Marvin Aubert Patrick Felix-Vimalaratnam

Rapport Snake

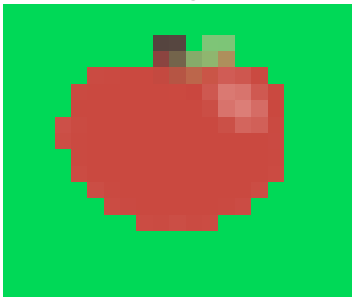
Sommaire:

Partie 1: Introduction

Le sujet est un jeu vidéo.
Ce jeu est un Snake. Le Snake est un jeu vidéo où le joueur contrôle un serpent, qui en ramassant des collectibles, grandit et où il doit se déplacer dans un environnement restreint.
L'obstacle principal dans ce jeu est le fait que si sa tête touche une partie de son corps ou les limites du terrain dans lequel il se trouve, il meurt. De plus le serpent mange de consommables et plus il va vite en plus de sa croissance.
Il a connu un regain de popularité quand l'entreprise Nokia l'a intégré à ses produits.

Partie 2: Description des fonctionnalités

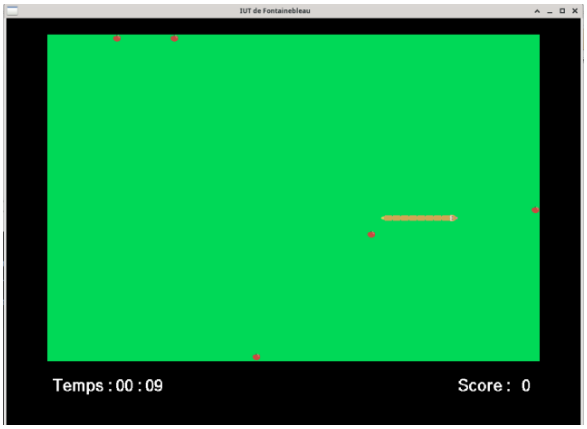
Dans notre jeu, on retrouve les différentes fonctionnalités du Snake qui est de manger des collectable pour augmenter son score.



Chaque pomme mangée augmente le score de 5 et agrandit la taille de serpent de 2 blocs.

Le serpent en début de partie fait une taille de 10 blocs de longueur.

Le jeu se déroule sur un plateau de 750 pixels de hauteur et 1050 pixels de longueur.



Il y a 5 pommes au maximum sur le plateau. Quand une des pommes est récupérée par le joueur, une nouvelle pomme apparaîtra pour la remplacer.

Lorsque le score est un multiple de 40, la vitesse du serpent augmente.

Nous avons également d'autres fonctionnalités: un menu start, un menu stop et un menu game over.

- Menu Start: Lancez le jeu avec la touche Entrée / Quittez le jeu avec la touche Echap

- Menu Stop: Mettre Pause en cliquant sur la Barre Espace / Relancer le jeu avec la Barre Espace / Quittez avec Echap Menu

- Game Over: Relancé une partie avec la touche Entrée / Quittez le jeu avec la touche Echap

(les os sont carrément invisibles sur du blancs)

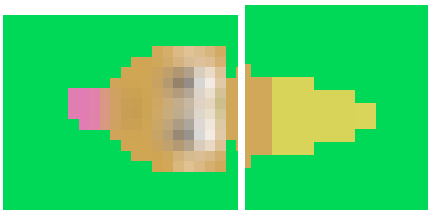
Partie 3: Justification du découpage des fichiers

Liste des fichiers .c:

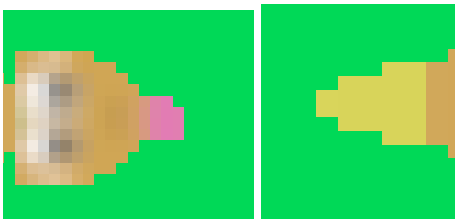
- main.c
- serpent.c
- pastille.c
- menu_terrain.c
- temps_score.c

Le fichier main.c contient le main

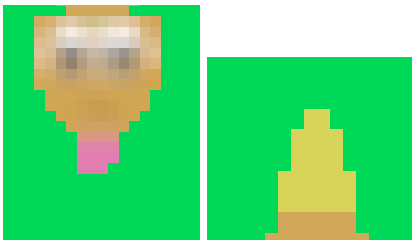
Le fichier serpent.c contient les fonctions, Création_Serpent, Afficher_Serpent, Afficher_Serpent_Haut, Afficher_Serpent_Droite, Afficher_Serpent_Bas, Afficher_Serpent_Gauche,Deplacer_Serpent, Mort_Serpent. La fonction Creation_Serpent sert à générer le serpent. Afficher_Serpent sert à afficher le serpent et Afficher_Serpent_… sert à afficher la tête et la queue du serpent en fonction de la direction.



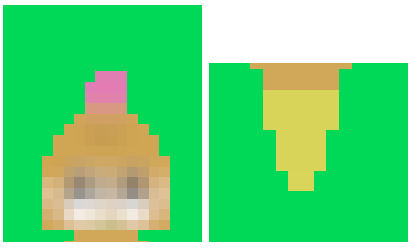
(Afficher_Serpent_Gauche)



(Afficher_Serpent_Droite)

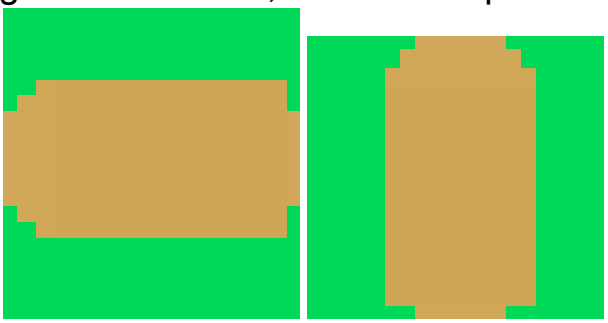


(Afficher_Serpent_Bas)



(Afficher_Serpent_Haut)

Il change également la direction du corps qui est soit horizontal pour gauche et droite, soit vertical pour haut et bas.



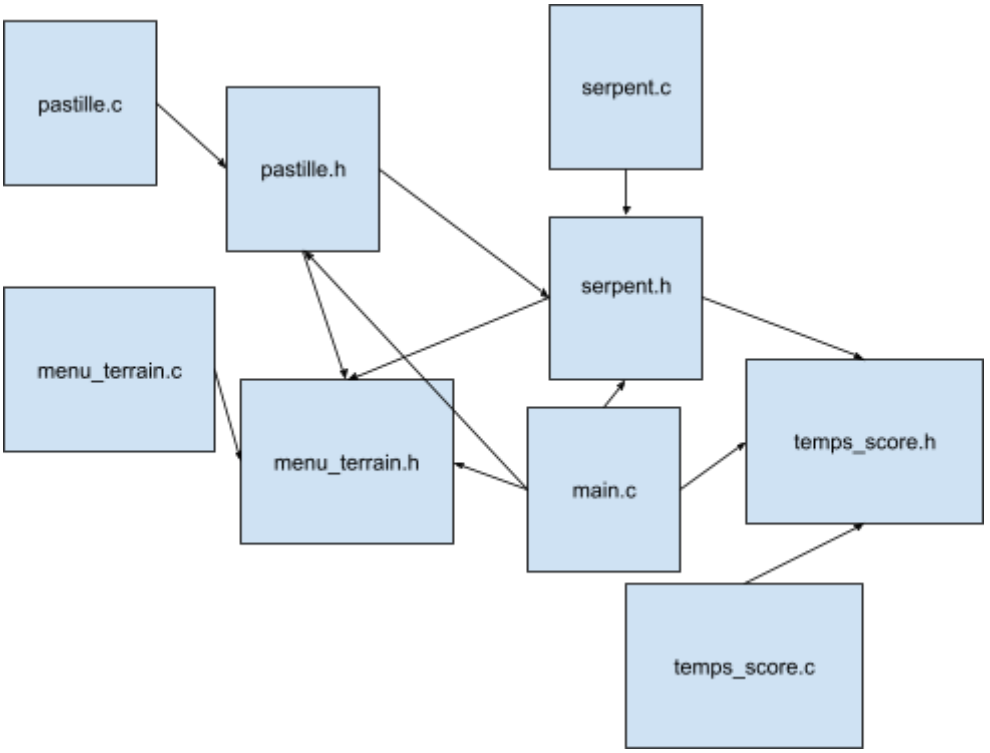
Deplacer_serpent permet au serpent de se déplacer. Mort_Serpent permet au serpent de mourir.

Le fichier menu_terrain.c contient les fonctions Terrain, Contour_Terrain, start , pause et perdu. Le fichier a pour but de créer le terrain, ces contours et les différents menus.

Le fichier pastille.c contient trois fonctions. Ces fonctions sont Création_Pastille, Afficher_Pastille, Validation_Coordonné. Le fichier a pour but de créer les pastilles (les collectables) sur le terrain, prendre l'image de la pomme et la charger sur les pastilles puis vérifier leur coordonnée pour qu'ils ne spawnent pas sur le serpent ou les contours.

Le fichier Temps_Score.c contient les fonctions attendre, Afficher_Temps, Afficher_Score. La fonction attendre sert au programme d'attendre un certain temps, Afficher_Score et Afficher_Temps affichent le temps et le score dans les contours et dans le menu Pause et Game Over.

Voici le diagramme des liens faites avec le Makefile:



Partie 4: Les données du serpent et ses transformations

On va expliquer les fonctions de serpent.c car ce sont les donnés qui composent le serpent:

Creation_Serpent:

La fonction creation_serpent crée un serpent de base avec une taille donnée en initialisant les positions de chaque segment du serpent.

Boucle de création du serpent :

- for (i = 0; i < taille; i++): Une boucle for itère taille fois pour créer chaque segment du serpent.

Initialisation des positions des segments :

- serpent[i].x = (COLONNES / 2) * TAILLE_CASE; :
  Initialise la coordonnée x du segment du serpent à la moitié du nombre de colonnes multiplié par la taille d'une case (TAILLE_CASE).
- serpent[i].y = (LIGNES / 2) * TAILLE_CASE; :
  Initialise la coordonnée y du segment du serpent à la moitié du nombre de lignes multiplié par la taille d'une case (TAILLE_CASE).

En résumé, la fonction creation_serpent crée un serpent de base en initialisant les positions de chaque segment du serpent à la position centrale de la grille. La taille du serpent est spécifiée par le paramètre taille. Le serpent est un tableau dynamiquement créé et chacun de ses éléments est créé via la structure “Segment”.

Afficher_Serpent_Haut:

La fonction afficher_Serpent_Haut affiche le serpent avec une taille donnée lorsque la tête du serpent vise vers le haut de l'écran.

Vérification de la rotation vers le haut :

- if (rotation == 2) { ... }: Vérifie si la rotation est égale à 2, indiquant que la tête du serpent vise vers le haut.

Affichage de la tête du serpent :

- ChargerImage("Image/Serpent_Tete_Haut.png",serpent[0].x, serpent[0].y, 0, 0, TAILLE_CASE, TAILLE_CASE); :
  Charge une image représentant la tête du serpent pointant vers le haut à la position de la première partie du serpent.

Affichage du corps du serpent :

- Une boucle for parcourt les segments du corps du serpent à partir de l'indice 1 jusqu'à la taille-1.
- Si le serpent a une rotation horizontale (les segments adjacents ont la même coordonnée y), alors il affiche une image de corps horizontal. Sinon, il affiche une image de corps vertical.

Affichage de la queue du serpent :

- RemplirRectangle(serpent[taille-1].x,serpent[taille-1].y, TAILLE_CASE, TAILLE_CASE); :
  Remplit le rectangle à la position de la queue du serpent pour éviter une superposition d'un bout du corps avec la queu.
- Selon les coordonnées du corps avant la queu et la queu, il charge une image représentant la queue du serpent pointant dans la direction appropriée (haut, bas, gauche, ou droite).

En résumé, la fonction afficher_Serpent_Haut affiche le serpent avec une tête pointant vers le haut, en utilisant une image spécifique pour la tête, le corps et la queue du serpent. La direction de chaque partie est déterminée par la variable rotation et les coordonnées avec l'élément précédent.

Afficher_Serpent_Bas:

La fonction afficher_Serpent_Bas affiche le serpent avec une taille donnée lorsque la tête du serpent vise vers le bas de l'écran.

Vérification de la rotation vers le bas :

- if (rotation == -2) { ... }: Vérifie si la rotation est égale à -2, indiquant que la tête du serpent vise vers le bas.

Affichage de la tête du serpent :

- ChargerImage("Image/Serpent_Tete_Bas.png",serpent[0].x, serpent[0].y, 0, 0, TAILLE_CASE, TAILLE_CASE); :
  Charge une image représentant la tête du serpent pointant vers le bas à la position de la première partie du serpent.

Affichage du corps du serpent :

- Une boucle for parcourt les segments du corps du serpent à partir de l'indice 1 jusqu'à la taille-1.
- Si le serpent a une rotation horizontale (les segments adjacents ont la même coordonnée y), alors il affiche une image de corps horizontal. Sinon, il affiche une image de corps vertical.

Affichage de la queue du serpent :

- RemplirRectangle(serpent[taille-1].x,serpent[taille-1].y, TAILLE_CASE, TAILLE_CASE); :
  Remplit le rectangle à la position de la queue du serpent pour éviter une superposition d'un bout du corps avec la queu..
- Selon les coordonnées du corps avant la queu et la queu, il charge une image représentant la queue du serpent pointant dans la direction appropriée (haut, bas, gauche, ou droite).

En résumé, la fonction afficher_Serpent_Bas affiche le serpent avec une tête pointant vers le bas, en utilisant une image spécifique pour la tête, le corps et la queue du serpent. La direction de chaque partie est déterminée par la variable rotation et les coordonnées avec l'élément précédent.

Afficher_Serpent_Droite:

La fonction afficher_Serpent_Droite affiche le serpent avec une taille donnée lorsque la tête du serpent vise vers la droite de l'écran.

Vérification de la rotation vers la droite :

- if (rotation == 1) { ... }: Vérifie si la rotation est égale à 1, indiquant que la tête du serpent vise vers le droite.

Affichage de la tête du serpent :

- ChargerImage("Image/afficher_Serpent_Droite.png",serpent[0].x, serpent[0].y, 0, 0, TAILLE_CASE, TAILLE_CASE); :
  Charge une image représentant la tête du serpent pointant vers le droite à la position de la première partie du serpent.

Affichage du corps du serpent :

- Une boucle for parcourt les segments du corps du serpent à partir de l'indice 1 jusqu'à la taille-1.
- Si le serpent a une rotation horizontale (les segments adjacents ont la même coordonnée y), alors il affiche une image de corps horizontal. Sinon, il affiche une image de corps vertical.

Affichage de la queue du serpent :

- RemplirRectangle(serpent[taille-1].x,serpent[taille-1].y, TAILLE_CASE, TAILLE_CASE); :

Remplit le rectangle à la position de la queue du serpent pour éviter une superposition d'un bout du corps avec la queu..
- Selon les coordonnées du corps avant la queu et la queu, il charge une image représentant la queue du serpent pointant dans la direction appropriée (haut, bas, gauche, ou droite).

En résumé, la fonction afficher_Serpent_Droite affiche le serpent avec une tête pointant vers la droite, en utilisant une image spécifique pour la tête, le corps et la queue du serpent. La direction de chaque partie est déterminée par la variable rotation et les coordonnées avec l'élément précédent.

Afficher_Serpent_Gauche:

La fonction Afficher_Serpent_Gauche affiche le serpent avec une taille donnée lorsque la tête du serpent vise vers la gauche de l'écran.

Vérification de la rotation vers la gauche :
- if (rotation == -1) { ... }: Vérifie si la rotation est égale à -1, indiquant que la tête du serpent vise vers le droite.

Affichage de la tête du serpent :
- ChargerImage("Image/Afficher_Serpent_Gauche.png",serpent[0].x , serpent[0].y, 0, 0, TAILLE_CASE, TAILLE_CASE); :
  Charge une image représentant la tête du serpent pointant vers le gauche à la position de la première partie du serpent.

Affichage du corps du serpent :
- Une boucle for parcourt les segments du corps du serpent à partir de l'indice 1 jusqu'à la taille-1.
- Si le serpent a une rotation horizontale (les segments adjacents ont la même coordonnée y), alors il affiche une image de corps horizontal. Sinon, il affiche une image de corps vertical.

Affichage de la queue du serpent :
- RemplirRectangle(serpent[taille-1].x,serpent[taille-1].y, TAILLE_CASE, TAILLE_CASE); :
  Remplit le rectangle à la position de la queue du serpent pour éviter une superposition d'un bout du corps avec la queu..
- Selon les coordonnées du corps avant la queu et la queu, il charge une image représentant la queue du serpent pointant dans la direction appropriée (haut, bas, gauche, ou droite).

En résumé, la fonction Afficher_Serpent_Gauche affiche le serpent avec une tête pointant vers la gauche, en utilisant une image spécifique pour la tête, le corps et la queue du serpent. La direction de chaque partie est déterminée par la variable rotation et les coordonnées avec l'élément précédent.

Afficher_Serpent:

La fonction Afficher_Serpent est une fonction complète pour afficher le serpent dans différentes directions en utilisant différentes fonctions spécifiques pour chaque direction.

Boucle de remplissage du serpent :

- for (i = 0; i < taille; i++) { ... }:
  Une boucle for parcourt les segments du serpent.

Remplissage du rectangle pour la tête et la queue :

- ChoisirCouleurDessin(CouleurParComposante(0, 217, 87)); :
  Choix de la couleur de dessin du fond (vert).
- RemplirRectangle(serpent[0].x, serpent[0].y, TAILLE_CASE, TAILLE_CASE); :
  Remplit le rectangle à la position de la tête du serpent de la couleur choisie.
- RemplirRectangle(serpent[taille-1].x, serpent[taille-1].y, TAILLE_CASE, TAILLE_CASE); :
  Remplit le rectangle à la position de la queue du serpent par la couleur choisie.

Appels aux fonctions d'affichage spécifiques pour chaque direction :

- afficher_Serpent_Haut(serpent, taille, rotation); :
  Appel à la fonction qui affiche le serpent lorsque la tête vise vers le haut.
- afficher_Serpent_Bas(serpent, taille, rotation); :
  Appel à la fonction qui affiche le serpent lorsque la tête vise vers le bas.
- afficher_Serpent_Droite(serpent, taille, rotation); :
  Appel à la fonction qui affiche le serpent lorsque la tête vise vers la droite.
- afficher_Serpent_Gauche(serpent, taille, rotation); :

Appel à la fonction qui affiche le serpent lorsque la tête vise vers la gauche.

En résumé, la fonction Afficher_Serpent remplit les rectangles représentant la tête et la queue du serpent en utilisant une couleur verte. Ensuite, elle appelle des fonctions spécifiques pour afficher le corps du serpent dans différentes directions en utilisant différentes images. La fonction prend en compte la rotation de la tête du serpent pour déterminer la direction dans laquelle le serpent doit être affiché.

Deplacer_Serpent:

La fonction Deplacer_Serpent permet au serpent de se déplacer dans une direction donnée.

Boucle de mise à jour des positions du serpent :

- for (i = *longueurSerpent - 1; i > 0; i--):
  Une boucle for parcourt les segments du serpent, en commençant par la fin du serpent et en déplaçant chaque segment vers l'élément d'indice précédent et gardant les coordonnées de la tête à l'indice 0.

Mise à jour de la position de la tête du serpent :

- La position de la tête du serpent (serpent[0]) est mise à jour en fonction de la direction spécifiée.
- switch (direction) { ... }:
  Utilisation d'une structure switch pour déterminer la direction du déplacement pour ensuite mettre à jour les coordonnées en conséquence.
- case 1: :
  Déplacement vers la droite (ajout de 1 à la valeur x de la tête).
- case -1: :
  Déplacement vers la gauche (soustraction de 1 à la valeur x de la tête).
- case 2: :
  Déplacement vers le haut (soustraction de 1 à la valeur y de la tête).
- case -2: :

Déplacement vers le bas (ajout de 1 à la valeur y de la tête).

Affichage du serpent mis à jour :

- Afficher_Serpent(serpent, *longueurSerpent, direction); :
  Appel à la fonction Afficher_Serpent pour mettre à jour l'affichage du serpent avec sa nouvelle position.

Attente d'un temps défini (vitesse_serpent) :

- Attendre(vitesse_serpent); :
  La fonction attend un certain temps défini par vitesse_serpent avant de continuer l'exécution. Cela contrôle la vitesse de déplacement du serpent.

En résumé, la fonction Deplacer_Serpent permet au serpent de se déplacer dans une direction donnée en mettant à jour les positions de ses segments et en actualisant l'affichage du serpent. La vitesse du serpent est contrôlée par la variable vitesse_serpent.

Mort_Serpent:

La fonction Mort_Serpent vérifie si la tête du serpent se trouve dans une zone hors limite ou entre en collision avec son propre corps. En cas de décès du serpent, la fonction effectue certaines actions, telles que l'affichage d'un rectangle noir à l'emplacement de la tête du serpent, une pause, l'exécution d'une fonction perdu, et la réinitialiosation des variables du jeux.

Vérification des limites de la zone de jeu :

- if (serpent[0].x < 5 * TAILLE_CASE || serpent[0].x >= (COLONNES * TAILLE_CASE) + 5 * TAILLE_CASE || serpent[0].y < CONTOURE_H * TAILLE_CASE || serpent[0].y >= (LIGNES * TAILLE_CASE) + 2 * TAILLE_CASE) { ... }:
  Vérifie si la tête du serpent est en dehors des limites de la zone de jeu. Si la condition est vraie, elle exécute des actions pour signaler la fin du jeu.

Affichage d'un carré noir à l'emplacement de la tête du serpent :

- ChoisirCouleurDessin(CouleurParNom("black")); :
  Choix de la couleur du dessin (noir).

- RemplirRectangle(serpent[0].x, serpent[0].y, TAILLE_CASE, TAILLE_CASE); :
  Remplit un rectangle noir à l'emplacement de la tête du serpent.

Pause avant d'exécuter d'autres actions :

- Attendre(1000); :
  Pause de 1000 millisecondes (1 seconde) pour que le joueur puisse savoir pourquoi il est mort.

Exécution de la fonction perdu et mise à jour des variables de contrôle du jeu :

- perdu(texte, score, minutes, secondes, rejouer); :
  Exécution de la fonction perdu pour effectuer des actions spécifiques en cas de défaite du joueur.
- *gameover = 0; :
  Mise à jour de la variable gameover à 0 pour signaler la fin du jeu.

Vérification de collision avec le corps du serpent :

- Une deuxième partie de la fonction parcourt le corps du serpent (à partir de l'indice 1) pour vérifier s'il y a une collision avec la tête du serpent.
- Si une collision est détectée, elle exécute des actions similaires à celles décrites ci-dessus.

En résumé, la fonction Mort_Serpent gère les conditions de mort du serpent, telles que les sorties des limites du terrain ou les collisions avec son propre corps. Elle effectue des actions appropriées en cas de décès du serpent, comme l'affichage d'un rectangle noir, une pause, l'exécution de la fonction perdu, et la réinitialisation des variables de contrôle du jeu.

Partie 5: Conclusion

Avis de Marvin:
J'aurais pensé que faire ce jeu aurait été plus simple, qu'on l'aurait conclue en 2-3 jours de travail.
Et bah j'avais tort, ils nous a pris plus de temps que prévu et la difficulté m'a vraiment surpris, je pense que c'est d'ailleurs aussi le cas dans d'autres groupes étant donné qu'on a tous eu besoin d'une semaine supplémentaire.
Une autre tâche qui m'a pris beaucoup de temps était la conception des menus, du serpent et des pommes. J'ai dessiné cela à partir d'une vieille tablette pas adaptée car je n'avais aucun PC (la tablette a rendue l'âme RIP). Il n'y aurait rien eu de compliqué là dedans si on pouvait modifier les images une fois finis. J'ai du refaire plusieurs fois les menus car j'avais oublié des détailles comme un idiot. Sinon la création du corps a été plutôt simple.
En dehors de cela la rédaction de ce document était plutôt simple
Je me souviendrai de ce projet comme quelque chose qui a volé mon week-end 2 fois.

Avis de Patrick:
J'aurais moi aussi pensé que faire ce jeu aurait été simple, qu'on l'aurait conclue en moins d'une semaine de travail.
Cependant j'avais tort, il nous a pris plus de temps que prévu et la difficulté m'a surpris, je pense que c'est d'ailleurs aussi le cas dans d'autres groupes étant donné qu'on a tous eu besoin d'une semaine supplémentaire.
Une autre tâche qui m'a pris beaucoup de temps c'était l'affichage des menus, du serpent et des pommes. J'ai du refaire plusieurs fois le code d'affichage pour les menus et surtout pour le serpent car j'avais oublié des détailles comme un idiot ou devait réajuster la taille pour être conforme à notre jeux.
J'ai beaucoup apprécié l'environnement dans lequel j'ai travaillé car moi et d'autres nous nous entraidons sur certains problèmes et pour les fou rire lors des essaies avec les autres binômes.