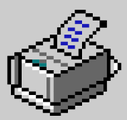






Snike

Press space to start



11:11PM

Qui sommes-nous?

NOTRE ÉQUIPE

SNIKE

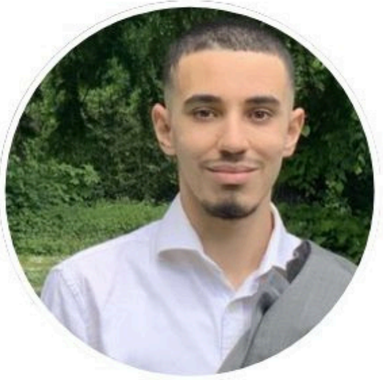
Présentation de l'équipe

BOUAZIZ
NESSIM

Chef de projet, gère la sauvegarde

ATTAR
RAYANE

Programmeur: S'occuper de la gestion de la vitesse



YOUS
ADEL

Programmeur: Gestion du nombres de pommes et de la longueur du serpent

1. BOUAZIZ Nessim

Chef de projet

Nessim, en plus d'être notre chef de projet, a également pris en charge la gestion de la programmation du système de sauvegarde pour notre jeu 'snake'. Il s'est assuré que chaque membre de l'équipe dispose des tâches nécessaires pour avancer, tout en programmant lui-même cette fonctionnalité essentielle. Grâce à son expertise technique et son organisation impeccable, il a non seulement coordonné le développement du projet, mais aussi implémenté une solution de sauvegarde fiable. Son travail a permis à l'équipe de progresser de manière structurée, garantissant ainsi le succès du jeu.

2 ATTAR Rayane

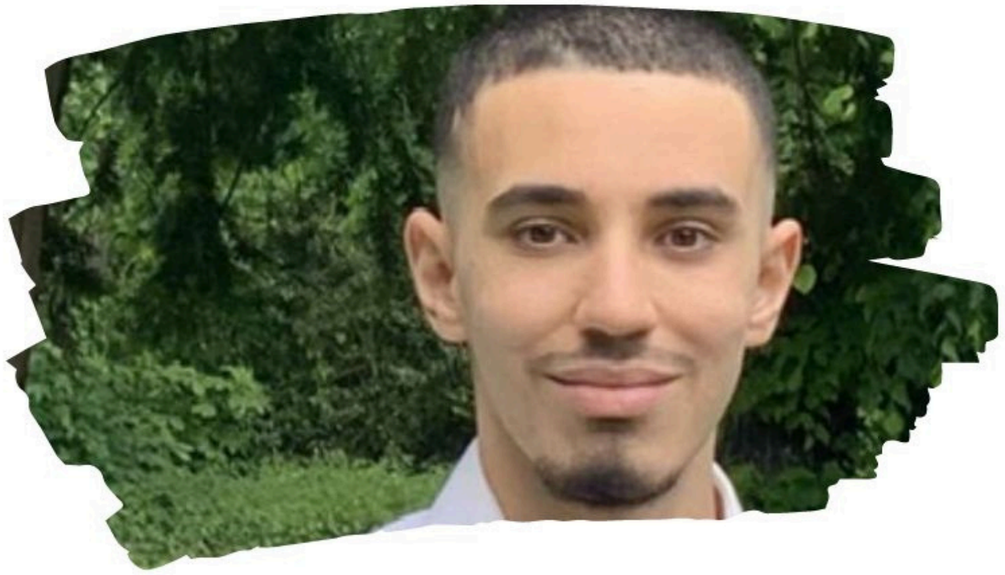
Programmeur:
Gestion de vitesse

Rayane est le programmeur en charge de la gestion de la vitesse du serpent dans notre jeu 'snake'. Grâce à ses compétences exceptionnelles et sa maîtrise du langage C, il a conçu un système fluide permettant aux joueurs de modifier la vitesse de déplacement du serpent en temps réel. Sa rigueur et sa passion pour le développement se reflètent dans chaque ligne de code qu'il produit. De plus, Rayane excelle dans l'optimisation des performances du jeu, assurant ainsi une expérience de jeu fluide et sans accroc. Son expertise technique et son engagement font de lui un atout indispensable pour notre équipe.

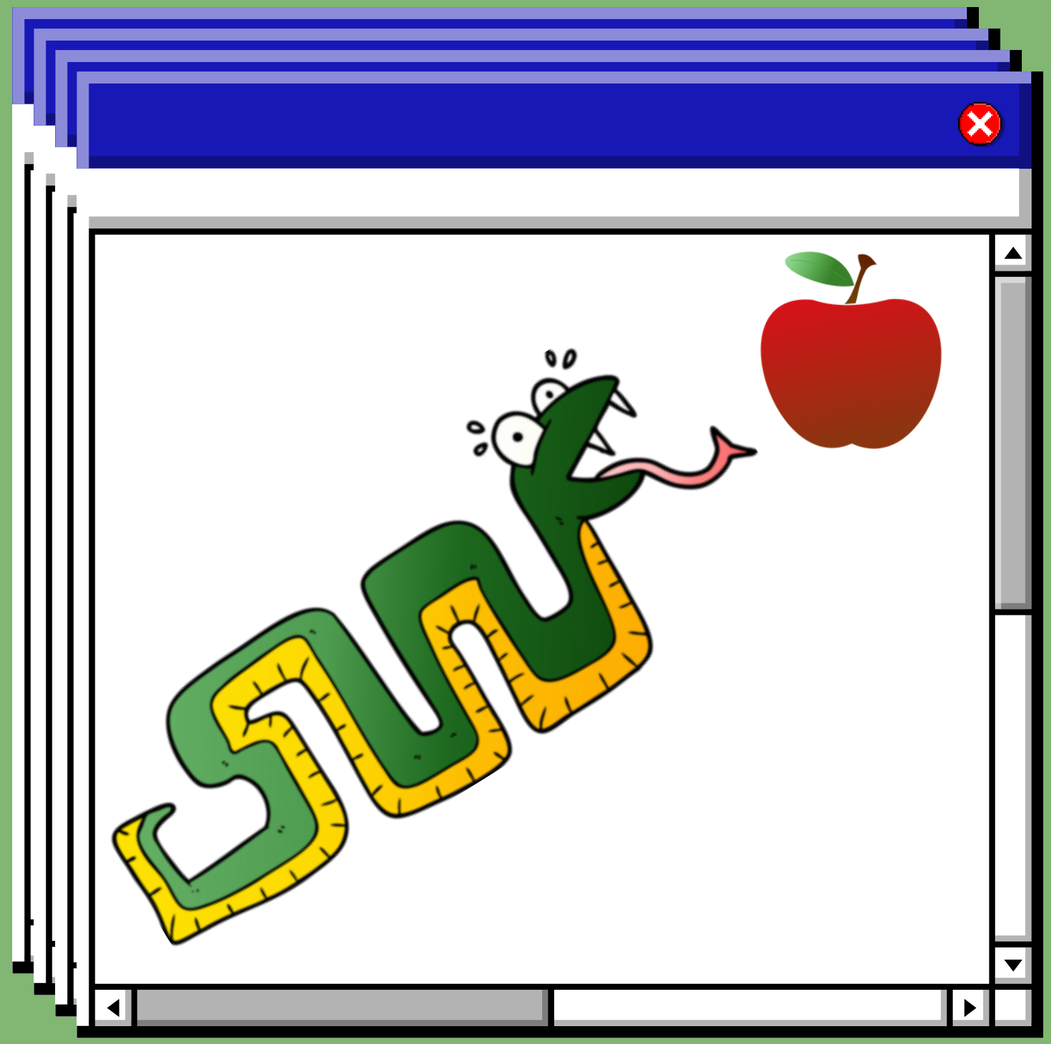
3. YOUS Adel

Programmeur:
Système des 3 pommes, et de la taille du serpent

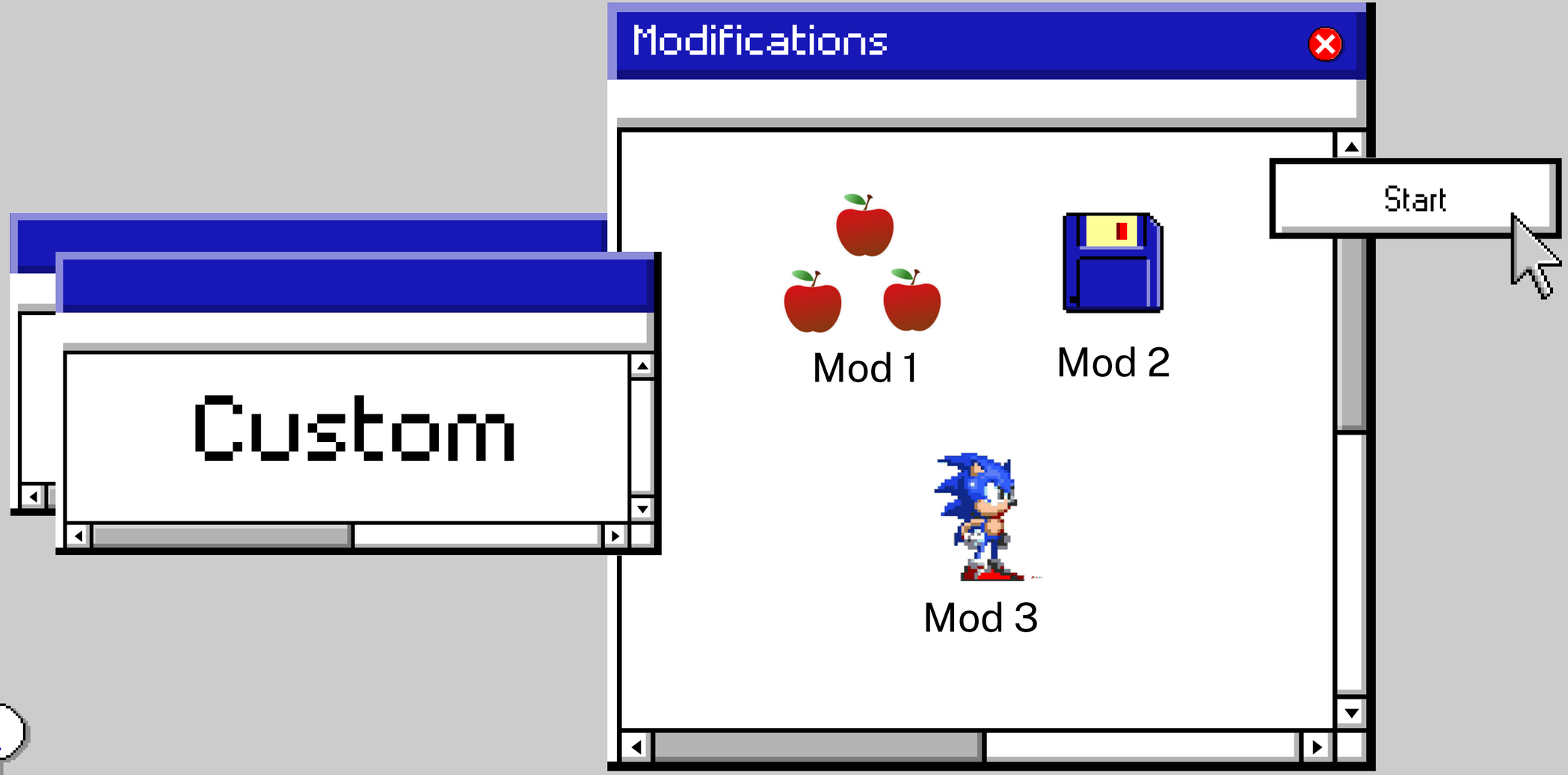
Adel est le programmeur en charge de l'apparition des trois pommes dans notre jeu 'snake', ainsi que de la gestion de la taille du serpent en fonction des pommes avalées. Bien qu'il n'ait pas encore atteint le niveau technique de ses camarades, il a choisi de se concentrer sur la conception et l'optimisation de ces aspects du jeu. Grâce à cette approche, il apporte une perspective unique en veillant à l'équilibre entre gameplay et esthétique, tout en renforçant l'expérience utilisateur. Adel joue ainsi un rôle essentiel dans la réussite de notre projet, en apportant une contribution précieuse qui dépasse les compétences purement techniques.



C'est quoi Snake ?



custom • custom • custom • custom • custom • custom • custom • custom • custom • custom • custom • custom • custom • custom • custom

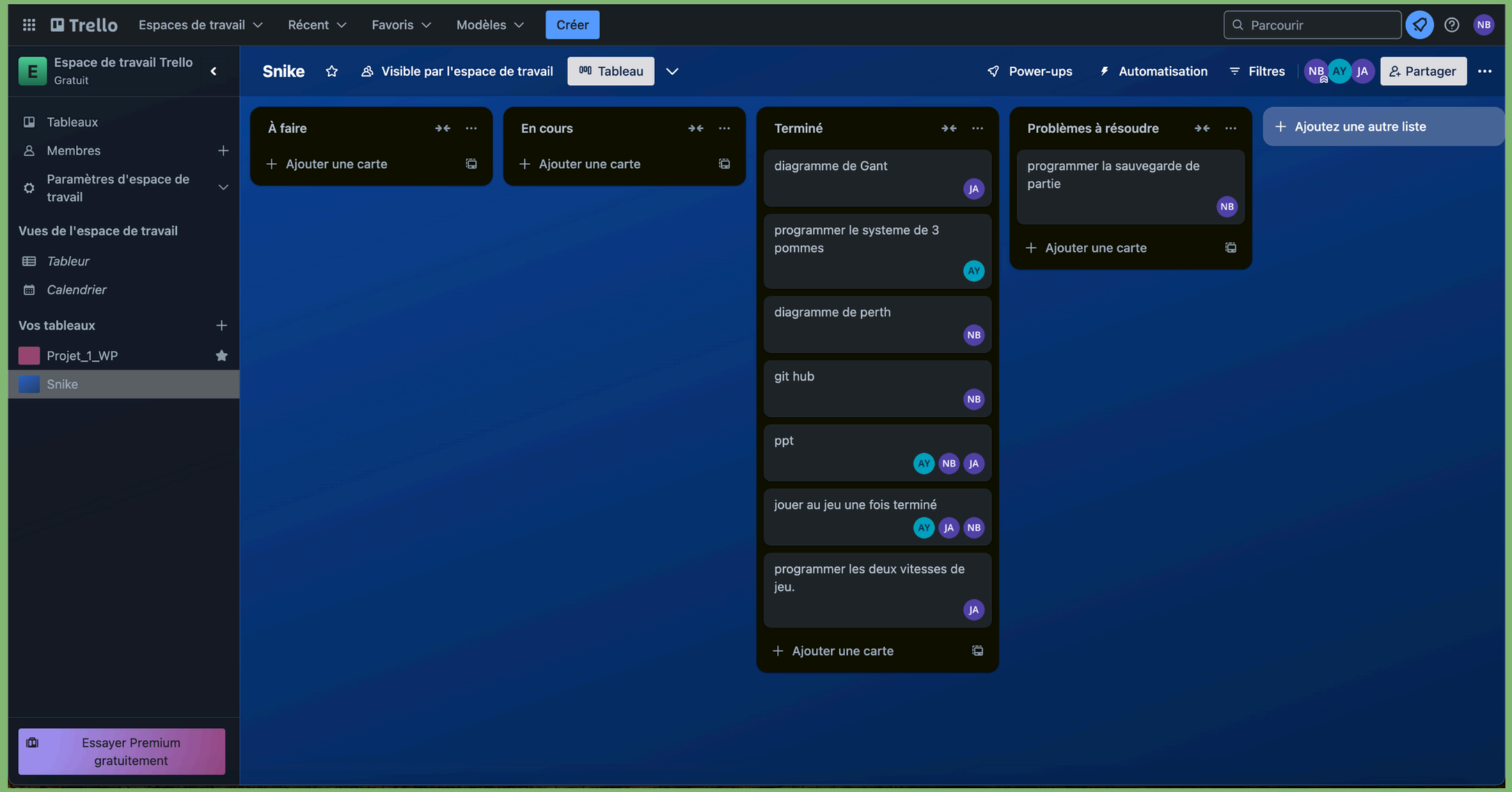


Timeline du projet

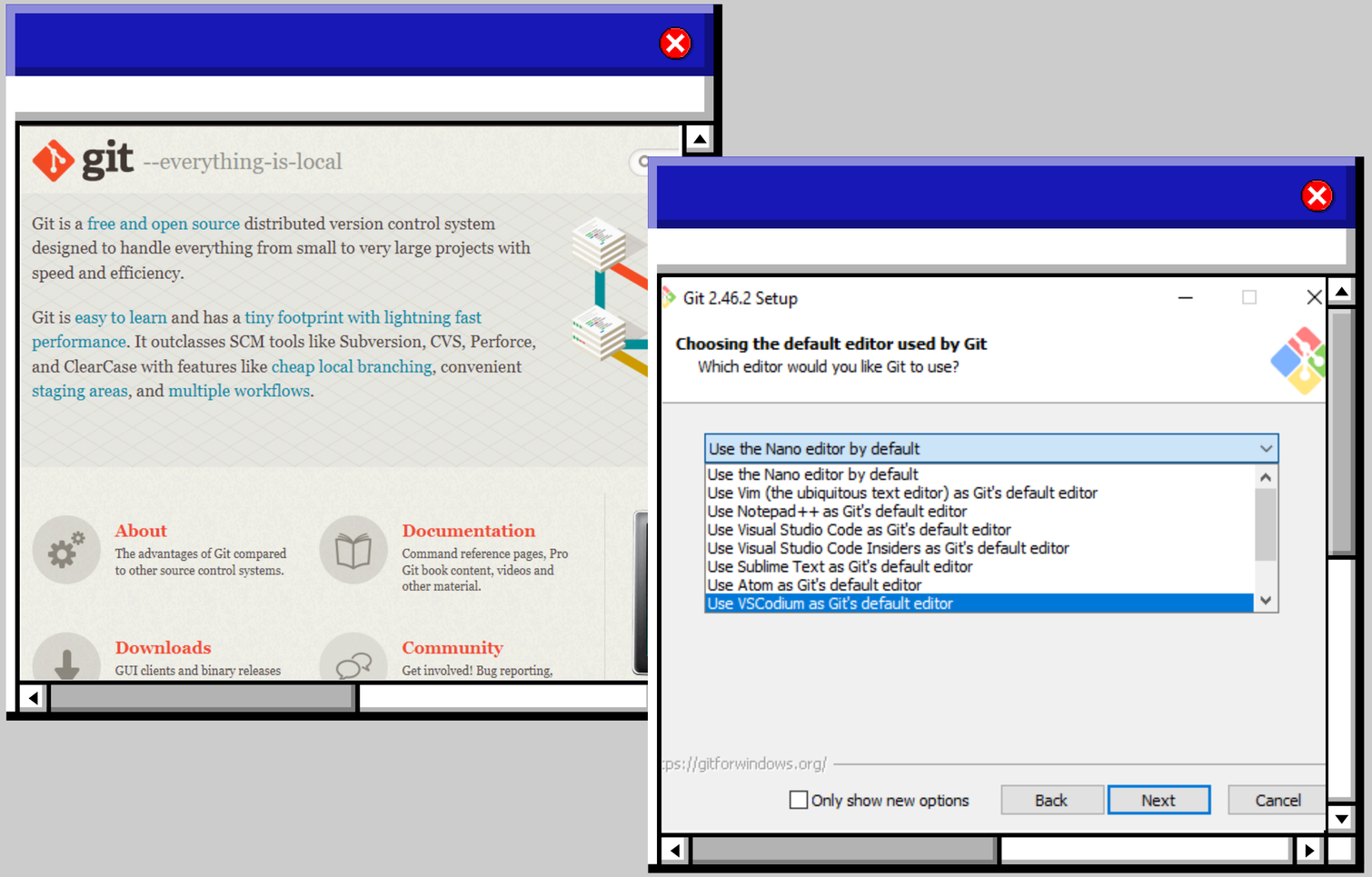
DIAGRAMME DE GANTT

Back to custom Page

Trello


Trello
Espaces de travail
Récent
Favoris
Modèles
Créer
Parcourir
NB
Espace de travail Trello
Gratuit
Tableaux
Membres
Paramètres d'espace de travail
Vues de l'espace de travail
Tableur
Calendrier
Vos tableaux
Projet_1_WP
Snike
Essayer Premium gratuitement
Snike
Visible par l'espace de travail
Tableau
Power-ups
Automatisation
Filtres
AY
JA
Partager
À faire
Ajouter une carte
En cours
Ajouter une carte
Terminé
diagramme de Gant
JA
programmer le systeme de 3 pommes
AY
diagramme de perth
NB
git hub
NB
ppt
AY
NB
JA
jouer au jeu une fois terminé
AY
JA
NB
programmer les deux vitesses de jeu.
JA
Ajouter une carte
Problèmes à résoudre
programmer la sauvegarde de partie
NB
Ajouter une carte
Ajoutez une autre liste

Mise en place de Git

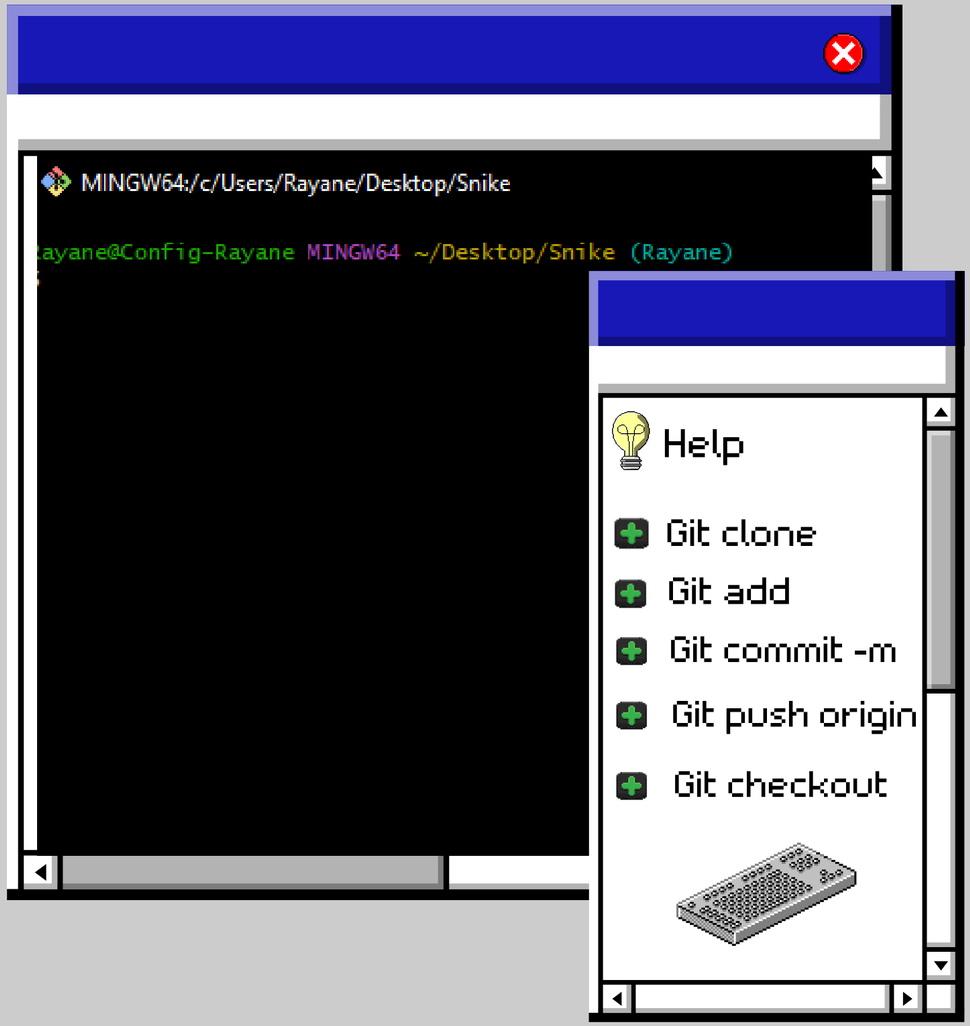


NOTE

Git est un système de gestion de versions distribué. En résumé, il permet aux développeurs de suivre les modifications apportées à leur code source, de collaborer efficacement sur des projets et de revenir à des versions antérieures si nécessaire.

Ok

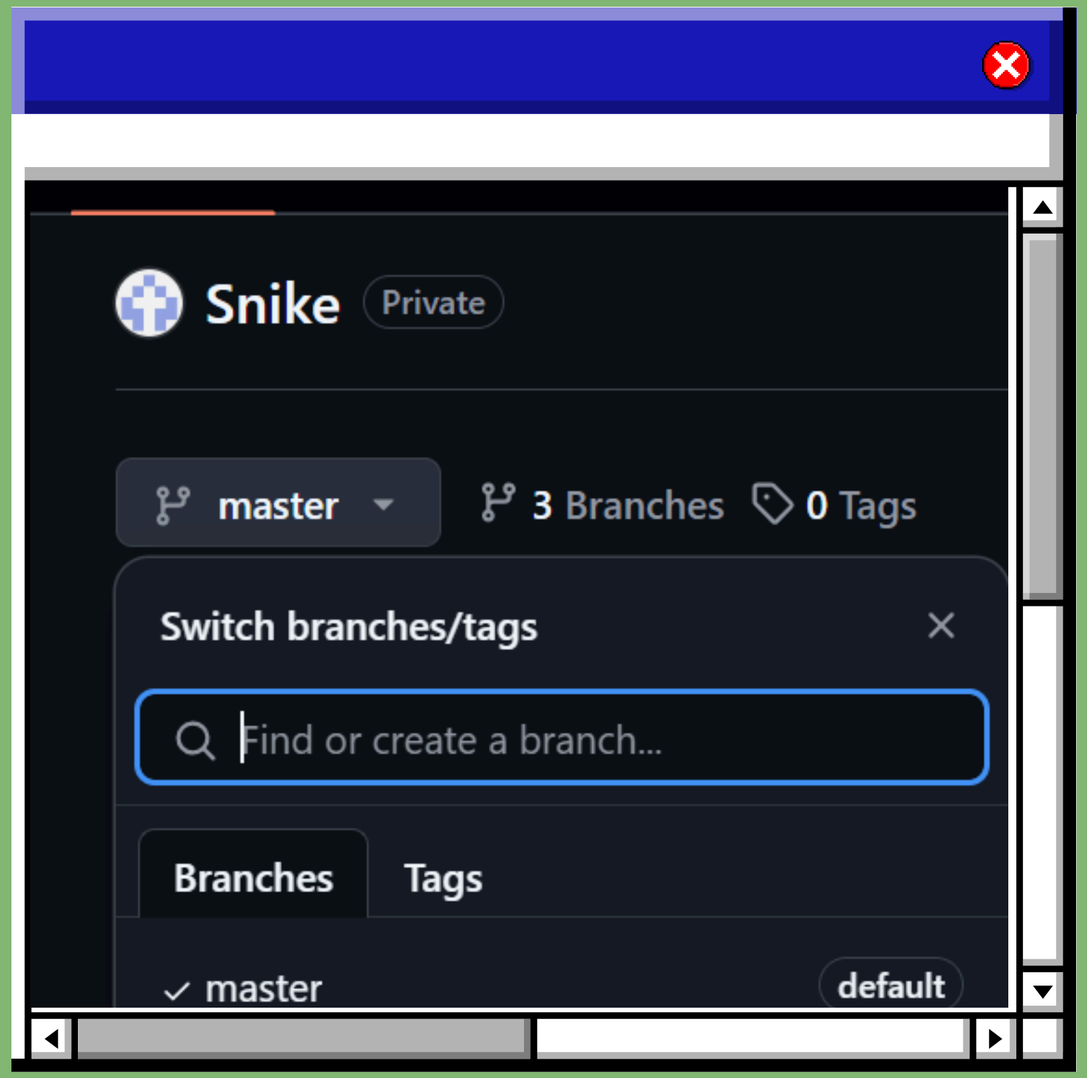
Principales commandes Git



NOTE

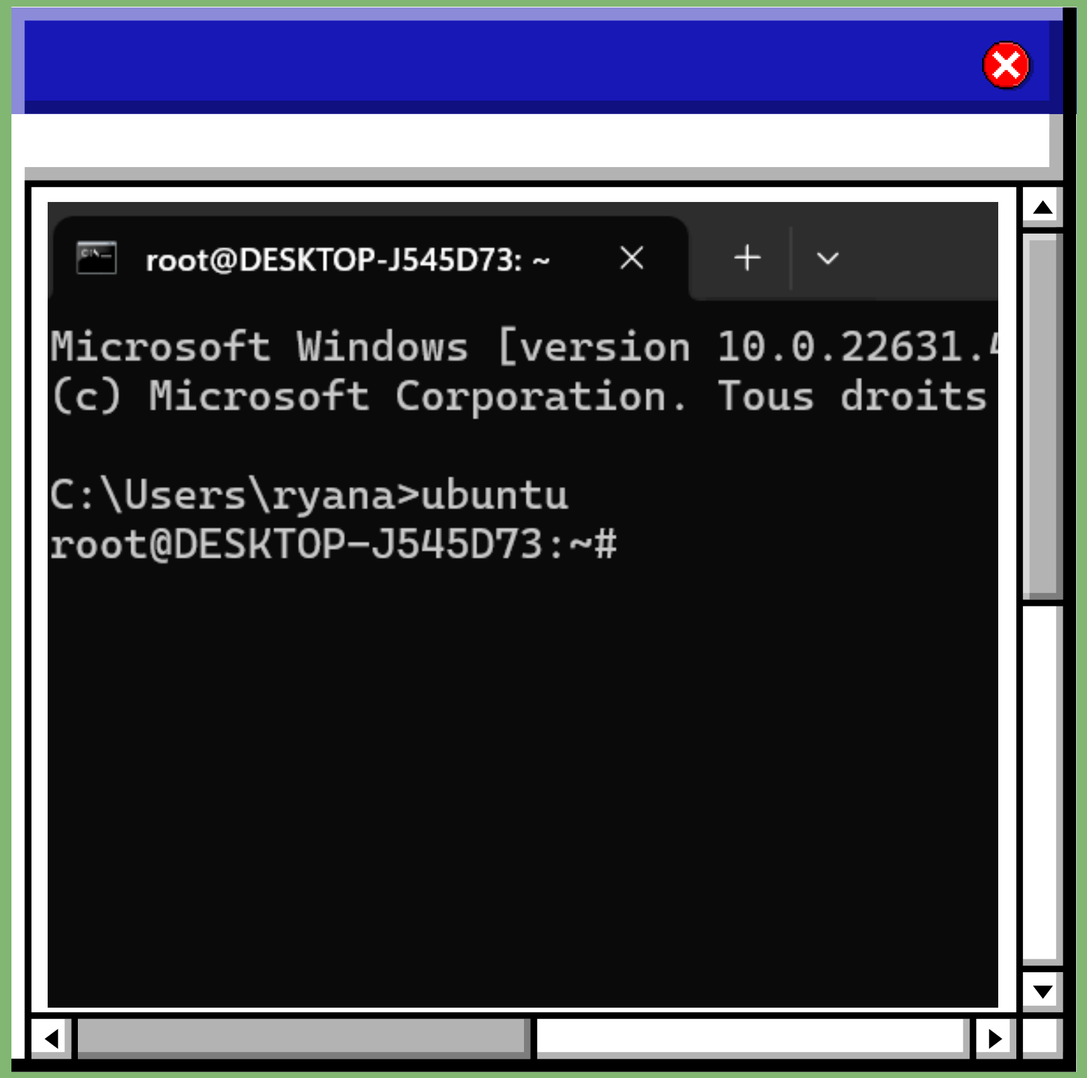
Git est un système de gestion de versions distribué. En résumé, il permet aux développeurs de suivre les modifications apportées à leur code source, de collaborer efficacement sur des projets et de revenir à des versions antérieures si nécessaire.

Ok

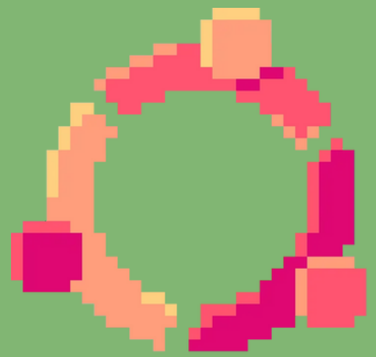


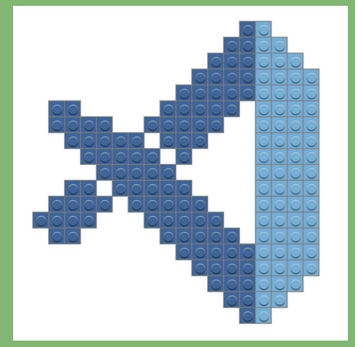
github

Back to custom Page

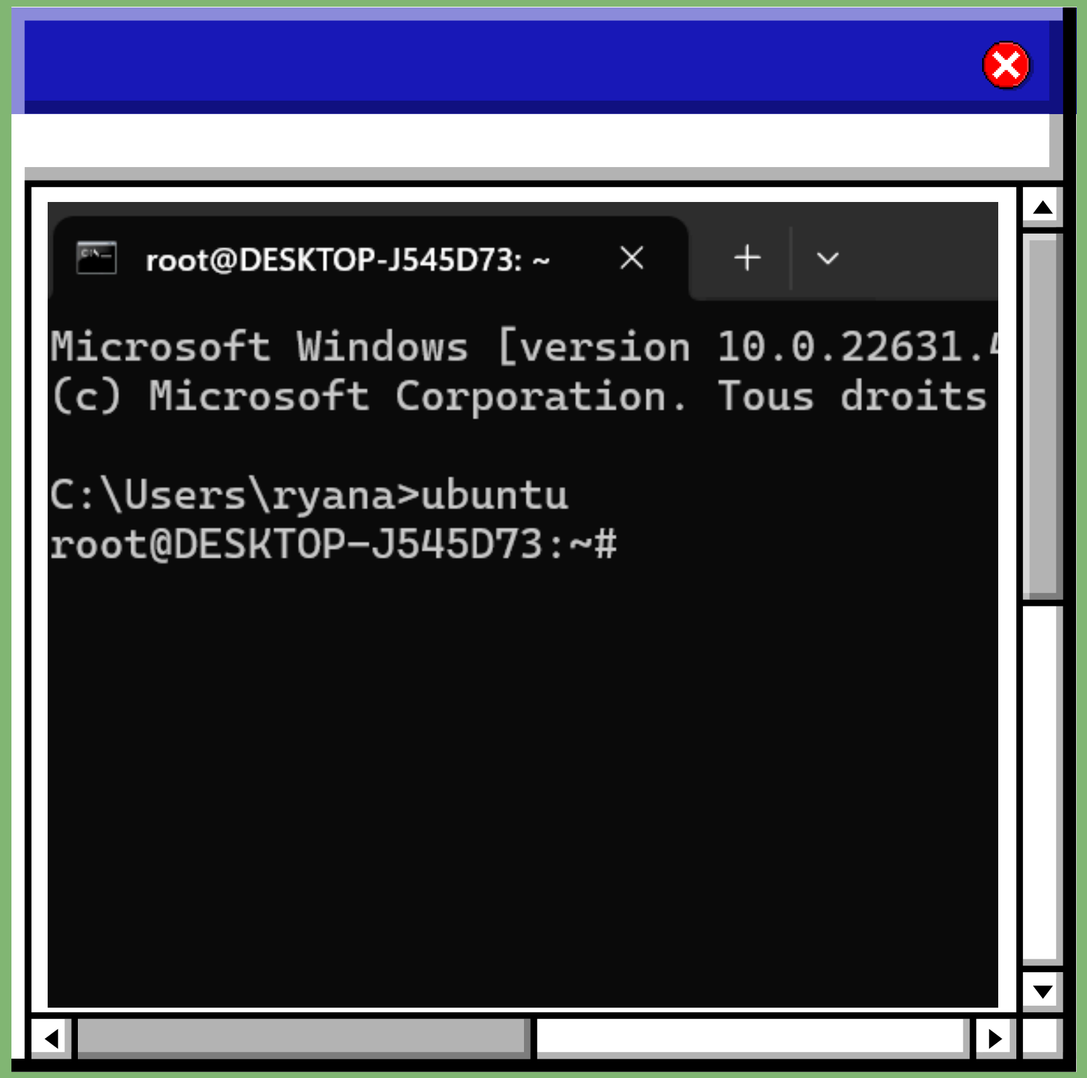


wsl et vscode





Back to custom Page



Compiler avec GCC les bibliothèques C

Back to custom Page

Mise en place de l'apparition des pommes 3 par 3 + serpent grandis toutes les pommes consommées

Partie Adeli

Création d'une fonction pour générer les pommes sur la grille

```c
void genererDuManger(char ** grille, int nbPommes) {
  for (int i = 0; i < nbPommes; i++) {
    int ligne = 0;
    int colonne = 0;
    int done = 0;

    // Génère des pommes jusqu'à ce qu'une case vide soit trouvée
    while (!done) {
      ligne = rand() % (nbLignes - 1);
      colonne = rand() % (nbColonnes - 1);
      if(grille[ligne][colonne] == CASE_VIDE) {
        grille[ligne][colonne] = DU_MANGER;
        done = 1; // Sortir de la boucle une fois la pomme placée
      }
    }
  }
}
```

Logique : Pour chaque pomme à générer (dans une boucle de 0 à nbPommes), un emplacement aléatoire sur la grille est choisi. Si cet emplacement est vide (indiqué par CASE_VIDE), une pomme (DU_MANGER) est placée à cet endroit.

Back to custom Page

Partie Adeli

Mise en place de l'apparition des pommes 3 par 3 + serpent grandis toutes les pommes consommées

Boucle afin de re générer 3 pommes a chaque fois que les 3 pommes sur la grille ont été mangé :

```
if (nbMange == POMMES_A_MANGER) { // Si 3 pommes ont été mangées
  genererDuManger(grille, POMMES_A_MANGER); // Générer 3 nouvelles pommes
  nbMange = 0; // Réinitialiser le compteur
}
```

Nouvelle génération de pommes : Chaque fois que le serpent a mangé trois pommes (déterminé par nbMange), trois nouvelles pommes sont générées sur la grille.

Partie Adeli

Mise en place de l'apparition des pommes 3 par 3 + serpent grandis toutes les pommes consommées

Dans la fonction main, tu appelles genererDuManger pour générer les pommes au début du jeu :

```
1   // Générer les 3 premières pommes
2   genererDuManger(grille, POMMES_A_MANGER);
```

Ici, POMMES_A_MANGER est défini comme 3, donc la fonction génère trois pommes sur la grille au démarrage.

Gestion de la consommation des pommes

```
1   // Si le serpent a mangé une pomme
2   if (*aMange) {
3     (*cptMange)++;  // Incrémente le compteur de pommes mangées
4   }
5
6   // Si le serpent a mangé 3 pommes, il peut grandir
7   if (*cptMange == 3) {
8     *cptMange = 0;  // Réinitialise le compteur
9   } else {
10    // Si le serpent n'a pas encore mangé 3 pommes, on supprime la queue
11    supprimerQueue(snake);
```

- Compteur de pommes mangées : cptMange est utilisé pour compter combien de pommes le serpent a mangées. Chaque fois que le serpent mange une pomme, cptMange est incrémenté.
- Vérification de la croissance : Si cptMange atteint trois, cela signifie que le serpent a mangé trois pommes, et on le réinitialise à zéro. Cela permettrait au serpent de grandir (le code pour gérer cela se trouve ailleurs dans ton programme).

Partie Rayane

```
touche = getch();

// Si une touche est appuyée, ajuster la vitesse si nécessaire
if (touche != ERR) {
  if (touche == 'z') {  // 'z' pour augmenter la vitesse
    delay = 100;  // vitesse maximale
  } else if (touche == 's') {  // 's' pour diminuer la vitesse
    delay = 500;  // vitesse minimale
  } else if (touche == ' ') {  // espace pour réinitialiser la vitess
    delay = 300;  // vitesse normale
  }
}

// Mise à jour du timeout avec le nouveau delay
timeout(delay);
```

Tronc principal de gestion de la vitesse

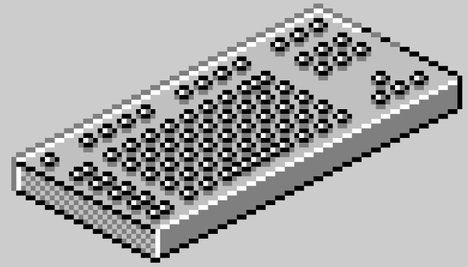
Back to custom Page

Partie Rayane

```c
initscr(); // Initialisation de ncurses
cbreak(); // Désactiver le buffer de ligne
noecho(); // Ne pas afficher les caractères saisis
keypad(stdscr, TRUE); // Activer les touches spéciales
srand(time(NULL)); // Initialiser le générateur de nombres aléatoires
getmaxyx(stdscr, nbLignes, nbColonnes); // Obtenir la taille de la fenêtre
```

Bibliothèque ncurses pour la gestion

Getch, timeout, delay...



Back to custom Page

Partie Nessou

Programmer la possibilité de sauvegarder et de charger la sauvegarde
1/2 SAUVEGARDER

```c
void sauvegarderJeu(unSnake snake, uneDirection direction) {
    FILE *f = fopen("save.txt", "w");
    if (f == NULL) {
        fprintf(stderr, "Erreur de sauvegarde du jeu.\n");
        return;
    }

    // Écrire les dimensions de la grille, la direction et le nombre de cases mangées
    fprintf(f, "%d %d %d %d %d\n", nbLignes, nbColonnes, direction.ligne, direction.colonne, nbCasesMangees);

    // Écrire l'état de la grille
    for (int i = 0; i < nbLignes; i++) {
        for (int j = 0; j < nbColonnes; j++) {
            fputc(grille[i][j], f);
        }
        fputc('\n', f); // nouvelle ligne pour chaque ligne de la grille
    }

    fclose(f);
}
```

Fonction sauvegarderJeu : Elle se trouve à partir de la ligne 202 dans notre code. Cette fonction est responsable de sauvegarder l'état actuel du jeu dans un fichier texte (save.txt).

- Paramètres :
  - unSnake snake : Le serpent actuel, qui est utilisé pour récupérer l'état de chaque cellule du serpent.
  - uneDirection direction : La direction actuelle du serpent (haut, bas, gauche, droite).
- Étapes de sauvegarde :
  - Fichier : La fonction ouvre le fichier save.txt en mode écriture ("w"), ce qui signifie que tout contenu précédent sera remplacé.
  - Sauvegarde des dimensions et direction : Elle écrit d'abord les dimensions de la grille (nbLignes et nbColonnes), la direction du serpent (direction.ligne, direction.colonne), et le nombre de cases mangées (nbCasesMangees).
  - Sauvegarde de la grille : Ensuite, elle parcourt la grille et sauvegarde son état, cellule par cellule, en écrivant chaque caractère (soit une partie du serpent, soit une pomme, soit une case vide).

Fermeture du fichier : Une fois la sauvegarde terminée, le fichier est fermé avec fclose.

Back to custom Page

2/2 Charger

Partie Nessou

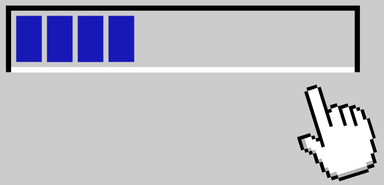
Fonction chargerJeu : Elle commence à partir de la ligne 221. Cette fonction permet de recharger une partie sauvegardée à partir du fichier save.txt.

- Paramètres :
  - unSnake *snake : Le serpent dans lequel on va charger l'état sauvegardé.
  - uneDirection *direction : La direction actuelle du serpent qu'on récupère depuis la sauvegarde.
- Étapes de chargement :
  - Ouverture du fichier : La fonction ouvre le fichier save.txt en mode lecture ("r"). Si le fichier n'existe pas ou ne peut pas être ouvert, elle affiche un message d'erreur et sort de la fonction.
  - Libération de la grille : Avant de charger la nouvelle grille, la mémoire de l'ancienne grille est libérée pour éviter les fuites de mémoire.
  - Lecture des dimensions, direction et cases mangées : Elle commence par lire les dimensions de la grille (nbLignes, nbColonnes), la direction du serpent (direction.ligne, direction.colonne), et le nombre de cases mangées (nbCasesMangees).
  - Allocation de la grille : Une fois les dimensions connues, la fonction réalloue la mémoire pour la grille.
  - Chargement de la grille : Elle lit ensuite la grille à partir du fichier, ligne par ligne, en ignorant les bordures.
  - Reconstruire le serpent : Le serpent est reconstruit en créant des cellules pour chaque partie du serpent (tête, corps), selon les positions enregistrées dans la grille.

```c
void chargerJeu(unSnake *snake, uneDirection *direction) {
    FILE *f = fopen("save.txt", "r");
    if (f == NULL) {
        fprintf(stderr, "Erreur de chargement du fichier de sauvegarde.\n");
        return;
    }

    // Libérer la mémoire actuelle de la grille
    for (int i = 0; i < nbLignes; i++) {
        free(grille[i]);
    }
    free(grille);

    // Lecture des dimensions, direction et cases mangées
    fscanf(f, "%d %d %d %d %d\n", &nbLignes, &nbColonnes, &direction->ligne, &direction->colonne, &nbCasesMangees);

    // Allocation de la grille
    grille = malloc(nbLignes * sizeof(char *));
    for (int i = 0; i < nbLignes; i++) {
        grille[i] = malloc(nbColonnes * sizeof(char));
    }

    // Lecture de la grille
    for (int i = 0; i < nbLignes; i++) {
        for (int j = 0; j < nbColonnes; j++) {
            grille[i][j] = fgetc(f);
        }
        fgetc(f); // pour sauter le caractère de nouvelle ligne
    }

    fclose(f);

    // Reconstruire le serpent en fonction de la grille
    // (logique similaire à celle décrite plus haut)
}
```

Logique générale du système Sauvegarde/Charge

Partie Nessou



Logique générale

Sauvegarde : Lorsque tu appuies sur la touche 'f', la fonction sauvegarderJeu enregistre dans un fichier :
La taille de la grille.
La direction actuelle du serpent.
Le nombre de pommes mangées.
L'état de la grille, c'est-à-dire la position de chaque élément (serpent, pommes, etc.).
Chargement : Lorsque tu appuies sur la touche 'l', la fonction chargerJeu lit le fichier save.txt pour :
Restaurer la grille.
Remettre le serpent et les pommes à leurs positions sauvegardées.
Reprendre le jeu à l'endroit où il avait été laissé.
Grâce à cette fonctionnalité, tu peux reprendre ta partie à n'importe quel moment en sauvegardant et en rechargeant l'état du jeu.

Utilisation dans le jeu

Dans ta boucle principale (ligne 296), tu as lié les actions de sauvegarde et de chargement à des touches spécifiques du clavier :

- Touche 'f' : Sauvegarde la partie actuelle en appelant la fonction sauvegarderJeu.
- Touche 'l' : Charge une partie sauvegardée en appelant la fonction chargerJeu.

```
if (touche == 'f') { // Exemple : si 'f' est pressé, sauvegarder le jeu
    sauvegarderJeu(snake, direction);
}

if (touche == 'l') { // Exemple : si 'l' est pressé, charger le jeu
    chargerJeu(&snake, &direction);
}
```

Bilan de compétences

Langage C

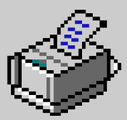
Maitrise du C et de ses différentes bibliothèques

Git/Github

Maitrise de l'environnement de travail du développeur

Compilateur

Maitrise des différents outils de compilation sur les différents os et selon bibliothèques incluses



Back to custom Page

Merci pour votre attention !!

Snike est la propriété de Mr Okacha Benahmed