Rapport d’avancement/Backlog

rapport d’avancement

- ✓ J’ai créé un fichier qui reprend les demandes du client et me permet de les décomposer en tâches à faire.
- ✓ J’ai appris à utiliser les projets github.
- ✓ J’ai appris à utiliser les branches et les pull request et les issues
- ✓ J’ai créé un repo github avec un projet contenant un tableau kaban et un tableau de gestion de bug
- ✓ J’ai pris les consignes et créé le product backlog
- ✓ Créer le release backlog
- ✓ Créer le sprint backlog

sprint backlog

- une fonction qui renvoi un nombre autour de 40 et dans certain cas, un pit ou un out
- une fonction de conversion pour afficher n’importe quel nombre au format hhmmssmsec
- une fonction qui renvoi une chaine de caractere qui sera l’entete du tableau et meme le tableau entier
- ajouter les valeurs dans le tableau generé par le fct ci dessus
- afficher un nouveau tab toute les x sec (pas en liste mais bien effacer et afficher)
- ecrire le dernier tableau dans un fichier

release backlog

- Version 1:

Pas de tri, ni de shmem ou sem. On veut juste des temps bien affichés.

  - générer s1, s2, s3, (s1+s2+s3) pit out
  - les convertir au format human readable
  - les afficher dans un tableau
  - rendre le tableau dynamique

    - stocker les résultats
- Version 2:

On doit déjà utiliser la shmem et sem ? ça serait bien de pouvoir s'occuper de ça plus tard

    - classement selon tour complet
    - stocker le best time tout le temps
    - (Q) éliminer 5 puis 5 puis classer dans l'ordre pour départ finale.
    - écrire les résultats (PQF) dans un fichier
    - affichage:
        - voiture|s1|s2|s3|besttime|pit|out
        - trié constamment pour que les meilleurs temps soit en haut
        - n'affiche plus de temps si il est out ou pit. Remettre la voiture à sa place dès son retour du pit
- Version 3:
    - synchroniser tout et s'assurer que les données sont cohérentes.
    - écriture d'un makefile
    - écriture doc, rapport
    - perfectionner le code en le paramétrant et refactor

product backlog

- un **afficheur** qui montre voiture|s1|s2|s3|tot|...
- 20 **voitures** qui génèrent des temps aléatoires de +/- 40sec/secteur, 2-3 pit, 2-3 crash
- 3 Périodes d'essai (P), 3 Qualifications (Q) où les 5 derniers de Q1 sont OUT et dernier du tableau (16-20), les 5 derniers de Q2 aussi (11-15) et enfin la Q3 sert à déterminer le classement de départ de la course finale (F)
- P/Q:
    - générer s1, s2, s3
    - classement selon tour complet (s1+s2+s3)
    - stocker le best time tout le temps
    - générer un pit (garde temps et classement)
    - générer un out (garde temps et classement)
    - (Q) éliminer 5 puis 5 puis classer dans l'ordre pour départ finale.
    - Stocker classement final du 3 et tour plus rapide
    - affichage:
        - voiture|s1|s2|s3|besttime|pit|out
        - trié constamment pour que les meilleurs temps soit en haut
        - n'affiche plus de temps si il est out ou pit. Remettre la voiture à sa place dès son retour
- écrire les résultats (PQF) dans un fichier

- Utiliser la **mémoire partagée** comme moyen de communication *inter-processus*

    - un père afficheur qui crée des fils qui génèrent les temps, pit, out.
- Utiliser les **sémaphores** pour synchroniser l'accès à la *mémoire partagée*
    - le père doit prendre les infos des fils dans la mémoire partagée mais il faut pas qu'il y ait de conflit.
- //paramétrer le programme pour insérer les options voulue (le nombre de tour car le circuit peut être +/- long)

- un makefile qui compile le code
- une documentation riche