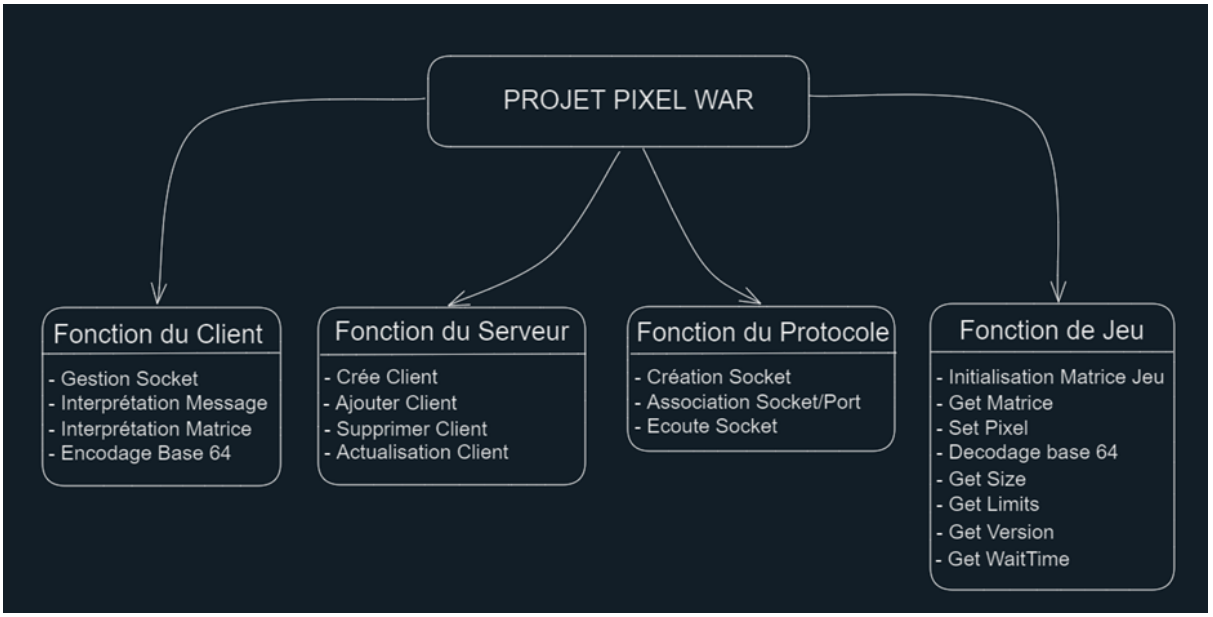
PROJET PIXEL WAR

EQUIPE

ARRAHMAIN Amin *Développeur*
DENIS Thomas *Chef de projet/Développeur*
MASSARD Rémi *Testeur*

LE PROJET

Le but de ce projet est de reproduire le jeu pixel war en se servant des notions de réseaux qui ont étés appliquées et vues en cours pour la plupart. On a donc un fichier Serveur qui reçoit des connexion et des demandes de plusieurs Clients.

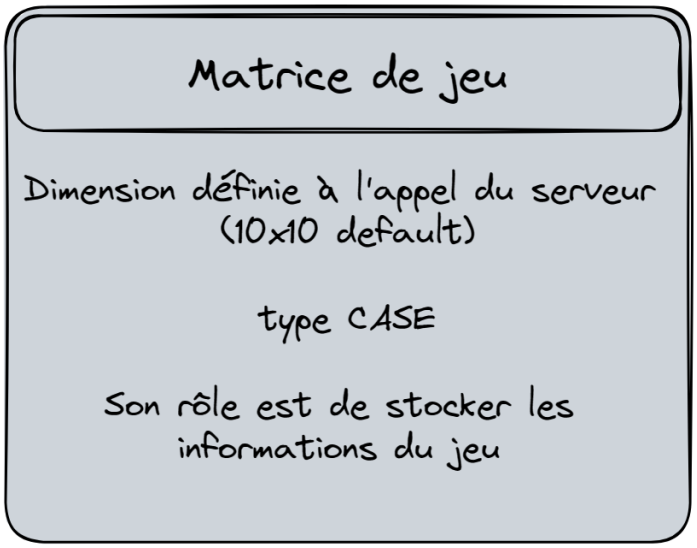


Pour un contenu détaillé sur chaque élément, aller voir dans le fichier de rendu “Projet-Reseau-S4\Doxygene” télécharger le dossier et exécuter “Doxygene\html\index.html”.

LE SERVEUR

Le serveur est lancé avec la commande adaptée et de façon facultative des paramètres supplémentaires sûr, la taille de la matrice, le port d'hébergement, le nombre de pixels max. Ensuite, on crée le socket du serveur, la mise en écoute, tout ce que l'on a fait dans le TP sur TCP. Ensuite, on ouvre une boucle d'écoute sur le poll qui va prendre un tableau mis à jour à chaque passage dans la boucle qui surveille les clients qui se connectent, se déconnectent ou envoient des messages. Chaque message est interprété et reçoit une réponse du serveur en fonction de la demande. Pour le cas du setPixel le serveur utilise une fonction pour décomposer le message, prendre la position, vérifier si elle est bonne puis prendre la couleur en base 64 et la décoder pour pouvoir la placer à la bonne position et déduire le nombre de pixels pour le client. Il y a aussi des fonctions pour gérer les clients (ajout, suppression, modification, ...).

Le serveur est composé de beaucoup d'éléments, notamment la Matrice de jeu :



Autres éléments importants, ce sont les clients qui sont utilisés pour chaque connexion.

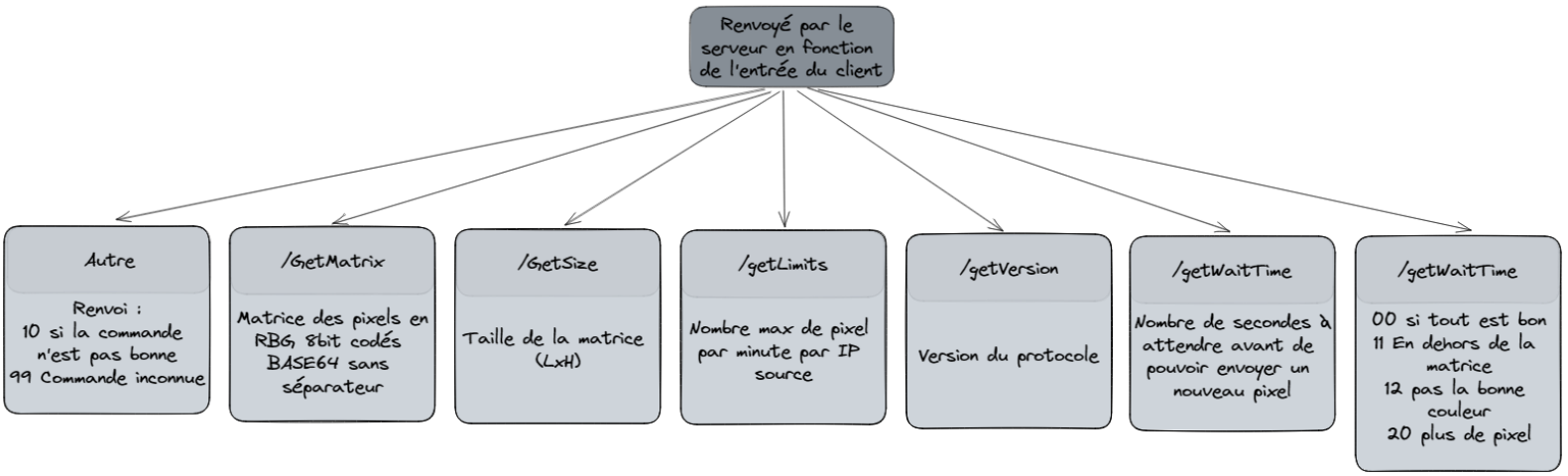
Client

Il est initialisé et associé à une nouvelle connexion

type CLIENT

Son rôle est de stocker les informations du Client

Ce que renvoi le serveur



LE CLIENT

La partie sans l’affichage va premièrement se connecter au serveur grâce à la commande de lancement, le code va récupérer en se connectant la taille de la matrice pour baser le code là-dessus. On lance une boucle infinie (jusqu'à déconnexion) d’écoute et d'envoi de message. Quand elle reçoit le message, elle va le traiter en fonction de son contenu : code d’erreur, informations à mettre en forme, reçu des commandes, et la matrice à afficher. Comme le serveur, le client va encoder et décoder la couleur.

Ncurses qui nous permet de créer une fenêtre dédiée pour afficher différents contenus