CESI exia
ÉCOLE D'INGÉNIEURS INFORMATIQUE

Projet eXiaSaver

RAPPORT DE PROJET

Juliette PORTE, Vincent SERE-PEYRIGAIN, Baptiste LAURAS
2016-2017 | CESI EXIA BORDEAUX

Table des matières

I. Rappel du besoin

Afin d'appliquer les compétences acquises depuis le mois d'octobre sur la « Programmation procédurale » et le « Système Linux », nous avons pour mission de réaliser un écran de veille sur Linux en mode console. Il se nommera eXiaSaver *et devra être un exécutable qui lancera « un joli termSaver » depuis la ligne de commande d'un terminal Linux.*

Notre eXiaSaver doit être composé de 3 type d'écrans différents : un statique, un dynamique et un interactif. L'écran se lancera de façon automatique sans l'interaction de l'utilisateur grâce à un « lanceur » qui choisira au hasard l'un des trois types à son démarrage. Le programme devra aussi proposer un historique grâce au paramètre « -stats ».

Voici le tableau des caractéristiques que chacune des parties de notre eXiaSaver doit remplir :

LE LANCEUR	Il existe et il peut être lancé depuis la console sans paramètres
	Le lanceur "vide" la console lors de l'exécution
	Historique - Enregistre les informations dans le fichier "historique"
	Le programme interprète correctement le paramètre '-stat' et ne lance pas un écran de veille
	Affichage menu stat avec plusieurs choix
	Calcul et affichage statistiques
	Choisit aléatoirement un de trois écrans
	Il lance bien un des 3 exécutables avec les bons paramètres
	Parcours répertoire des images PBM (ou PPM)
	Choix aléatoire d'une image pour le lanceur de type statique
	Créer au moins 5 images de tailles différentes
	On voit l'implémentation d'un algorithme de tri simple pour les stats
ECRAN DE TYPE STATIQUE	Image chargée correctement
	Affichage image centrée dans la console
	Déblocage en touchant sur « espace »
	Utilisation de fork pour chargement de l'image

ECRAN DE TYPE DYNAMIQUE	Images de chiffres chargées en mémoire (normalement 1 seule fois)
	Affichage heure centrée dans la console
	Affichage message d'attente et réactualisation
	On voit bien passer les secondes (10" par défaut) entre 2 réaffichages
	Affichage heure centrée dans la console après refresh auto de l'écran
	Déblocage du processus par « espace »
	Affichage heure avec tailles différentes (5x3, 7x4, …) Au moins 2 !
ECRAN DE TYPE INTERACTIF	Images des avions chargées en mémoire depuis les fichiers PBM
	Affichage de l'avion à la position (initiale) passée en paramètre & choix aléatoire de la direction et du sens.
	L'avion réagit quand on tape sur une des 4 touches de commande
	Affichage de l'avion à sa nouvelle position
	Si on ne tape rien (seulement Entrée) l'avion avance dans le même sens, d'une case à la fois
	Quand l'avion sort de l'espace aérien, il rentre du côté opposé selon spécification
	Sortir de l'écran en tapant une touche précise

II. Organisation du groupe

Afin de réaliser ce projet, nous nous sommes réparti le travail d'une certaine façon. Nous avons commencé en nous répartissant les différentes fonctions du ScreenSaver, reprenons le tableau précédent pour illustrer la répartition dans le groupe.

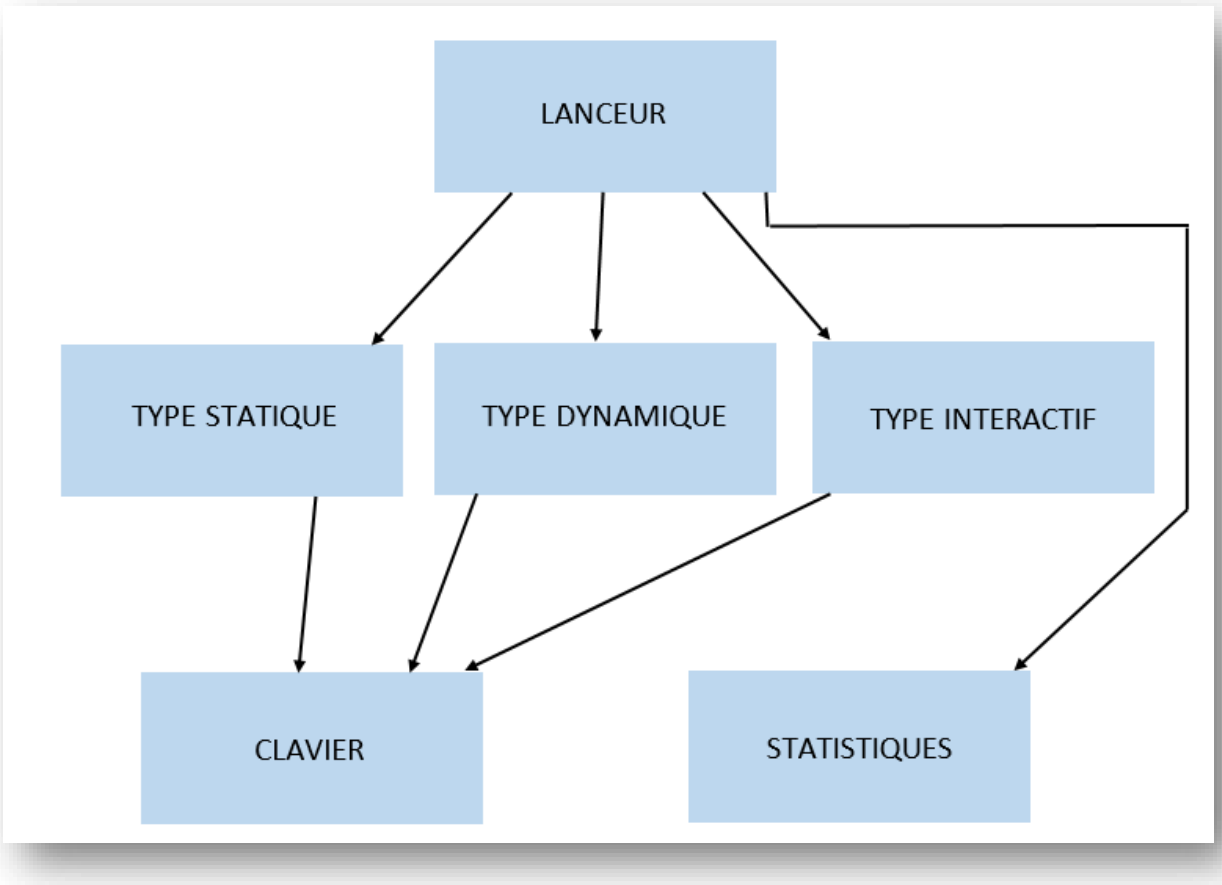
La couleur jaune représente le travail de Juliette, la couleur bleue représente le travail de Baptiste et la couleur rouge représente le travail de Vincent. En vert les fonctions qui ont été travaillée par Vincent et Baptiste.

LE LANCEUR	Il existe et il peut être lancé depuis la console sans paramètres
	Le lanceur "vide" la console lors de l'exécution
	Historique - Enregistre les informations dans le fichier "historique"
	Le programme interprète correctement le paramètre '-stat' et ne lance pas un écran de veille
	Affichage menu stat avec plusieurs choix
	Calcul et affichage statistiques
	Choisit aléatoirement un de trois écrans
	Il lance bien un des 3 exécutables avec les bons paramètres
	Parcours répertoire des images PBM (ou PPM)
	Choix aléatoire d'une image pour le lanceur de type statique
	Créer au moins 5 images de tailles différentes
	On voit l'implémentation d'un algorithme de tri simple pour les stats
ECRAN DE TYPE STATIQUE	Image chargée correctement
	Affichage image centrée dans la console
	Déblocage en touchant sur « espace »
	Utilisation de fork pour chargement de l'image

ECRAN DE TYPE DYNAMIQUE	Images de chiffres chargées en mémoire (normalement 1 seule fois)
	Affichage heure centrée dans la console
	Affichage message d'attente et réactualisation
	On voit bien passer les secondes (10" par défaut) entre 2 réaffichages
	Affichage heure centrée dans la console après refresh auto de l'écran
	Déblocage du processus par « espace »
	Affichage heure avec tailles différentes (5x3, 7x4, …) Au moins 2 !
	Images des avions chargées en mémoire depuis les fichiers PBM
ECRAN DE TYPE INTERACTIF	Affichage de l'avion à la position (initiale) passée en paramètre & choix aléatoire de la direction et du sens.
	L'avion réagit quand on tape sur une des 4 touches de commande
	Affichage de l'avion à sa nouvelle position
	Si on ne tape rien (seulement Entrée) l'avion avance dans le même sens, d'une case à la fois
	Quand l'avion sort de l'espace aérien, il rentre du côté opposé selon spécification
	Sortir de l'écran en tapant une touche précise

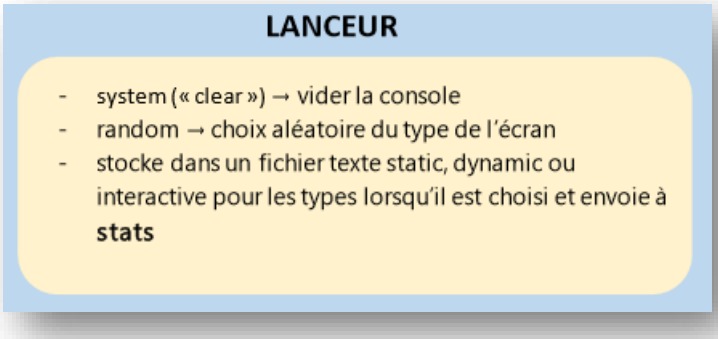
III. Technique

Après un brainstorming en groupe et quelques schéma réalisés au tableau, nous avons réalisé l'architecture du programme de notre eXiaSaver :



Il se divise en 6 « .c », et autant de « .h » lancés dans le LANCEUR. Nous allons développer en 6 parties comment nous avons programmé ces parties. Nous avons aussi décidé de coder notre eXiaSaver entièrement en anglais (sauf commentaires en français) pour que tout soit homogène.

a. Le lanceur



Juliette - Pour réaliser le launcher.c, j'ai créée 6 fonctions permettant de le faire tourner voici la liste :

- *screenChoice()* : le premier critère du lanceur est de choisir un des trois type d'écran au hasard. Cette fonction permet justement et simplement de faire un random qui choisit entre 1 (type statique), 2 (type dynamique) ou 3 (type interactif) et de renvoyer sa valeur

- *writeFile()* : cette fonction permet de faire une sauvegarde du nombre (correspondant à l'écran choisi en random) dans un fichier texte contenu dans le même répertoire que le programme (nommé hist.txt). En effet, grâce à un pointeur et la fonction « fopen », la fonction ouvre le fichier texte afin d'y écrire chacun des nombres dès qu'ils sont trouvé par le random. Un appel system permet aussi de trier les caractères de hist.txt en les copiant (triés) dans un autre fichier nommé histTrie.txt. La fonction écrit aussi un message d'erreur si le pointeur ne trouve pas le fichier.

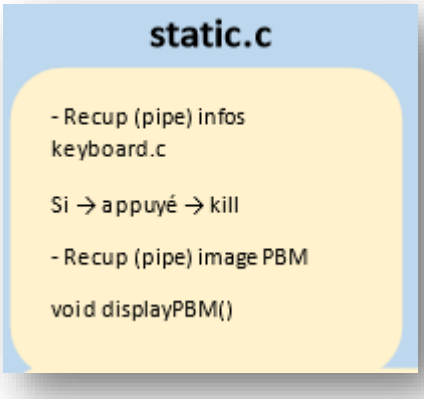
- *comptFiles()* : Elle ouvre un fichier texte contenant le chemin d'accès de chaque pbm destinées à être lues dans le Statique et à compter le nombre de fichier référencés dans ce fichier texte.

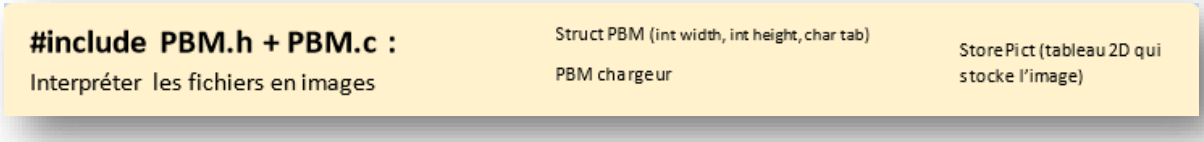
- *randomPbm()* : cette fonction sert à tirer un nombre au hasard parmi le nombre de fichiers référencés dans le fichier texte.

- *stockPbmName()* : Cette fonction va se placer à la ligne correspondante au numéro tiré au hasard et va copier le chemin du pbm dans un tableau de caractères.

- *readHist()* : cette fonction est un void qui permet uniquement de lire l'historique non trié grâce au même principe d'ouverture que la fonction writeFile() (fopen). Ensuite, la fonction lit tous les caractères un à un et les affiche.

b. Le type statique





Baptiste – Le type statique est composé du lecteur PBM (qui fonctionne dans tous les types d'écran) uniquement puisqu'il doit simplement afficher une image sous format PBM.

- *picture* : c'est une structure permettant de stocker toutes les informations d'un fichier .pbm, le nombres de colonnes, de lignes, son indice(P1…) et un tableau à deux dimensions contenant le fichier convertit en caractères. C'est l'élément central du programme statique.

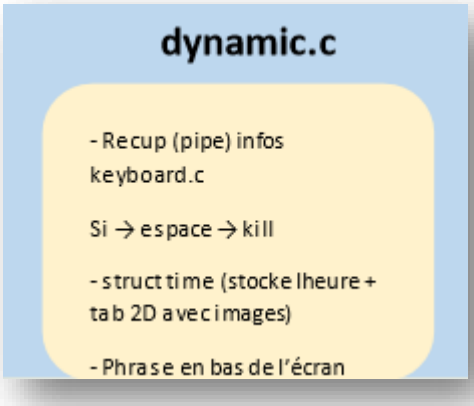
- *convertPictureIntoArray()* : Elle me permet de transformer les 1 et 0 du fichier .pbm en caractères et les stocks dans le tableau à deux dimensions de la structure picture.

- *printPicture()* : cette fonction me permet d'afficher les caractères du tableau à deux dimensions de façon centrée à l'écran sur la longueur.

- *height()* : elle me permet de centrer l'affichage sur la hauteur dans la console.

- *pbmReader()* : cette dernière fonction me sert à lire les caractéristiques stockées dans un fichier .pbm et de les stocker dans la structure picture.

c. Le type dynamique



Vincent – L'écran de type dynamique doit afficher l'heure en PBM sur la console tout en étant centré. Pour réaliser ce programme j'ai utilisé les fonctions ci-dessous.

- *printf_center()* : Cette fonction permet de faire le centrage de l'heure et de la phrase à afficher : « time will be refresh in few seconds » suivis des 5 points qui apparaisse un à un au centre de la console.

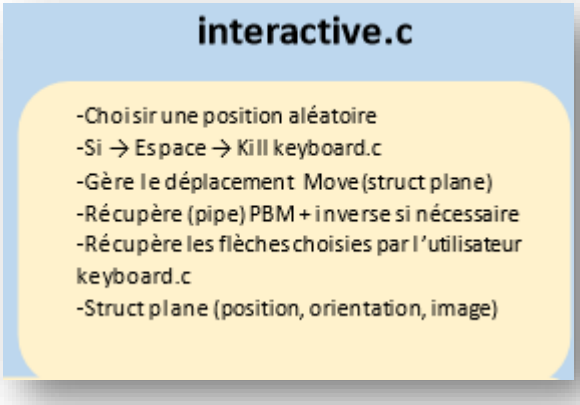
- *timeRead()* : cette fonction est la fonction qui récupère l'heure avec la bibliothèque <time.h> et qui l'affiche dans la console.

- *prinfSentenceRefresh()* : Cette fonction écrit la phrase « time will be refresh in few seconds » avec une boucle qui affiche les points un à un jusqu'à en voir 5 puis reviens 0 et ainsi de suite.

- *dynamic()* : Le « main » des fonctions du type Dynamique. Les appellent lors de la compilation et de l'exécution.

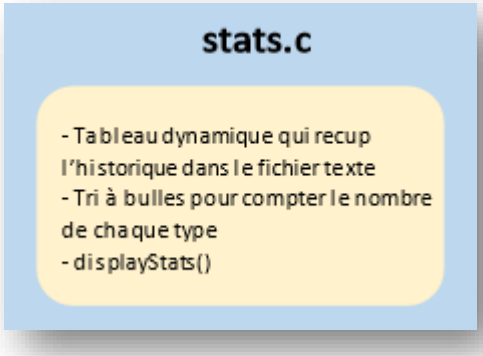
- *conio.h* : Bibliothèque window modifié pour linux qui donne accès à la commande getch() qui permet de récupérer des caractères sans les echo dans la console. En principe, elle devait être utile pour arrêter le programme mais elle n'a pas été utilisé dans le type dynamique.

d. Le type interactif



Nous n'avons pas beaucoup travaillé sur le type interactif par manque de temps accordé à celui-ci car nous avons préféré nous consacrer à la réalisation du launcher de l'affichage statique et au type dynamique.

e. Statistiques



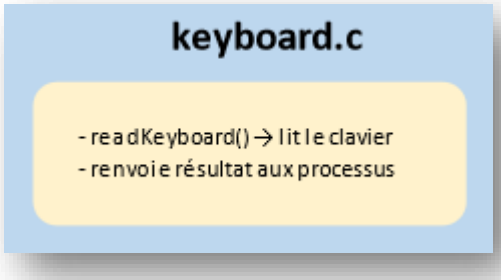
Juliette – Le menu statistique permet de calculer, à partir d'un historique les statistiques de chacun des écrans, c'est-à-dire le nombre de fois où chacun est apparu comparé au nombre total de choix d'écran.

- *statsMenu()* : comme son nom l'indique, cette fonction affiche le menu des statistiques. Elle propose deux options : ouvrir l'affichage des statistiques (1), ou quitter (2). Grâce à la fonction de kill de processus lorsqu'on appuie sur 2, le programme se ferme. De même lorsqu'on est rentré dans la première partie des statistiques (en appuyant sur 1), en appuyant sur la barre espace, le programme se ferme.

- *readHistTri()* : cette fonction lit le fichier où les nombres stockés des choix d'écrans sont triés. En effet, grâce à un pointeur et fopen, la fonction ouvre le dossier histTrie.txt avec comme paramètre « r », c'est-à-dire uniquement de la lecture. Ensuite, grâce à fgetc(), le pointeur lit chacun des caractère et lorsqu'il tombe sur un '1', il augmente d'un le nombre correspondant au nombre de fois où le random du lanceur est tombé sur « 1 ». De même pour 2 quand il tombe sur '2' et 3 quand il tombe sur '3'. Tout cela jusqu'à « EOF », c'est-à-dire le dernier caractère du fichier. Pour finir, la fonction crée un tableau « statistic » de 4 cases pour stocker toutes les variables de nombres ainsi que la variable englobant le total.

- *displayStats()* : enfin, cette fonction affiche les statistiques lorsqu'on rentre dans le premier paramètre du menu (lorsuq'on appuie sur 1 dans le menu).

f. Clavier



Juliette- Pour réaliser cette partie, je n'ai pas créé de « .c » dédié à la reconnaissance du clavier. J'ai simplement utilisé la bibliothèque « cornio.h ».

En effet, cette dernière a un fonctionnement semblable à la fonction getch() sous windows. Elle permet de récupérer le caractère ASCII d'une touche lorsqu'on appuie dessus. Ensuite, à la fin du programme, j'ai créé une variable « key » qui permet d'écrire « key = getch() » qui fonctionne comme un scanf sans besoin d'appuyer sur entrée pour exécuter la fonction. Ensuite, pour rendre cela plus esthétique, j'ai utilisé l'appel system « setterm -cursor off » qui permet de cacher le curseur blanc qui clignote.

Par exemple, si je veux que mon programme s'éteigne quand j'appuie sur « 2 », dans une boucle infinie, je place un if qui me dit « si key == 50 », je quitte le programme grâce à un « exit(0) ». En effet, sur la table ASCII, 50 correspond à 2. On peut faire cela pour toutes les touches du clavier.

IV. Bilan du projet

a. Bilan du groupe

LE LANCEUR	Il existe et il peut être lancé depuis la console sans paramètres
	Le lanceur "vide" la console lors de l'exécution
	Historique - Enregistre les informations dans le fichier "historique"
	Le programme interprète correctement le paramètre '-stat' et ne lance pas un écran de veille
	Affichage menu stat avec plusieurs choix
	Calcul et affichage statistiques
	Choisit aléatoirement un de trois écrans
	Il lance bien un des 3 exécutables avec les bons paramètres
	Parcours répertoire des images PBM (ou PPM)
	Choix aléatoire d'une image pour le lanceur de type statique
	Créer au moins 5 images de tailles différentes
	On voit l'implémentation d'un algorithme de tri simple pour les stats
ECRAN DE TYPE STATIQUE	Image chargée correctement
	Affichage image centrée dans la console
	Déblocage en touchant sur « espace »
	Utilisation de fork pour chargement de l'image

ECRAN DE TYPE DYNAMIQUE	Images de chiffres chargées en mémoire (normalement 1 seule fois)
	Affichage heure centrée dans la console
	Affichage message d'attente et réactualisation
	On voit bien passer les secondes (10" par défaut) entre 2 réaffichages
	Affichage heure centrée dans la console après refresh auto de l'écran
	Déblocage du processus par « espace »
	Affichage heure avec tailles différentes (5x3, 7x4, …) Au moins 2 !
	Images des avions chargées en mémoire depuis les fichiers PBM
ECRAN DE TYPE INTERACTIF	Affichage de l'avion à la position (initiale) passée en paramètre & choix aléatoire de la direction et du sens.
	L'avion réagit quand on tape sur une des 4 touches de commande
	Affichage de l'avion à sa nouvelle position
	Si on ne tape rien (seulement Entrée) l'avion avance dans le même sens, d'une case à la fois
	Quand l'avion sort de l'espace aérien, il rentre du côté opposé selon spécification
	Sortir de l'écran en tapant une touche précise

Le travail n'a pas été totalement fini. Le début du projet fut difficile pour le groupe : nous avons eu du mal à cerner les attentes et l'organisation dans un projet (malgré une compréhension du sujet). De plus, cette difficulté à démarrer a engendré quelques mésententes dans le groupes dues à des soucis de compréhension les uns des autres. On a su se reprendre et atteindre les objectifs que nous nous étions fixés.

b. Bilan personnel

Juliette (chef de projet) :

Au début du projet, je me suis sentie directement très sereine, emballée et confiante. Je me suis mise en groupe avec deux personnes avec qui j'avais déjà des affinités et avec qui je travaillais souvent le soir après les cours. Nous formons depuis le début de l'année un petit groupe de travail qui a toujours bien fonctionné, mêlant travail et plaisir de travailler. Je savais alors dès le début que notre groupe pouvait marcher. Le début fut un peu laborieux, nous avons eu du mal à nous lancer et avons chacun essayé du coup de chercher des réponses à nos questions comme : comment tuer un processus depuis un programme, comment afficher un PBM ou encore comment reconnaitre les touches d'un clavier. C'est de là que nous avons commencé à nous répartir naturellement le travail comme expliqué dans la partie II. Evidemment, chacun s'est tenu à son travail mais comme nous étions déjà assez soudés, nous connaissions chacun les points faibles des autres et nous aidions tout naturellement. De ce fait, malgré quelques petits blocages sur du code, nous avons chacun bien réalisé nos parties.

J'estime avoir fait du bon travail, j'ai appris énormément grâce à ce projet surtout en programmation et GitHub. Je suis fière de la partie que j'ai fournie et je pense m'être vraiment investie malgré 2 jours où je ne fus pas en cours pour cause de maladie en gardant tout de même contact avec mon groupe. J'aurais pu m'améliorer si le début n'avait pas été aussi « laborieux » pour comprendre ce que l'on attendait vraiment de nous.

C'est moi qui me suis proposée en tant que **chef de groupe** et Vincent et Baptiste ont été d'accord car je voulais tenter l'expérience. En tant que chef de projet, je suis fière de mon groupe, en effet, je savais dès le début que Baptiste était bon en programmation, et malgré quelques blocages, il me l'a encore prouvé à travers ce projet tout en montrant de la volonté à vouloir réussir. Quand à Vincent qui appréhendait au début du projet, j'ai pu découvrir qu'il s'est lui aussi bien donné dans le projet, et a réalisé des parties assez importantes du code ainsi que la grosse partie sur GitHub qu'il a étudié avant de nous expliquer. Je pense que Vincent et moi avons fait des réels progrès en programmation (Baptiste aussi mais lui avait déjà, comme je l'ai dit plus haut, un niveau plutôt bon). Malgré quelques petits accrochages entre nous dû à nos caractères assez forts, on a su en faire abstraction. Je pense avoir bien mené et gérer mon groupe qui finalement s'est aussi bien géré lui-même et qui voulait vraiment réussir sans baisser les bras. J'ai apprécié manager le groupe, répartir le travail et m'occuper du rendu word. Mon unique crainte était de devenir trop autoritaire et finalement je ne pense absolument pas avoir fait d'excès. J'ai bien aimé ce projet qui m'a beaucoup apporté que cela soit par mon travail dans le groupe en programmation ou encore en tant que chef de projet.

***Baptiste** :*

Au début de projet j'avais plein d'idées afin de pouvoir le réaliser, il ne m'avait pas l'air très compliqué cependant il semblait assez long. Mais finalement je me suis retrouvé confronté à plusieurs problèmes et il s'est avéré que mes idées n'étaient pas si bonnes. J'ai perdu deux journées à essayer de corriger des erreurs dans des codes.

Concernant le groupe, je me sentais bien dedans car nous avons l'habitude de travailler ensemble même si nous avons connus quelques tensions tout revenait à la normal en peu de temps. De plus Juliette, notre chef de projet, a su bien répartir les tâches et à définir les programmes prioritaires afin d'avoir un début de programme fonctionnel.

Je pense que nous aurions pu aller plus loin dans ce projet si nous avions travaillés à plusieurs sur les parties les plus difficiles afin de mieux penser nos programmes et éviter de perdre du temps à trouver des solutions aux problèmes.

Pour ma part j'ai plutôt bien apprécié ce projet car il réutilisait tout ce que nous avons vu durant les cours et il m'a permis de voir mon niveau et je me sens désormais plus à l'aise sur le langage C notamment sur l'usage des pointeurs. Il m'a aussi permis de découvrir ce à quoi correspondait le travail d'équipe car avec une personne en moins le projet aurait été trop difficile à gérer pour deux personnes.

Vincent :

Début du projet : du mal à démarrer étant donné la charge de travail et mon niveau en programmation C. Je me sentais plutôt perdu lors de la répartition du travail et je n'avais aucune idée de comment j'aillais réaliser mes programmes. Une grande étape de recherche a donc été faite puis avec l'aide de baptiste j'ai réalisé me premier code. Suivis de cette partie (2 jours environ) j'ai pu réaliser mes codes moi-même et je ressens une grande satisfaction de savoir que je suis capable de programmer.

De plus, j'ai réalisé le GitHub de mon groupe, mis en place les branche de chaque membre de mon groupe et appris les commandes afin d'utiliser GitHub comme demandé par le projet. Cette partie fut intéressante pour ma part et m'a permis d'enlever une charge de travail de recherche pour les membres de mon groupe sans oublié de leur expliquer les étapes de l'utilisation de GitHub.

Milieu de projet : Amélioration de mes codes et agencement nous étions en phase de travail personnel. Cependant nous avons eu un manque de communication car à la suite de cela, nos codes ne correspondaient pas et la fin du projet allé être compliqué.

Nous avons aussi essayé de réaliser la partie interactive du projet avec Baptiste cependant, le manque de temps nous à obliger de laisser de côté cette partie pour bien faire les autres.

Fin du projet : Assemblage des codes très compliqué, problème de bibliothèque et de correspondance des codes. Nous nous sommes réunis pour faire le rapport et faire le point sur tout ce que nous avions fait durant la période de projet.

Groupe : Un début compliqué, beaucoup de mésentente, chef de projet malade durant deux jours, cependant, Juliette a su rester en contact avec nous et nous attribuer les tâches à accomplir. Suite à une mise au point pour tout le monde pour les problèmes de groupe et plusieurs travaille réaliser ensemble, l'entente c'est amélioré, le but de ce projet était bien précis pour nous tous et une vraie cohésion s'est créer et suite à cela nous avons pu avancer à grand pas.

Pour conclure, ce projet m'a apporté énormément d'un point de vue technique et moral, il m'a permis de prendre plus confiance en moi et de me montrer que j'étais capable de faire des réalisations en programmation. D'un point de vue groupe et management de projet, j'ai compris que le travail de groupe était primordial pour la réussite du projet. Le travail personnel et bien-sûr important mais ne peux en aucun cas tout faire, la construction du travail à faire à plusieurs et je pense le plus important. Ce projet fut donc très intéressant et très pédagogue pour ma part.