**Dossier du projet**



**Sommaire :**

* **Présentation du projet**

*Présentation des fonctionnalités générales du jeu*

* **Répartition du travail**

*Répartition des différents éléments du programme*

* **Outils utilisés**

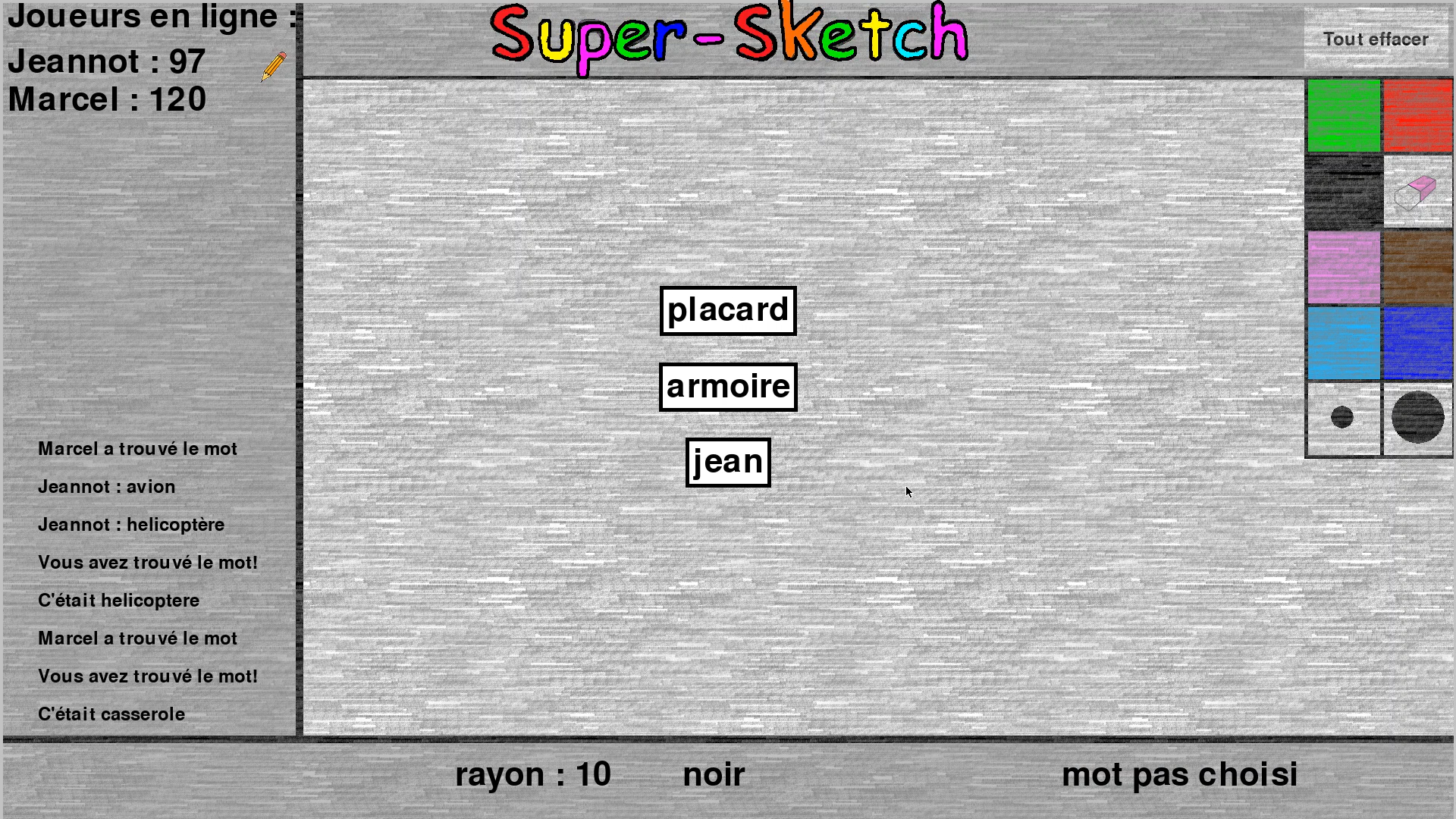
*Présentation des différents outils qui ont permis à la conception du projet*

* **Problèmes rencontrés**

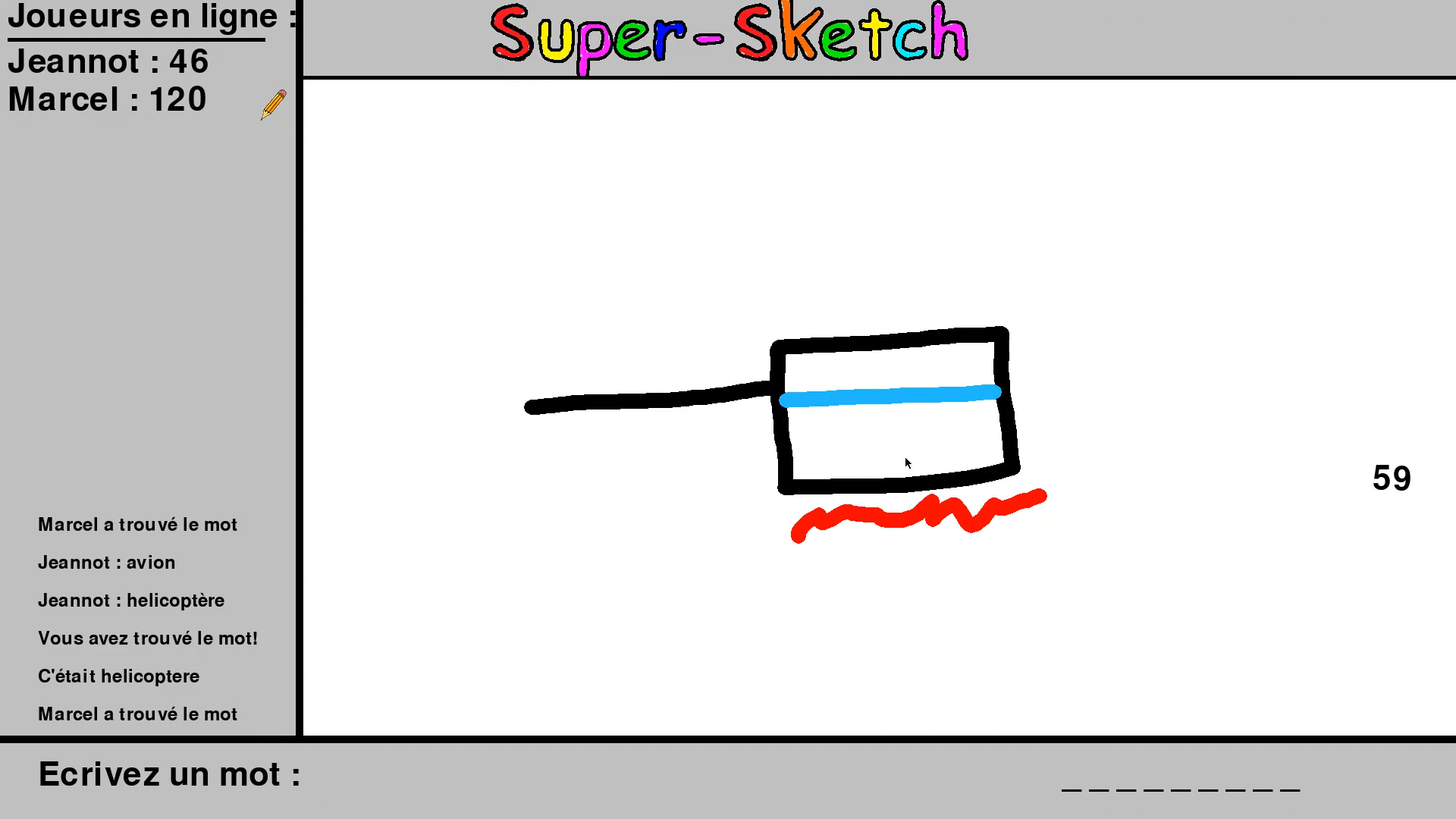
*Présentation des différents problèmes rencontrés lors de la programmation*

1. **Présentation du projet**

Super-Sketch est un jeu de dessin du style pictionnary (jeu de société). L’objectif est de réussir à faire deviner le mot que l’on a choisi à son adversaire, ou de trouver ce que l’adversaire dessine. Celui qui obtient le plus de points remporte la partie.



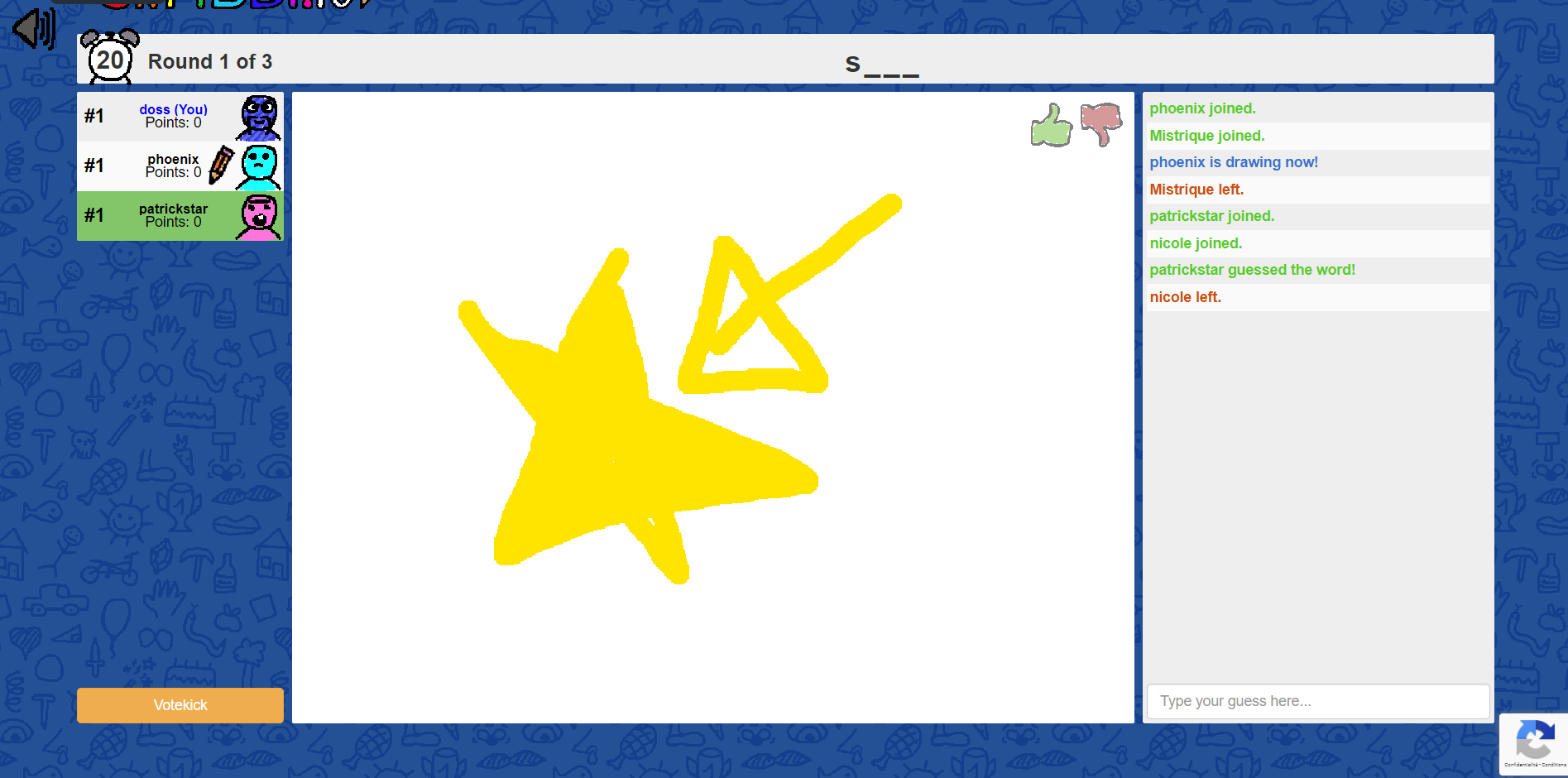
Le jeu peut se jouer en LAN (réseau local) et la découverte d’hôte est automatique, Ou bien en réseau. Dans ce dernier cas, il faudra ouvrir le port 5000 de sa box internet et transmettre son IP aux joueurs que vous voulez inviter dans votre partie.



Ce jeu est très accessible et plaira au plus grand nombre. Il y a un chat intégré, des sons, des animations, un joli design. Tout cela pour permettre la meilleure expérience utilisateur possible !

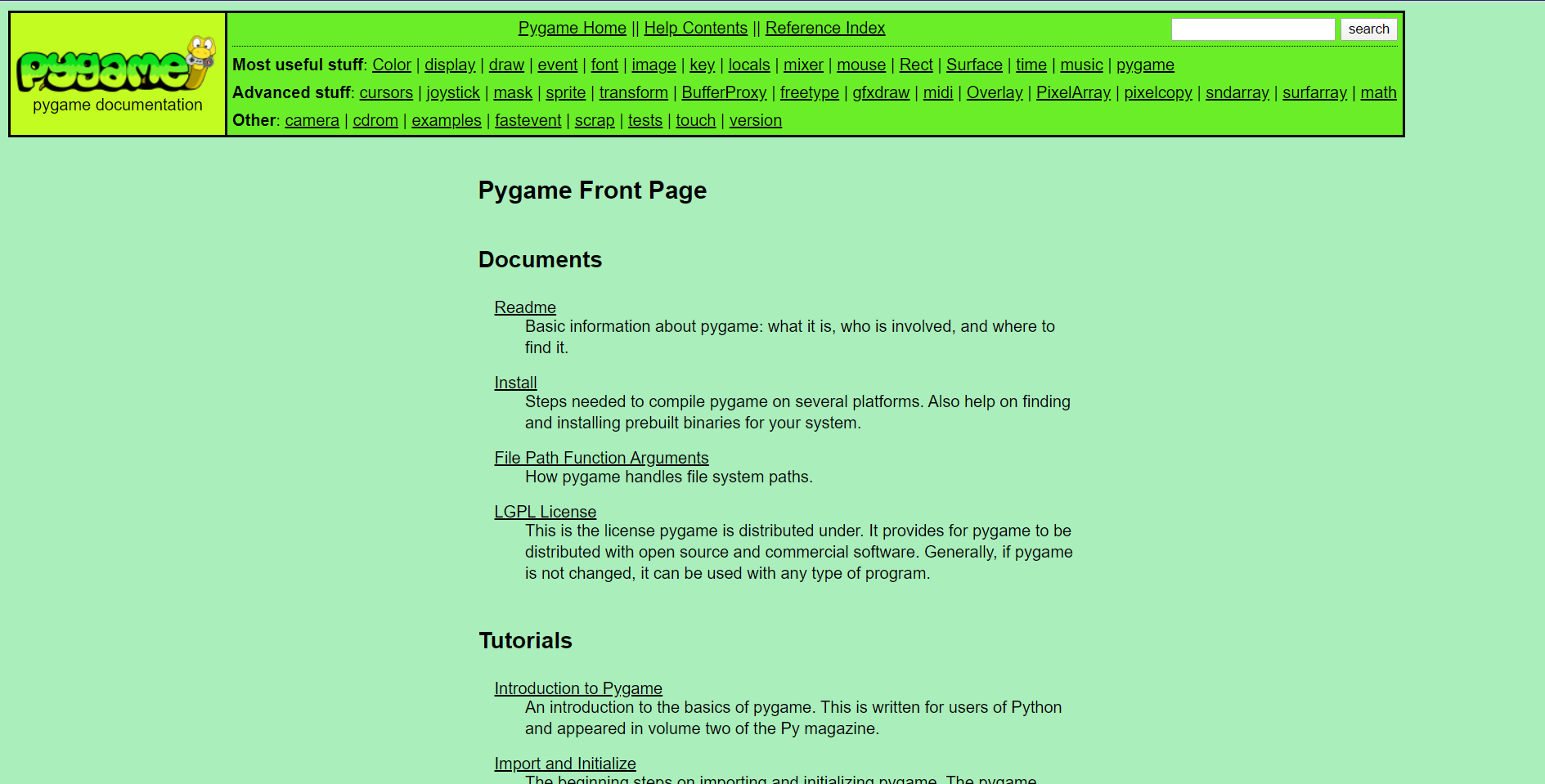
1. **Analyse de besoins Recherche d'idées**

L’idée nous est venue car nous jouions souvent à skribbl.io (un jeu sur internet). En voyant à quel point ce jeu était convivial, nous avons décidé de reprendre le principe de ce jeu mais à notre manière. C’est un jeu très accessible et très sympa.



L’objectif principal était de réussir à faire un jeu en réseau que ce soit LAN ou en ligne. Ensuite l’objectif était de faire un jeu joli avec un style enfantin et très simple à prendre en main

Le jeu a été réalisé avec python et surtout le module Pygame. Nous nous sommes principalement documentés sur la documentation officielle pygame ainsi que quelques exemples trouvés sur le site officiel de pygame.



1. **Répartition du travail**

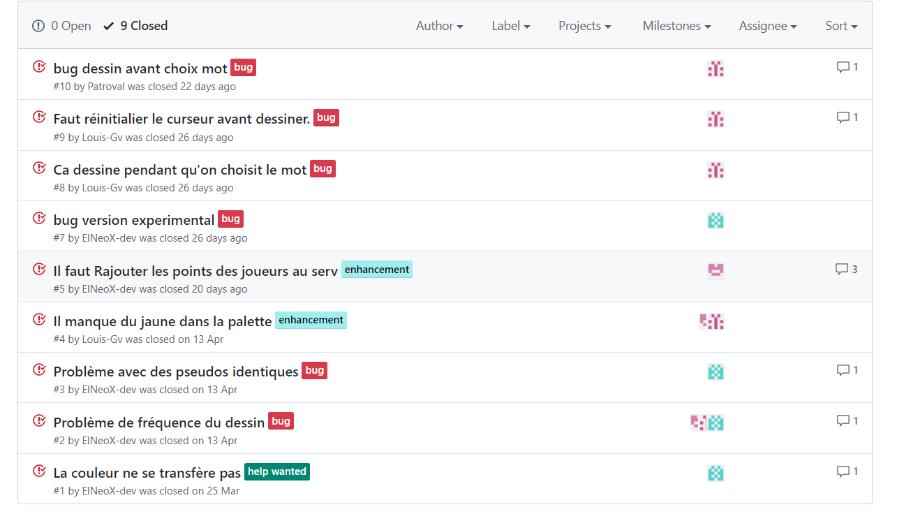
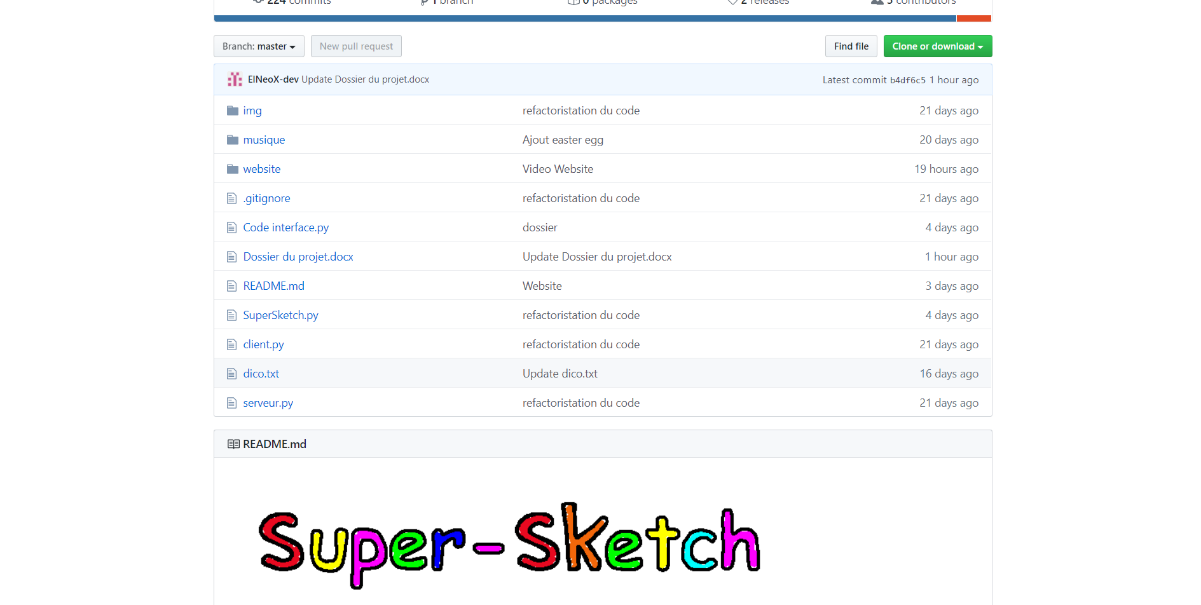
Nous avons développé Super-Sketch à 3. Il y a Louis Give, Hugo Warin et Yanis Peltre. Nous avons chacun développé une partie assez précise du programme.

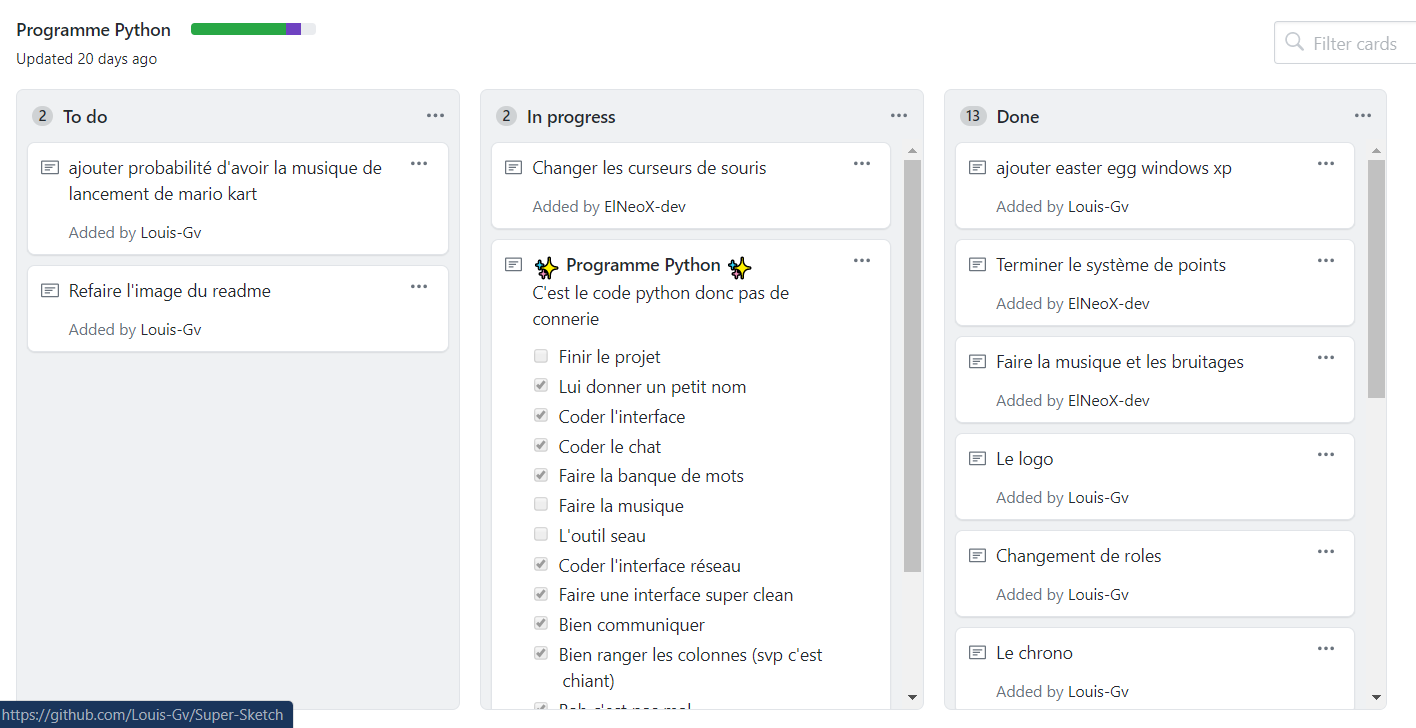
Louis Give : C’est chargé principalement de la communication entre tous les joueurs et de faire l’accueil du jeu.

Hugo Warin : C’est chargé de l’algorithme de dessin, du système de proposition des mots (du chat), des easters eggs ainsi que la transmission d’information et une petite partie du lobby

Yanis Peltre : C’est principalement chargé du Sound design et de la structure de l’interface (*idem*)

Pour communiquer et mettre en commun notre travail, nous avons utilisé l’outil GitHub. Ce gestionnaire de version nous permet de mettre nos modifications du code en commun (en respectant les conflits pour pouvoir travailler simultanément sur le même fichier), de reporter certains bugs et les attribuer à certains membres du groupe et de faire une liste des objectifs à faire.



****

1. **Réalisation**

Le programme est constitué de 3 grands états qui évoluent au fil du temps :

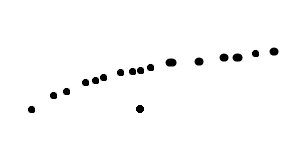
Soit le joueur est dans l’accueil, soit il dessine, soit il devine.

* 1. **Le Dessin**

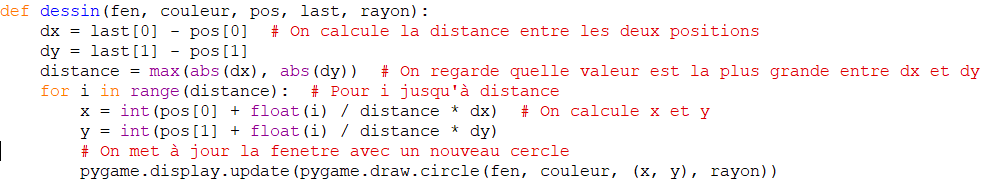
La première étape a été de trouver comment afficher le dessin de la souris avec le module pygame et de designer l’interface

Dessin :

Au départ nous récupérions la position de la souris à chaque rafraichissement du programme, puis nous affichions un cercle à cet endroit. Malheureusement cette technique avait un problème, elle dépendait énormément de la fréquence de rafraichissement. Si on dessinait trop vite, ça affichait ce genre de dessin. De plus, cela surchargeait le serveur et le client si on augmentait la fréquence de rafraichissement.



Nous avons donc dû trouver un autre algorithme bien plus efficace qui consiste à tracer pleins de cercles entre les points pour faire des segments.



* 1. **Le réseau**

Je me suis aidé du [cours sur les sockets en python de developpez.com](https://python.developpez.com/cours/TutoSwinnen/?page=page_20) où on apprend à créer une messagerie rudimentaire entre 2 IP. Mais les communications étaient limitées et bloquantes. Soit on attendait de recevoir un message et le programme était bloqué par la commande recv(), soit on envoyait un message sans savoir si un autre message nous a été envoyé. Cette méthode de communication n’était pas adaptée.

J’ai donc dû créer un processus client pour ne pas interférer avec le programme principal et des threads à l’intérieur de ce processus pour pouvoir recevoir et envoyer des messages simultanément. Ce processus communique avec le programme principal avec un objet Pipe().

Un processus serveur a lui aussi été créé pour celui qui héberge la partie. Il rediffuse chaque paquets reçus à tous les autres clients.

Pour que les clients se comprennent entre eux les données sont structurées de cette manière : 'D, 875;745, 845;730, 0;0;0, 10, @'.

Type de message, position, ancienne position, couleur, épaisseur, fin

‘D’ = trait du dessin

‘P’ = point

‘t’ = texte

‘R’ = changement de rôle (a été compliqué à réaliser)

…

Pour pouvoir envoyer plusieurs infos dans le même message

* 1. Design et son (Pour Yanis)

En ce qui concerne le design de l'interface de jeu il a fallu commencer par ce mettre d'accord sur le nombre et les emplacements des différents ajouts tel-que le chat ou les boutons. Une fois ceci fait j'ai donc placé les différents boutons grâce à :

- « pygame.draw.rect(fenetre,couleur, (coordonnées) »

- « pygame.draw.circle(fenetre,couleur, (coordonnées) »

Chaque bouton a été mis dans différentes variables qui nous permette de savoir quel bouton a été pressé en fonction des coordonnées du clic et de celle du bouton. Pour détecter le clic nous utilisons la fonctionnalité « pygame.mouse.get\_pressed() » de pygame.

Pour rendre tout cela plus esthétique j'ai fait des contours noir en dessinant des rectangles comme pour les boutons, ainsi que des zones grise clair pour ajouté encore plus de détails.

Pour la partie son j'ai tout d'abord dû chercher les bruitages que j'ai converti en .ogg ensuite je l'ai est importé grâce a la commande suivante :

nom du song = pygame.mixer.Sound(file="dossier/nom du song.ogg")

Une fois importé je n'avais plus qu'à les jouer au bon moments en intégrant le code dans la même boucle que son événement, pour cela j'utilise:

- nom du song\_song.play(0, 0, 0)

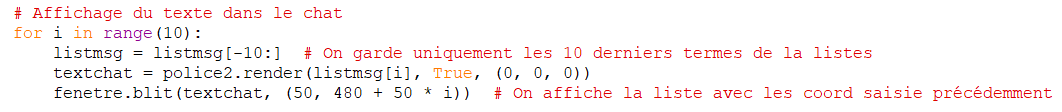
Le nombre de fois que

le morceau sera jouer

temps du fondu d'ouverture

Le temps du son

Le tchat : Le jeu étant en ligne, nous avons décidé d’incorporer un système de tchat. Pour cela, on attend de recevoir le message de la part du serveur, une fois celui-ci reçu, on l’ajoute à une liste et on l’affiche avec ce code



1. Bilan

Nous avons codé la plupart des fonctionnalités que nous souhaitions. En effet, le jeu en ligne est fonctionnel et avec très peu de bug, tout comme le jeu en local. Tous les algorithmes sont optimisés et ne surchargent pas le serveur. Nous avons un design simple et épuré (avec quelques easters eggs…)

Pistes d’améliorations : Nous pouvons encore rajouter pas mal de fonctionnalités, telle que donner des indices, l’affichage du mot qui était à trouver avec un écran de fin spécial. Au début du jeu, demander le nombre de round que nous voulons jouer, changer l’écran d’attente de l’accueil.

Ce projet nous a permis de découvrir comment fonctionnait un jeu en réseau, la programmation d’une interface graphique et comment optimiser des algorithmes.

1. Diffusion du projet

Le projet est diffusé sur GitHub à ce lien : <https://github.com/Louis-Gv/Super-Sketch/releases>

1. Annexe