DOCUMENT D'ARCHITECTURE

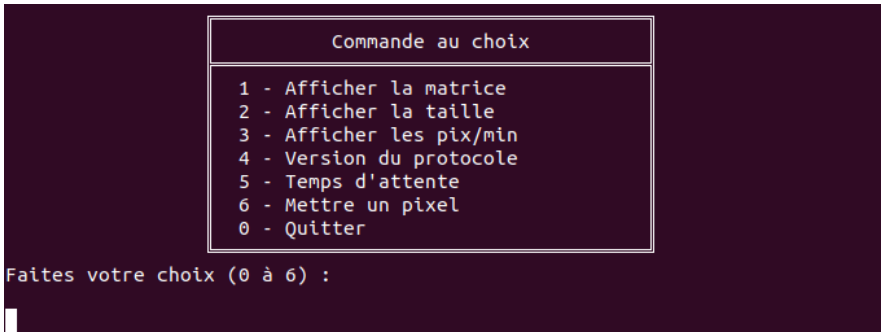
LE CLIENT

Le client est composé de 4 documents : header.h, Client.c et color_convert.c

Le header.h en plus de contenir tous les prototypes des fonctions utilisées dans tout le code mais aussi la structure « parametre » qui a la même utilité que celle du serveur mais elle permet aussi de renseigner cette fois-ci l'adresse IP du client.

Lors du lancement du client, les paramètres rentrés sont récupérés et enregistrés dans la structure paramètre grâce à la fonction paramétrage.

Lors du lancement de la connexion de notre client au serveur, ce dernier affichera dans le terminal un menu où chacune des actions demandées dans le cahier des charges sont disponibles. Ces choix sont restreints de 0 à 6 par une fonction ("sasieInt")rajouter qui permet d'avertir l'utilisateur lors d'une rentrée d'un caractère ou nombre erronés.



Les cinq premiers choix permettent d'envoyer des messages prédéfinis (les requêtes) grâce à la fonction "write", sur la socket de discussion à laquelle le client est connecté avec le serveur.

Le sixième choix requiert l'attention de l'utilisateur afin de pouvoir placer le pixel de la couleur de son choix sur la matrice du serveur grâce à la requête "/setPixel". L'action de l'utilisateur est nécessaire afin de pouvoir renseigner les paramètres de cette requête, dont la couleur du pixel à déposer.

Etant donné que le serveur attend une chaine de caractères en base 64 on ne peut pas lui envoyer n'importe quoi, mais l'utilisateur ne sait pas forcément convertir les couleurs en base 64.

L'interface du client demande donc de renseigner la couleur dans ses valeurs rgb, pour ensuite la convertir en un binaire que sera découpé en quatre paquets de six bits, chacun correspondant à un des caractères de la base64.

(Le tout est réaliser grâce à la fonction "rgb_to_string" du ficher "color_convert.c".)

Une fois la que la couleur est transcrite en base 64, sa chaine de caractère est collée au reste de la chaine de caractère contenant la requête. Puis le tout est envoyé au serveur.

Initialement, nous avions prévu que le client récupère automatiquement la matrice du serveur sans que l'utilisateur n'ai besoin de lui demander (comme sur le vrai pixel-war). Mais étant donné que la fonction "saisieInt" à un appel bloquant, nous avons eu l'idée d'utiliser une structure "pollfd" qui, contrairement au serveur ne surveille pas la socket, mais les entrées dans le terminal.

Nous avions aussi prévu de pouvoir afficher la matrice en couleur, et pour cela nous avions coder la fonction "ascii_to_rgb".

Lorsque le client formule la requête "/getMatrix", il reçoit une chaine de caractère contenant chaque couleur de la matrice codées en base 64. Afin de pouvoir afficher ces couleurs il faut pouvoir retrouver leur valeurs rgb.

Chacun de quatre caractères de la couleur en b64 correspond à un binaire de 6 bits qui collés bout à bout forme un binaire de 24 bits.

Celui-ci est redécoupé en paquet de 8 bits pour être ensuite converti en nombre entier correspondant à des valeur rgb, grâce à la fonction "binaire_toRGB".

Les valeurs retournées par la fonction "ascii-to_rgb" étaient prévues pour afficher en couleur la matrice, ce qui n'a pas été réalisé par manque de temps.

La fonction et l'affichage de la matrice en couleur n'ont donc pas pu être implémenté par manque de temps.