

Formation (BTS SIO SLAM – IRIS)

Snake Project

Membre du groupe:

Titouen

Aïssa

Liamidi Saobane



Next Slide 



2025

Sommaire



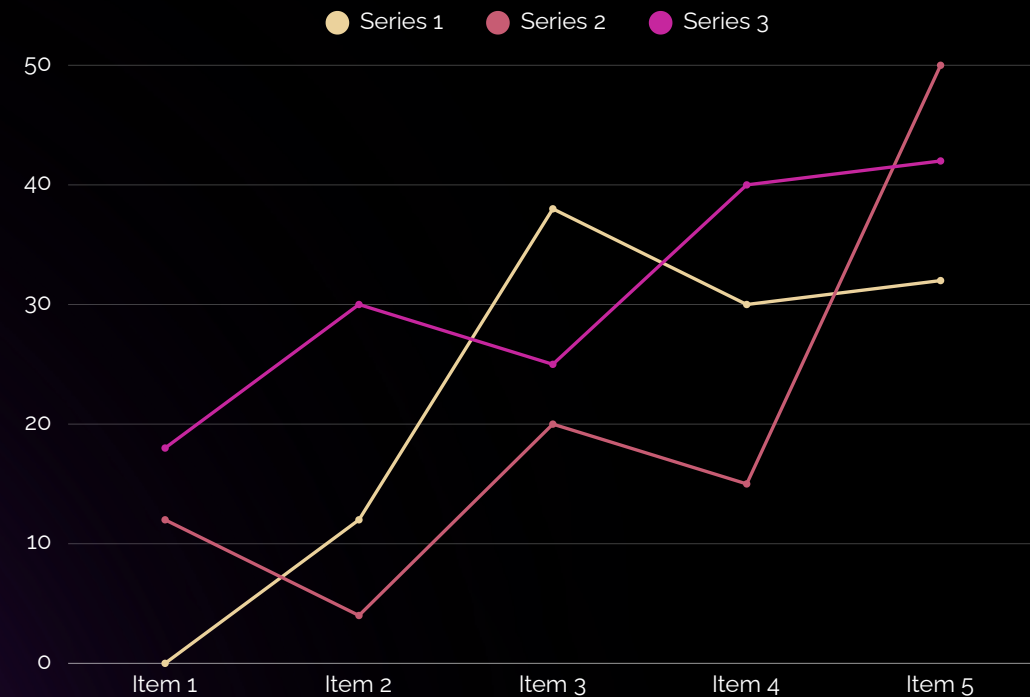
- Contexte & objectifs
- Présentation du groupe
- Cahier des charges du site
- Outils utilisés
- Organisation du travail
- Perspectives et conclusion





2025

Contexte & Objectif



Contexte et but du jeu

Le jeu Snake est un jeu vidéo classique apparu dans les années 1970, popularisé notamment sur les anciens téléphones Nokia.

Le principe est simple : le joueur contrôle un serpent qui se déplace dans un espace délimité. À chaque fois que le serpent mange une pomme (ou un autre élément), il grandit.

Le but du jeu est de manger un maximum de pommes sans se mordre la queue ni heurter les murs.

Dans le cadre de notre projet, le jeu a été revisité pour ajouter de nouvelles fonctionnalités :

- Génération de trois pommes à la fois ;
- Croissance du serpent uniquement après avoir mangé les trois ;
- Possibilité de sauvegarder et reprendre une partie ;
- Ajout de deux vitesses de jeu (normale et rapide).



Présentation du groupe



2025





Cahier des charges



2025

Rubrique	Contenu prévu
Terminal	l'endroit du déroulement du jeu





2025

Organisation du travail: Trello

A faire ...

Créer un jeu Snake

+ Add a card 

Terminer ...

chercher le code du jeu déjà fait sur le net
👁 SL AJ TR

comprendre le code
👁 SL AJ TR

modifier en fonction des taches données
👁 SL AJ TR





Planning



2025

- Chercher le code source d'un snake déjà fait
- Comprendre le code
- Trouver les parties à modifier
- Changer le code en fonction de la consigne demandé





Code source



2025

```
444         printf(" ");
445         len++;
446         if (len == length)
447             break;
448     }
449 }
450 }
451 }
452 void Boarder()
453 {
454     system("cls");
455     int i;
456
457     // Afficher les 3 pommes
458     for (i = 0; i < 3; i++)
459     {
460         if (food[i].x != 0 && food[i].y != 0) // Afficher seulement les pommes non mangées
461         {
462             GotoXY(food[i].x, food[i].y);
463             printf("P");
464         }
465     }
466
467     for (i = 10; i < 71; i++)
468     {
469         GotoXY(i, 10);
470         printf("|");
471         GotoXY(i, 30);
472         printf("|");
473     }
474     for (i = 10; i < 31; i++)
475     {
476         GotoXY(10, i);
477         printf("|");
478         GotoXY(70, i);
```

```
545 int Score()
546 {
547     int score;
548     GotoXY(20, 8);
549     score = length - 5;
550     printf("Score : %d", (length - 5));
551     score = length - 5;
552     GotoXY(50, 8);
553     printf("Vies : %d", life);
554     return score;
555 }
556 int Scoreonly()
557 {
558     int score = Score();
559     system("cls");
560     return score;
561 }
562 void Up()
563 {
564     int i;
565     for (i = 0; i <= (bend[bend_no].y - head.y) && len < length; i++)
566     {
567         GotoXY(head.x, head.y + i);
568         {
569             if (len == 0)
570                 printf("^");
571             else
572                 printf("");
573         }
574         body[len].x = head.x;
575         body[len].y = head.y + i;
576         len++;
577     }
578     Bend();
579     if (!khit())
580         head.y--;
581 }
```

```
407         for (j = 1; j <= diff; j++)
408         {
409             /*GotoXY(bend[i].x, bend[i].y);
410             printf("");*/
411             body[len].x = bend[i].x;
412             body[len].y = bend[i].y - j;
413             GotoXY(body[len].x, body[len].y);
414             printf("");
415             len++;
416             if (len == length)
417                 break;
418         }
419     }
420     else if (bend[i].y == bend[i - 1].y)
421     {
422         diff = bend[i].x - bend[i - 1].x;
423         if (diff < 0)
424             for (j = 1; j <= (-diff) && len < length; j++)
425             {
426                 /*GotoXY((bend[i].x - j), bend[i].y);
427                 printf("");*/
428                 body[len].x = bend[i].x - j;
429                 body[len].y = bend[i].y;
430                 GotoXY(body[len].x, body[len].y);
431                 printf("");
432                 len++;
433                 if (len == length)
434                     break;
435             }
436     }
437     else if (diff > 0)
438     {
439         for (j = 1; j <= diff && len < length; j++)
440         {
441             /*GotoXY((bend[i].x + j), bend[i].y);
442             printf("");*/
443             body[len].x = bend[i].x + j;
444             body[len].y = bend[i].y;
445             GotoXY(body[len].x, body[len].y);
```

```
256     for (i = 4; i < length; i++) //starts with 4 because it needs minimum 4 element to touch its own body
257     if (head.x <= 10 || head.x >= 70 || head.y <= 10 || head.y >= 30 || check != 0)
258     {
259         life--;
260         if (life >= 0)
261         {
262             head.x = 25;
263             head.y = 20;
264             bend_no = 0;
265             head.direction = RIGHT;
266             Move();
267         }
268     }
269     else
270     {
271         system("cls");
272         printf("Vous êtes mort !\nAppuyez sur une touche pour quitter\n");
273         record();
274         exit(0);
275     }
276 }
277 void food()
278 {
279     int i;
280     int allEaten = 1;
281
282     // Vérifier si toutes les pommes ont été mangées
283     for (i = 0; i < 3; i++)
284     {
285         if (food[i].x != 0 && food[i].y != 0) // Si une pomme existe encore
286         {
287             allEaten = 0;
288             break;
289         }
290     }
291
292     // Si toutes les pommes sont mangées, réinitialiser les 3 pommes et agrandir le serpent
293     if (allEaten == 1)
```

```
319     left();
320 }
321 else if (head.direction == DOWN)
322     Down();
323 }
324 else if (head.direction == UP)
325     Up();
326 }
327 while (!kbhit())
328     continue();
329 a = getch();
330 if (a == 'v')
331 {
332     system("cls");
333     exit(0);
334 }
335 key = getch();
336
337 if ((key == RIGHT && head.direction != LEFT && head.direction != DOWN) || (key == LEFT && head.direction != RIGHT))
338 {
339     bend_no++;
340     bend[bend_no] = head;
341     head.direction = key;
```

```
302 {
303     length++;
304     foodsEaten = 0;
305     for (i = 0; i < 3; i++)
306     {
307         food[i].x = rand() % 70;
308         if (food[i].x <= 10)
309             food[i].x += 11;
310         food[i].y = rand() % 30;
311         if (food[i].y <= 10)
312             food[i].y += 11;
313     }
314 }
315
316 // Vérifier les collisions avec chaque pomme
317 for (i = 0; i < 3; i++)
318 {
319     if (head.x == food[i].x && head.y == food[i].y)
320     {
321         food[i].x = 0; // Marquer la pomme comme mangée
322         food[i].y = 0;
323         foodsEaten++;
324         break;
325     }
326 }
327
328 // Opération initiale des pommes (seulement au début du jeu)
329 static int firstTime = 1;
330 if (firstTime)
331 {
332     firstTime = 0;
333     for (i = 0; i < 3; i++)
334     {
335         food[i].x = rand() % 70;
336         if (food[i].x <= 10)
337             food[i].x += 11;
338         food[i].y = rand() % 30;
```

```
285 void Food()
286 {
287     if (firstTime)
288     {
289         for (i = 0; i < 3; i++)
290         {
291             food[i].y = rand() % 30;
292             if (food[i].y <= 10)
293                 food[i].y += 11;
294         }
295     }
296 }
297 void Left()
298 {
299     int i;
300     for (i = 0; i <= (bend[bend_no].x - head.x) && len < length; i++)
301     {
302         if (len == 0)
303             printf("<");
304         else
305             printf("");
306     }
307     body[len].x = head.x + i;
308     body[len].y = head.y;
309     len++;
310 }
311 Bend();
312 if (!khit())
313     head.x--;
314 }
315 void Right()
316 {
317     int i;
318     for (i = 0; i <= (head.x - bend[bend_no].x) && len < length; i++)
319     {
320         //GotoXY((head.x - i), head.y);
321         body[len].x = head.x - i;
322         body[len].y = head.y;
323         GotoXY(body[len].x, body[len].y);
```

```
226 void Down()
227 {
228     int i;
229     for (i = 0; i <= (head.y - bend[bend_no].y) && len < length; i++)
230     {
231         GotoXY(head.x, head.y - i);
232         {
233             if (len == 0)
234                 printf("v");
235             else
236                 printf("");
237         }
238         body[len].x = head.x;
239         body[len].y = head.y - i;
240         len++;
241     }
242     Bend();
243     if (!khit())
244         head.y++;
245 }
246 void Delay(long double k)
247 {
248     Score();
249     long double i;
250     for (i = 0; i <= (100000000); i++)
251     {
252     }
253 }
254 void ExitGame()
255 {
256     int i, check = 0;
257     for (i = 4; i < length; i++) //starts with 4 because it needs minimum 4 element to touch its own body
258     {
259         if (body[0].x == body[i].x && body[0].y == body[i].y)
260         {
261             check++; //check's value increases as the coordinates of head is equal to any other body coord
262         }
263         if (i == length || check != 0)
264             break;
265     }
266 }
```





2025

Bilan personnel



Ce projet m'a permis de renforcer ma compréhension de la logique algorithmique et de la programmation en C. J'ai appris à structurer un programme complet, à gérer les déplacements du serpent et la détection des collisions. J'ai également découvert l'importance de la rigueur dans la gestion de la mémoire et du débogage. Ce travail m'a aidé à progresser dans la conception de programmes interactifs et à mieux comprendre le fonctionnement des boucles et conditions imbriquées.



Grâce à ce projet, j'ai développé mes compétences en travail d'équipe et en organisation du code. J'ai participé à la mise en place des différentes fonctionnalités du jeu, notamment la génération des pommes et la gestion du score. J'ai appris à lire et modifier un code existant pour l'adapter aux nouvelles consignes, ce qui m'a permis de gagner en autonomie et en efficacité dans mes approches de résolution de problèmes.



Ce projet m'a permis de mieux comprendre le fonctionnement d'un jeu vidéo simple en langage C. J'ai travaillé sur la partie affichage et sur l'adaptation du programme aux nouvelles fonctionnalités, comme la sauvegarde et la reprise d'une partie. J'ai également découvert l'intérêt de bibliothèques externes comme conio.h ou SDL pour gérer l'affichage et les interactions. Cette expérience m'a montré l'importance du travail collaboratif et du partage de connaissances dans la réussite d'un projet.





2025

Conclusion

Ce projet Snake nous a permis d'appliquer nos connaissances en langage C et en algorithmique. Nous avons appris à analyser et modifier un programme existant, à ajouter de nouvelles fonctionnalités et à collaborer efficacement en équipe. C'était une expérience enrichissante sur les plans technique et pratique.





2025

Thank You!