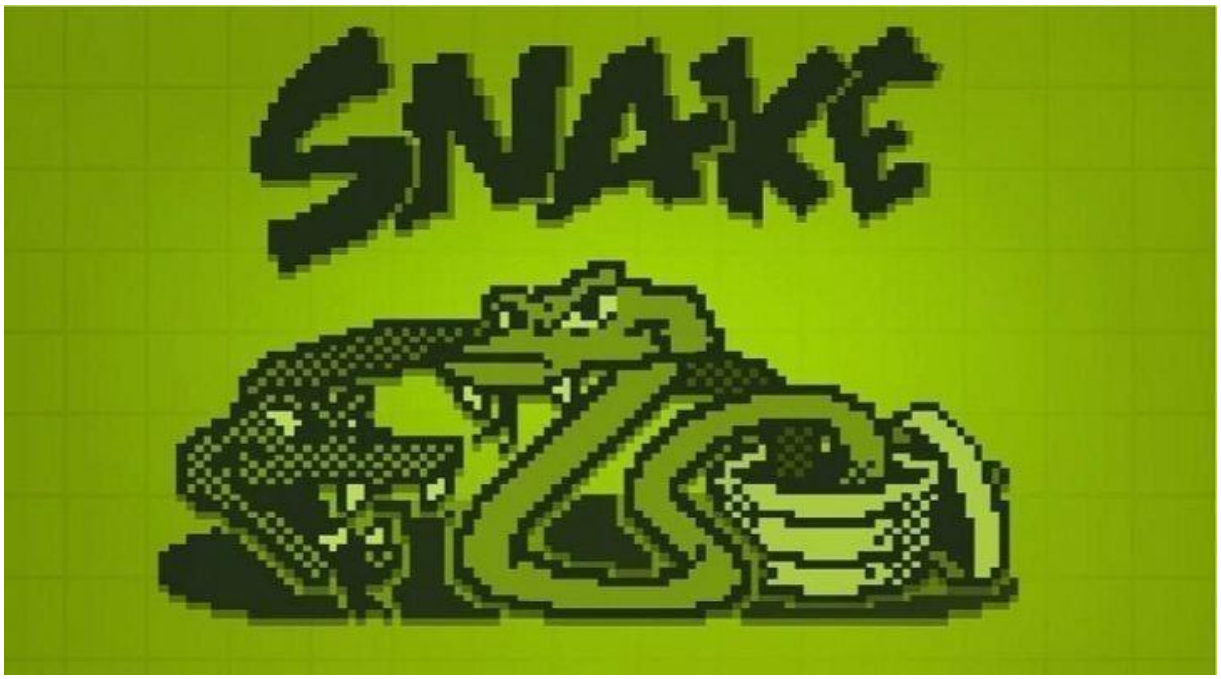
Projet personnel réalisé en cours de formation :

Développement d'un jeu de Snake en C

avec la bibliothèque SDL3



Initiation au langage C · Projet individuel

Sommaire

I. Introduction

Le Snake est un jeu vidéo classique, idéal pour illustrer les bases de la programmation en langage C :

- Manipulation de structures de données (tableaux …)
- Boucle de jeu et gestion du temps
- Gestion des entrées clavier (événements SDL)
- Rendu graphique bas niveau avec SDL3
- Lecture et écriture de fichiers binaires

> ***Objectif** : Développer un jeu de Snake jouable en C avec la bibliothèque SDL3, intégrant un menu, deux niveaux de difficulté, un système de score, ainsi qu'une sauvegarde et un chargement de partie.*

II. Analyse du besoin

2.1 Règles du jeu

Le Snake se joue sur une grille de 32 × 24 cases. Le joueur contrôle un serpent qui se déplace en continu :

- Le serpent avance automatiquement dans la direction courante.
- Le joueur change de direction avec les touches fléchées.
- Manger un fruit fait grandir le serpent d'un segment et incrémente le score.
- Le jeu se termine si le serpent touche un mur ou son propre corps.
- Il est impossible de faire demi-tour immédiat (direction opposée bloquée).

2.2 Fonctionnalités attendues

L'application doit :

- Afficher un menu permettant de choisir entre le mode Normal et le mode Difficile
- Faire se déplacer le serpent à une vitesse adaptée à la difficulté (120 ms / 60 ms par déplacement)
- Afficher le score en temps réel pendant la partie
- Afficher un écran « Game Over » avec le score final et la possibilité de relancer
- Permettre de sauvegarder la partie en cours (touche S) et de la recharger (touche L)

III. Développement en C

3.1 Structure du code

Le projet est contenu dans un seul fichier source C. Il s'organise autour de plusieurs structures de données et d'une machine à états qui pilote le déroulement du jeu.

Structure	Rôle
`Point`	Coordonnée (x, y) sur la grille de jeu.
`Snake`	Corps du serpent (tableau de Points), longueur, croissance en attente, direction courante et suivante.
`SaveData`	Regroupe toutes les données à persister : serpent, fruit, score, vitesse et difficulté.
`GameState`	Énumération des 3 états : STATE_MENU, STATE_GAME, STATE_GAME_OVER.

3.2 Logique de jeu

Trois fonctions principales gèrent la mécanique du jeu :

reset_snake() — Initialise le serpent au centre de la grille, d'une longueur de 4 segments, se déplaçant vers la droite.

spawn_fruit() — Place un fruit sur une case aléatoire non occupée par le serpent.

move_snake() — Calcule la prochaine position de la tête, vérifie les collisions avec les murs et le corps, fait avancer le serpent et gère la croissance si un fruit est mangé.

```
static bool move_snake(Snake *snake, Point fruit, bool *fruit_eaten) {
    // Applique la prochaine direction demandée
    snake->direction = snake->next_direction;

    // Calcule la future position de la tête
    Point next_head = snake->body[0];
    if (snake->direction == DIR_UP)    next_head.y -= 1;
    if (snake->direction == DIR_DOWN)  next_head.y += 1;
    if (snake->direction == DIR_LEFT)  next_head.x -= 1;
    if (snake->direction == DIR_RIGHT) next_head.x += 1;

    // Collision mur
    if (next_head.x < 0 || next_head.x >= GRID_WIDTH ||
        next_head.y < 0 || next_head.y >= GRID_HEIGHT)
        return false;
```

```
    // Collision corps → game over
    for (int i = 0; i < segments_to_check; i++)
        if (snake->body[i].x == next_head.x &&
            snake->body[i].y == next_head.y) return false;

    // Déplacement des segments et croissance éventuelle
    // ...
    return true;
}
```

3.3 Rendu graphique

Tout le rendu est fait manuellement avec SDL3, sans image externe ni police système. Chaque élément est dessiné par superposition de rectangles colorés :

- Le fond est un dégradé vertical calculé ligne par ligne avec une interpolation linéaire de couleur.
- Le serpent est rendu avec des effets de brillance et d'ombre simulés en transparence. La tête dispose d'yeux avec des pupilles qui suivent la direction.
- Le fruit possède un reflet, une ombre portée et une petite feuille verte, tous dessinés géométriquement.
- Le texte utilise une police bitmap maison : chaque caractère est encodé dans 7 octets, chaque bit indiquant si un pixel est allumé.

```
typedef struct {
    char ch;
    uint8_t rows[7]; // 7 lignes de pixels, 5 colonnes par bit
} FontGlyph;

// Exemple : le caractère 'A'
{'A', {14, 17, 17, 31, 17, 17, 17}}
//    0b01110  →  .###.
//    0b10001  →  #...#
//    0b10001  →  #...#
//    0b11111  →  #####
//    0b10001  →  #...#
//    0b10001  →  #...#
//    0b10001  →  #...#
```

3.4 Sauvegarde et chargement

Pendant une partie, le joueur peut appuyer sur S pour sauvegarder et sur L pour recharger. Les données sont écrites dans un fichier binaire save.dat avec fwrite() et relues avec fread().

Des vérifications de cohérence sont effectuées à la lecture (longueur du serpent dans les bornes, retour de fread vérifié) pour éviter les erreurs en cas de fichier corrompu.

```
// Sauvegarde
fwrite(&data->score,        sizeof(int),       1, f);
fwrite(&data->fruit,        sizeof(Point),     1, f);
fwrite(&data->move_delay_ms, sizeof(int),       1, f);
fwrite(&data->is_difficult,  sizeof(bool),      1, f);
fwrite(&s->length,           sizeof(int),       1, f);
fwrite(s->body,              sizeof(Point), s->length, f);

// Chargement avec vérification
if (fread(&s->length, sizeof(int), 1, f) != 1) goto fail;
if (s->length < 0 || s->length > MAX_SNAKE_LENGTH) goto fail;
```

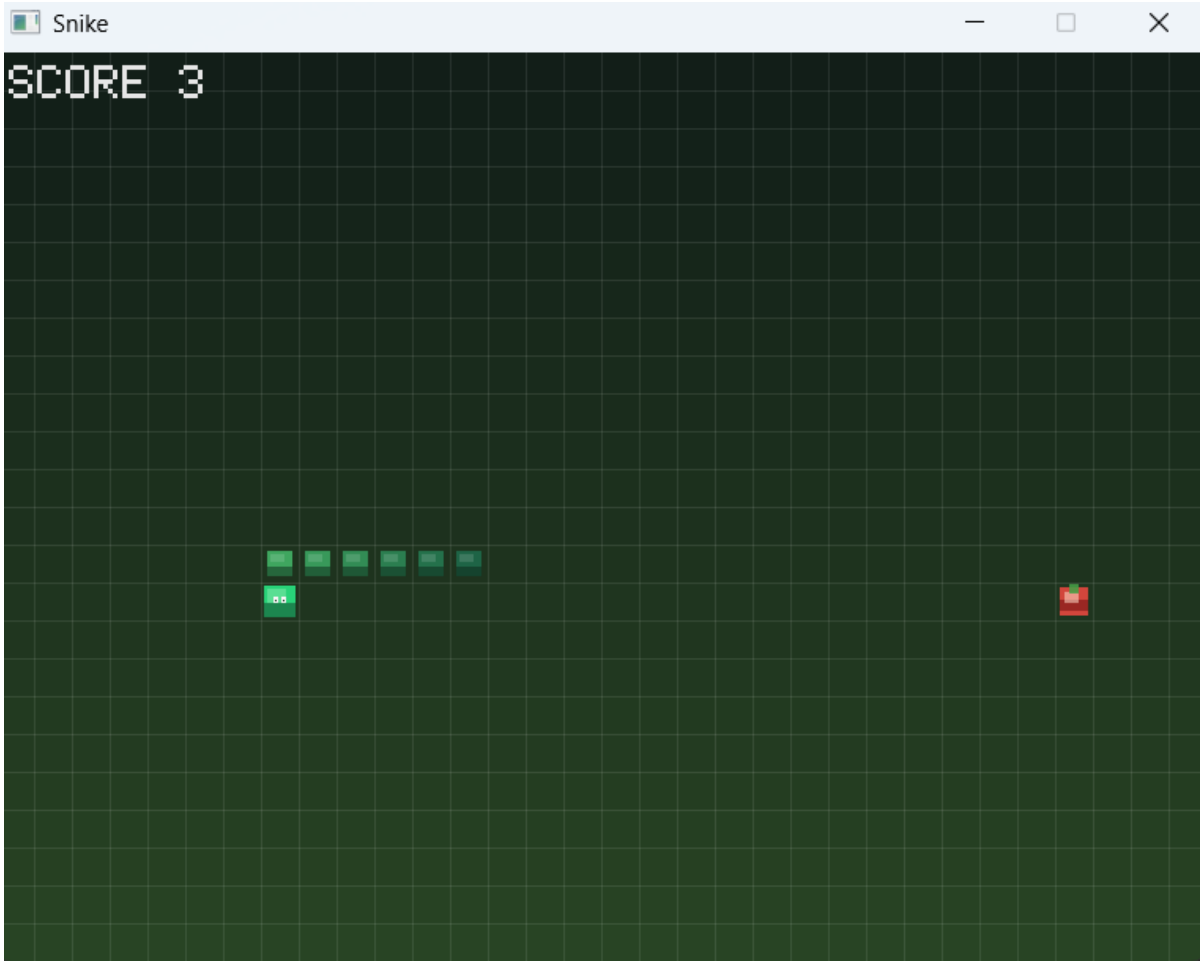
IV. Résultat final

Écran de menu

Au lancement, le joueur choisit son niveau de difficulté en appuyant sur N (Normal, 120 ms entre chaque mouvement) ou D (Difficile, 60 ms). L'écran titre affiche le nom du jeu : SNIKE.

Partie en cours

La grille de 32 × 24 cases est affichée avec un fond dégradé et un quadrillage transparent. Le score s'affiche en haut à gauche. Le serpent et le fruit sont rendus avec leurs effets visuels respectifs.



> *Touches : Flèches directionnelles pour piloter le serpent · S pour sauvegarder · L pour charger · Échap pour quitter*

Écran Game Over

Lorsque le serpent meurt (collision mur ou corps), un voile semi-transparent recouvre la grille et affiche « GAME OVER » avec le score final. Le joueur peut appuyer sur R pour revenir au menu.

V. Conclusion

La réalisation de ce jeu de Snake en C a été une première expérience concrète avec la programmation système et le rendu graphique bas niveau. Ce projet, bien que volontairement simple, a permis de manipuler des notions fondamentales du langage C dans un contexte ludique et visuel.

Bilan technique

Le développement de cette application a permis de mettre en pratique :

- La définition et l'utilisation de structures (struct, typedef, enum)
- La gestion de la mémoire et des tableaux de taille fixe
- L'écriture et la lecture de fichiers binaires avec fwrite / fread
- L'utilisation d'une bibliothèque externe (SDL3) pour la fenêtre et les événements
- Le dessin bas niveau par accumulation de rectangles et calculs de couleurs

Compétences acquises

- Analyser un besoin simple et le traduire en structures de données et en fonctions.
- Organiser un programme C en blocs logiques distincts.
- Déboguer des problèmes liés aux collisions et à la gestion des états.

Perspectives d'évolution

Plusieurs axes d'amélioration pourraient enrichir ce projet dans une prochaine version :

- Sauvegarde du meilleur score entre les sessions.
- Ajout d'effets sonores avec SDL_mixer.
- Mode de jeu supplémentaire avec obstacles fixes sur la grille.
- Affichage d'une animation lors de la mort du serpent.

> *En somme, ce projet constitue une base solide pour une première initiation au C : il illustre de manière concrète les mécanismes essentiels du langage tout en produisant un résultat jouable et visuellement soigné.*