Rapport:
ChenYI-Tech

SOMMAIRE:

CHOIX DU SUJET:

Lors du choix du sujet, nous avons longuement hésité entre Cy Fighter et ChenYI-Tech car ils étaient pour nous les deux seuls projets qui nous parlaient, dans lequel nous pouvions nous projeter afin de respecter l'échéance du projet, tout en apportant notre touche personnelle. Après avoir analysé la difficulté de chacun des deux projets, nous nous sommes décidés à prendre le Chenil-Tech à condition de rajouter de la fantaisie. C'est ainsi que notre réflexion s'est orientée vers l'univers de STAR WARS . Étant nous deux fan de la Saga cela n'a pas fait partie des choix difficiles à faire durant le projet. Cet univers nous a permis d'apporter des espèces originales et rajouté des personnages mythiques de la Saga. Impliquant le ChenYI-Tech dans cet univers. Ce qui crée une histoire autour de notre refuge comme si nous avions ajouté notre touche personnelle dans cette Saga mythique.

ORGANISATION:

- Échanges sur plusieurs jours afin de se répartir les tâches à suivre.

- Long travail pendant les vacances chacun de notre côté, mais mise en commun régulièrement via Google Doc, Discord, Github et autre.. Incluant remarques et avis.

- Lors de nouvelles idées, on exposait cette dernière à notre binôme et nous décidions si elle était intéressante.

- Les TD nous ont été utiles pour souligner certains points à corriger ou à changer.

- En dehors des TD, nous nous retrouvions sur les PC de l'école pour tester la compatibilité de notre programme sur Linux. Une nécessité inéluctable en vue des nombreux problèmes que nous avons rencontrés sur Linux et non présents sur notre compilateur Windows. Ce qui nous a permis de les éliminer.

CHOIX IMPORTANT:

- Espèce propre à l'univers de Star Wars avec leurs caractéristiques propres (ex:Poids, temps nettoyages).

- Dans le main.c, on ne fait appel qu' à des procédures pour pouvoir exécuter le programme en boucle et ne pas avoir à gérer des retours dans tous les sens. Une fois une fonction exécutée, on passe à la suivante sans devoir gérer les exécutions précédentes. (Nouvelle commande, nouveaux codes, nouvelles ouvertures de fichiers).

- Notre fonction main.c ne s'arrête que lorsque l'on rentre '5' ce qui vient stopper la boucle et donc le programme avec un petit message de au-revoir.

- Pour créer un registre respectant le format nécessaire, on a créé un programme fabricant automatiquement un fichier avec les caractéristiques attendues.

- Chaque fonction ouvre elle-même le fichier "registre" et l'utilise comme elle le doit. On a choisi de ne pas transmettre le fichier entre chaque fonction pour éviter tous problèmes liés à un problème de mémoire.

- Utilisation importante de tableaux statiques, cependant permet tout de même un fonctionnement correct et d'éviter tous problèmes de mémoire (malloc/free).

- La fonction Quicksort provient du cours, convertit pour obtenir un résultat dans l'ordre décroissant

- La fonction Race Espèce provient de chat gpt, permettant de renvoyer une chaîne de caractère invariable lorsqu'on lui donne un nombre correspondant à une espèce.

- Utilisation des ASCII Arts ajoutant au programme une interaction plus poussée ainsi que du cachet.

- Toutes les fonctions ont d'abord été réalisées séparemment du programme principal.

PROBLÈMES RENCONTRÉS:

- Pouvoir compiler les programmes en utilisant des fichiers sur nos PC personnels était assez difficile au début.

- Le passage de Linux à Windows est assez contraignant, l'inverse moins. Certaines fonctionnalités sont indisponibles sur les IDE que nous avions sur nos PC personnels: "clear", ou les couleurs d'écritures dans le terminal.

- Gestion du registre, lors des premiers lancements des registres, le programme crashait constamment lors de la conversion du registre en tableau.
    - Solution: Avant le lancement du programme, il faut vérifier qu'il n'existe pas de ligne nulle dans le registre

- La fonction de recherche a été difficile. Au début, il était difficile de s'imaginer comment on allait gérer cette option.
    - Solution: La création de tableau avec des ID corrects ou à -1. Cependant, le fait que beaucoup de cases soient inutiles (-1) allonge le temps d'exécution.

- Recherche des animaux, la complexité du programme est assez importante mais ça ne gêne pas l'exécution.

- Sur windows, la compilation est différente. Notamment, l'ouverture des fichiers lors de l'exécution du programme se fait dans un dossier à part.
    - Solution: lors de chaque ouverture, on a eu l'obligation de sortir du dossier.
      "../Animaux/Registre.txt" ->ANNEXE: Image 1.

- Lors de la recherche par prénoms, si on saisissait un nom en majuscule, alors qu'il n'était pas dans le même format dans le registre. L'animal n'était pas trouvé.
    - Solutions: Formaté tous les noms de la même manière, c'est à dire, tous avec une majuscule uniquement pour la première lettre

- Lors de la recherche par âge, spécifiquement les animaux qui ont moins de 20 ans, des animaux qui n'existaient pas dans le registre étaient affichés.
    - Solutions: Le tableau des animaux étant initialement à 0, le programme affichait des animaux qui avaient 0 ans. Une vérification était donc nécessaire, age>0.

- Effectuer la vidéo était assez contraignant, aucun logiciel sur les PC de l'école n'était présent et il nous était impossible de télécharger nous même un logiciel car il nous fallait un code Admin

- Problèmes rencontrés quelques fois vers la fin du projet, notamment lors de la vidéo effectuée sur Windows. Affichage: "Erreur race 'afficher registre' " (nettoyage.c/l.105)
    - Suppositions: Conversion Linux/Windows problématique.
      Lecture de cases mal initialisées (nettoyage.c/l.83)
    - Solutions: Aucunes appliquées, crainte de modification sur Windows non compatibles sur Linux

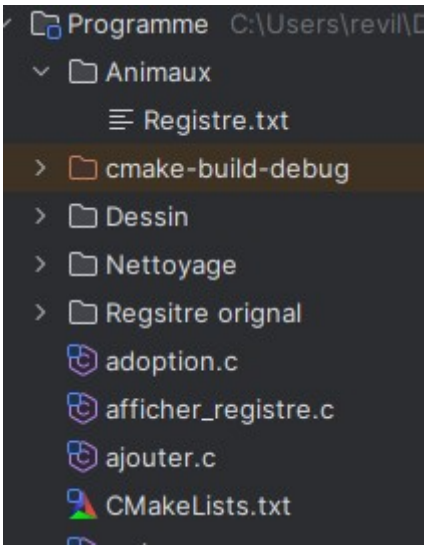
- Pour les ASCII Arts il fallait gérer les espaces qui étaient laissés car lors de la compilation ils pouvaient faire sauter une ligne et détruire le dessin. De même pour les bulles de discussions des personnages qui ne devaient pas être trop loins du personnages.

- Toutes les saisies qui ne correspondaient pas à ce qui était attendu faisait crasher le programme (Ex: '*', 'abc'...). Nous avons pu le régler avec une boucle incluant un getchar.

- La fonction Adoption n'était pas simple à réaliser. Au début nous voulions partir sur une idée impliquant des tableaux mais nous avons vite bloqué . Alors nous avons changé d'idée et utilisé deux fichiers 'registre', un qui était l'original puis l'autre un temporaire qui copie le registre original sans la ligne de l'animal qui doit être adopté. Puis on renomme le registre 'temp' en 'registre' et on supprime le premier 'registre' (l'original), si l'adoption est possible. Une sorte de mise à jour du registre avec les fonctions rénale et remove.

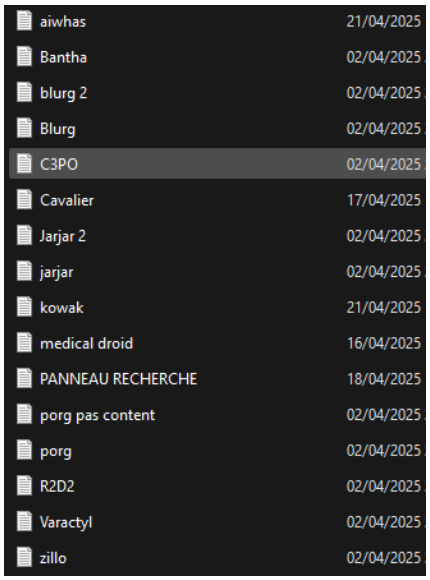
ANNEXE

IMAGE 1 :



L'exécutable se trouve dans 'cmake-build-debug' mais le registre et donc dans un dossier différent.

IMAGE 2 :



Recheche d'ASCII Art compatible avec notre projet.

IMAGE 3 :

Recherche de nos espèces

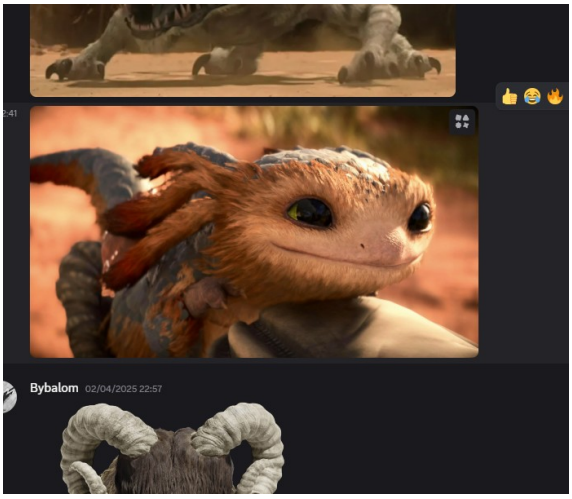


IMAGE 4 :

Fonctions test