou plusieurs paramètres. Exemples : « CLIENT_CONNECT|Pseudonyme » « SERVER_START_GAME » « Émetteur_Type|param1|param2|param…|paramN »
Génération de statistiques	Écriture d'un fichier LaTeX à partir d'un code C dans le processus « serveur », puis conversion en PDF via un outil externe.	LaTex est similaire à un langage à balise, donc assez simple à générer à partir d'un langage de programmation comme C. La création du fichier LaTeX à la conversion en PDF sera fait dans le processus « serveur » en code C.
Classement	Écriture d'un fichier texte à partir d'un code C dans le processus « serveur » et affichage en fin de partie.	
Gestion du temps	Mesure du temps écoulé avec une fonction C comme clock().	La mesure du temps pourra servir à mesurer le temps de réaction des joueurs par exemple.

ÉTAPES ET PHASES DE JEU

PHASE DE PRÉPARATION

Lancement du serveur. Il attend la connexion de clients.

Lorsque les clients se connectent au serveur, ils rejoignent un lobby où ils peuvent spécifier leur pseudonyme et spécifier le nombre de robots dans la partie.

La partie démarre lorsque tous les joueurs sont prêts.

PHASE DE JEU

Les joueurs et robots jouent, et le serveur gère les communications et la gestion de la partie.

PHASE DE FIN DE PARTIE

La partie est terminée, les statistiques et classements sont générés. Les joueurs peuvent quitter le serveur, ou recommencer une partie. Les robots se terminent.