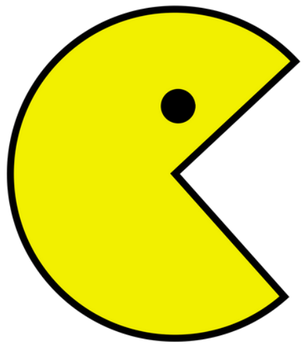
# Jeu Pacman




**B2**

**Floriane Guillou** -> grp 2
**Marion Lafosse** -> grp 1

# Sommaire

1. Description du projet
2. Organisation du projet
3. Difficultés rencontrées
4. Résultats

# 1. Description du projet

Le projet consiste à développer une version simplifiée du jeu Pacman en utilisant le langage C et la bibliothèque graphique Allegro. L'objectif est de simuler le déplacement d'un personnage (Pacman) dans un labyrinthe, tout en collectant des points et en évitant des fantômes.

Fonctionnalités Implémentées :

Déplacement de Pacman dans un labyrinthe. ✅
Collecte de points disséminés dans le labyrinthe. ✅
Déplacement automatique des fantômes en fonction de la position de Pacman. ✅
Gestion des collisions entre Pacman et les fantômes. ✅
Affichage des vies restantes. ✅
Fin de jeu avec message "Game Over" lorsque toutes les vies sont perdues. ✅
Suite de jeu avec message "Félicitation vous avez réussi" lorsque tous les cubes ont été mangés. ✅
Passage à un niveau suivant. ✅

# 2. Organisation du projet

En raison de contraintes matérielles et organisationnelles, nous étions seulement deux pour ces projets :

- Marion Lafosse : qui a travaillée principalement sur le projet C++, contribuant également aux aspects conceptuels et fonctionnels du projet Allegro
- Moi-même : qui a travaillée principalement sur ce projet, car mon ordinateur était le seul à pouvoir exécuter cette bibliothèque. J'ai également contribué aux aspects conceptuels et fonctionnels du projet C++

La gestion du jeu :

- game.c : Contient la logique principale du jeu, comme la boucle principale (game_loop) et les fonctions (par ex: la de gestion des points et des collisions).
- menu.c : Gère l'affichage de l'entrée de jeu.
- ghost.c : Gère le déplacement et le comportement des fantômes.
- graphics.c : Contient les fonctions pour dessiner Pacman, les fantômes, le labyrinthe, et les vies.
- main.c : Point d'entrée du programme, initialisation d'Allegro, et lancement du jeu.
- Headers (*.h) : Définissent les structures et les prototypes de fonctions.

# 3. Difficultés rencontrées

Tout au long de ce projet nous avons rencontrés de nombreux problèmes, tout d'abord l'installation d'Allegro.

Ensuite les **principaux problèmes** qu'on à eu c'est le son , l'affichage du jeu, le déplacement automatique des fantômes ( +en fonction du Pacman ) , l'affichage du labyrinthe, la gestion des collisions entre Pacman et les fantômes.
Mais nous avons réussi au fur et mesure améliorer notre code pour cela puisse fonctionner.

Je peux vous donner quelques exemples:

- Pour le déplacement automatique des fantômes nous avons ajouté une logique pour la collision avec les murs. Puis nous avons ajoutés la fonction **chase_pacman** pour la poursuite de Pacman. Aussi j'avais un problème pour les faire bouger à un intervalle régulier du coup j'ai incrémenté un compteur dans la boucle principale : **if (cycle % GHOST_SPEED == 0)**

```
Fant├┤me 0 : position (30, 120)
Fant├┤me 1 : position (30, 150)
Fant├┤me 2 : position (30, 120)
Fant├┤me 3 : position (30, 150)
```

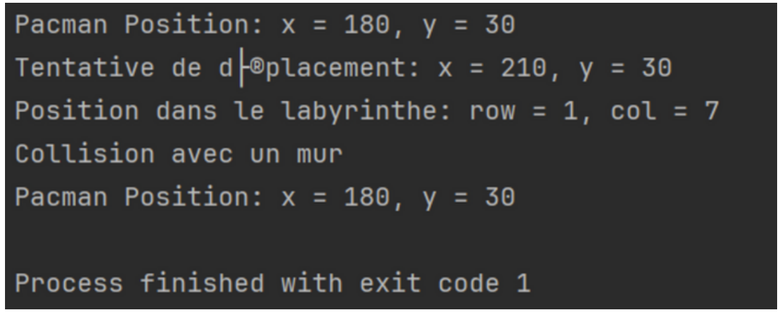
```
void chase_pacman(Ghost *ghost, Pacman *pacman) {
    int dx = pacman->x - ghost->x;
    int dy = pacman->y - ghost->y;

    // Prioriser les mouvements horizontaux ou verticaux
    if (abs(dx) > abs(dy)) {
        if (dx > 0 && labyrinth[ghost->y / TILE_SIZE][(ghost->x + TILE_SIZE) / TILE_SIZE] != 1) {
            ghost->direction = 1; // Droite
        } else if (labyrinth[ghost->y / TILE_SIZE][(ghost->x - TILE_SIZE) / TILE_SIZE] != 1) {
            ghost->direction = 3; // Gauche
        } else if (dy > 0) {
            ghost->direction = 2; // Bas
        } else {
            ghost->direction = 0; // Haut
        }
    } else {
        if (dy > 0 && labyrinth[(ghost->y + TILE_SIZE) / TILE_SIZE][ghost->x / TILE_SIZE] != 1) {
            ghost->direction = 2; // Bas
        } else if (labyrinth[(ghost->y - TILE_SIZE) / TILE_SIZE][ghost->x / TILE_SIZE] != 1) {
            ghost->direction = 0; // Haut
        } else if (dx > 0) {
            ghost->direction = 1; // Droite
        } else {
            ghost->direction = 3; // Gauche
        }
```

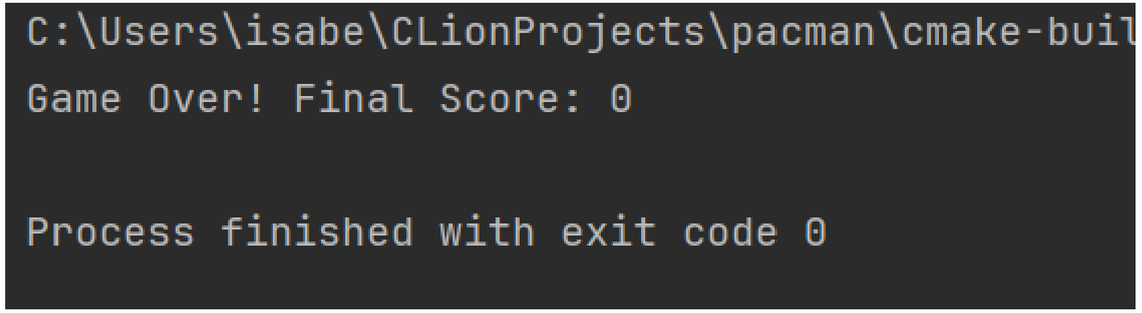
- Les collisions entre Pacman et les fantômes fonctionnaient pas donc en d'autres termes le jeu fonctionnait pas entièrement. Au final après pleins de recherche on a fini par bien définir la fonction **check_ghost_collision** ce qui a permis de faire fonctionner le jeu.

```
void check_ghost_collision(Pacman *pacman, Ghost *ghosts, bool *running) {
    for (int i = 0; i < 4; i++) {
        if (abs(pacman->x - ghosts[i].x) < TILE_SIZE &&
            abs(pacman->y - ghosts[i].y) < TILE_SIZE) {
            lives--;  // Réduire les vies
            printf("Collision! Lives remaining: %d\n", lives);
            pacman->x = TILE_SIZE;  // Réinitialiser Pacman
            pacman->y = TILE_SIZE;

            if (lives <= 0) {
                game_over();
            }
            rest(1000);
        }
    }
}
```



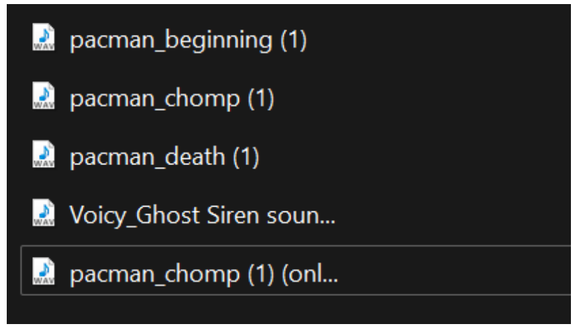

On peut voir ici qu'au tout début Pacman se prenait le mur et n'avançait pas




On peut voir ici que le jeu ne fonctionnait pas car dès qu'on entrait ça ne fonctionnait pas

Ici voilà le problème que l'on a pas réussi à résoudre malgré tous nos moyens :
Et j'aimerai comprendre d'où viens le soucis ... ( je pense que c'est surement des fichiers audio car tout le reste semble fonctionnelle, il doit y avoir des règles spécifiques pour les audios que je n'ai pas trouvés)

```
// Charger les fichiers audio
collision_sound = load_sample( filename: "audio/pacman_chomp(1)(online-audio-converter.com).wav");
game_over_sound = load_sample( filename: "audio/pacman_death(1).wav");
background_music = load_sample( filename: "audio/pacman_beginning(1).wav");
```




On voit bien que les fichiers ont le bon chemin car il ont été mis dans un dossier audio

```
SAMPLE *collision_sound;   // Son pour les collisions
SAMPLE *game_over_sound;   // Son pour le Game Over
SAMPLE *background_music;  // Musique de fond
```

Ici je vous mets le code que j'avais mis dans game.c

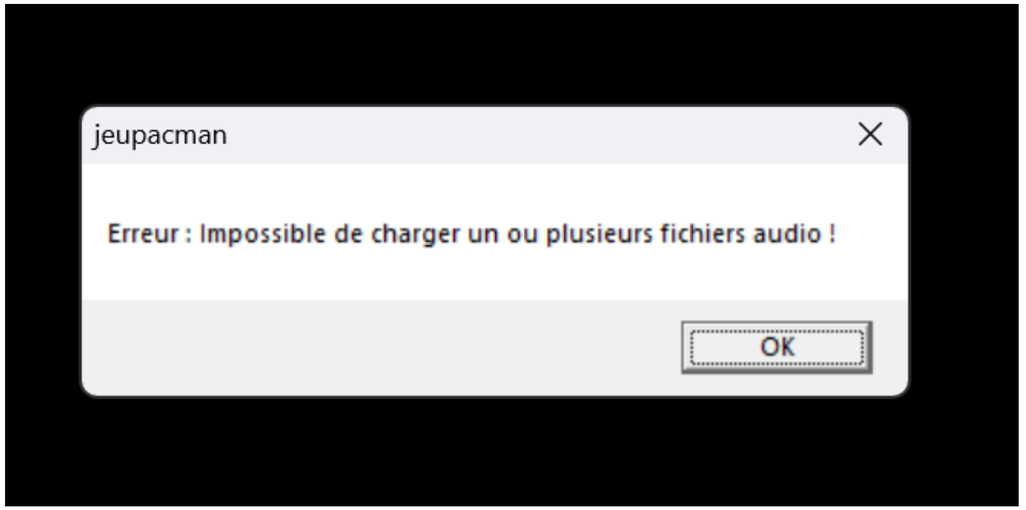
```
}
if (collision_sound) destroy_sample( spl: collision_sound);
if (game_over_sound) destroy_sample( spl: game_over_sound);
if (background_music) destroy_sample( spl: background_music);
```

```
play_sample( spl: collision_sound, vol: 255, pan: 128, freq: 1000, loop: 0);
```

exemple de ce que jai pu mettre dans check_ghost_collision

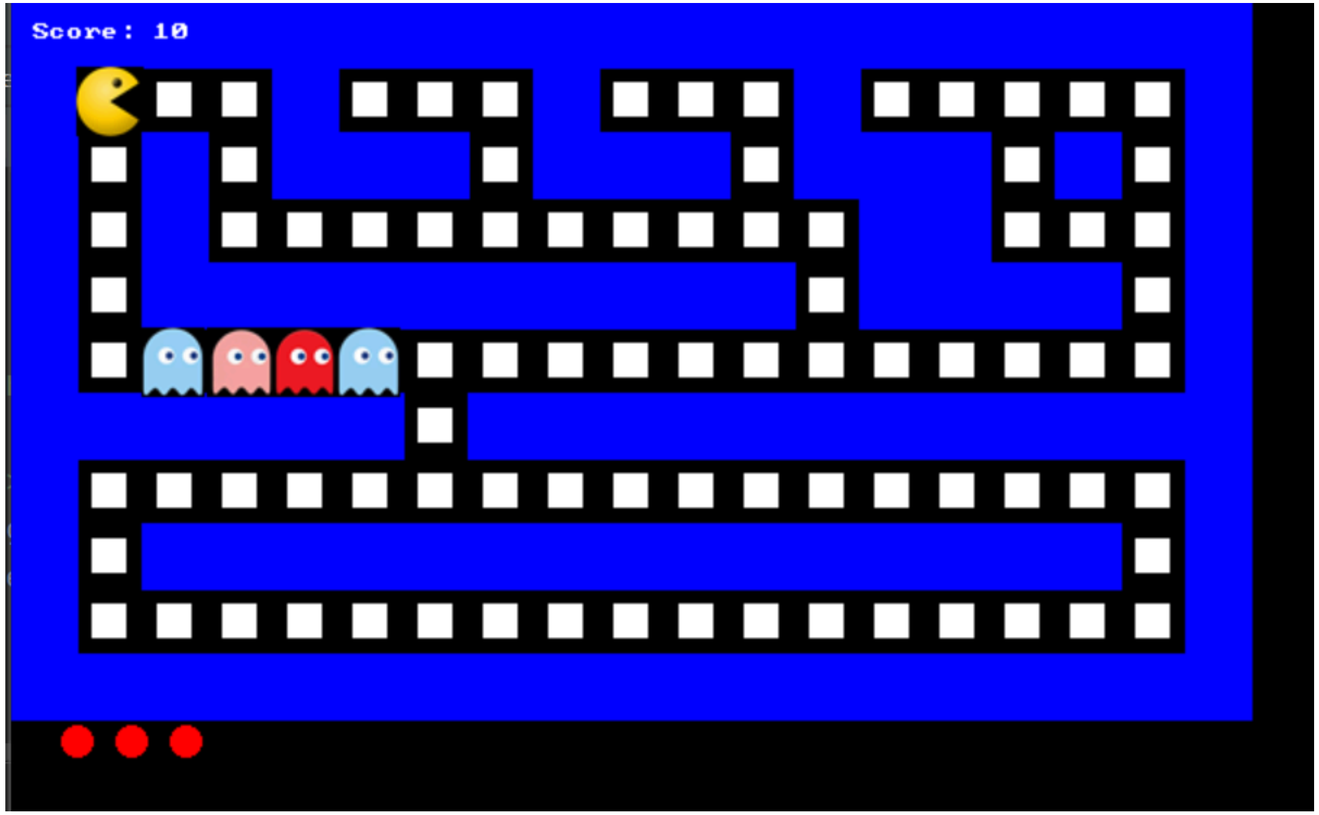
```
extern SAMPLE *collision_sound;
extern SAMPLE *game_over_sound;
extern SAMPLE *background_music;
```

game.h



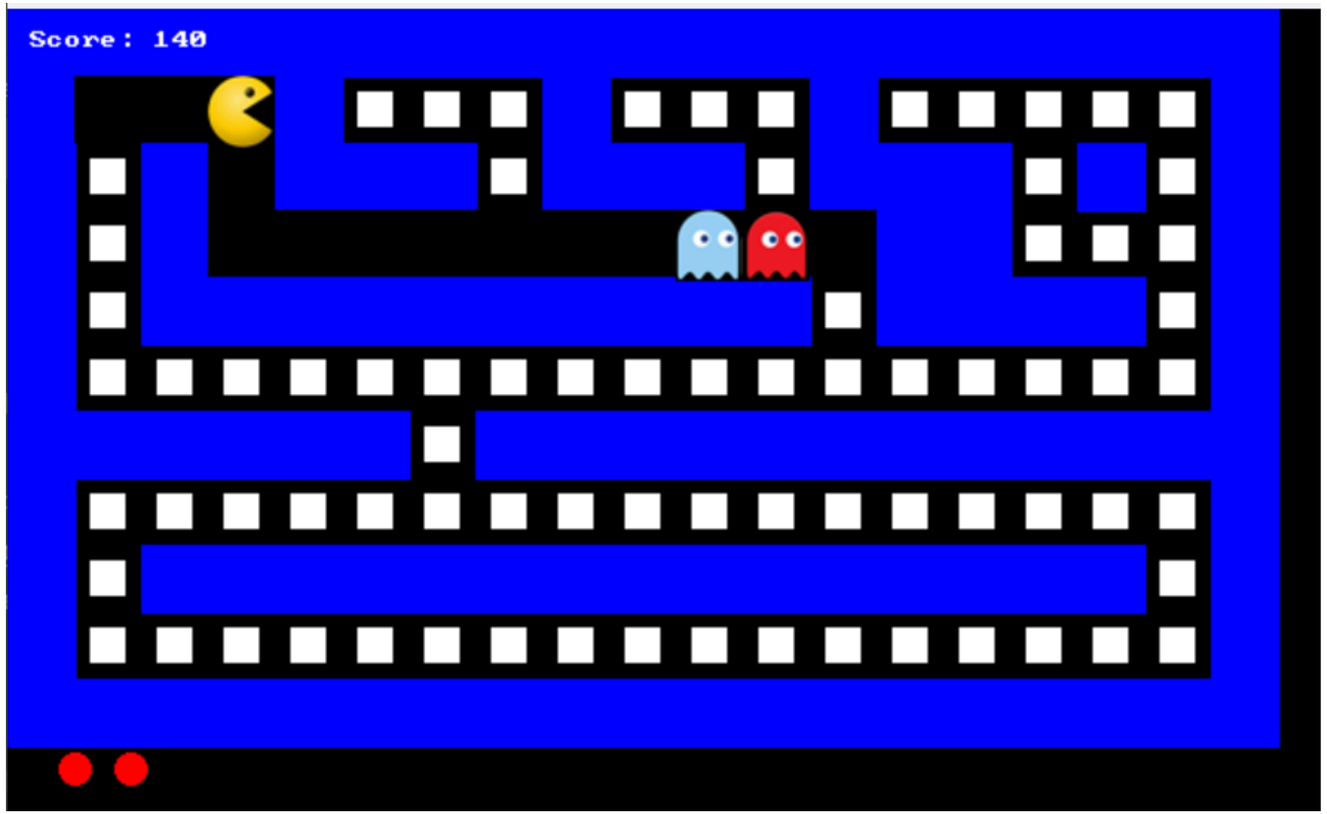

Résultat.....

# 4. Résultats

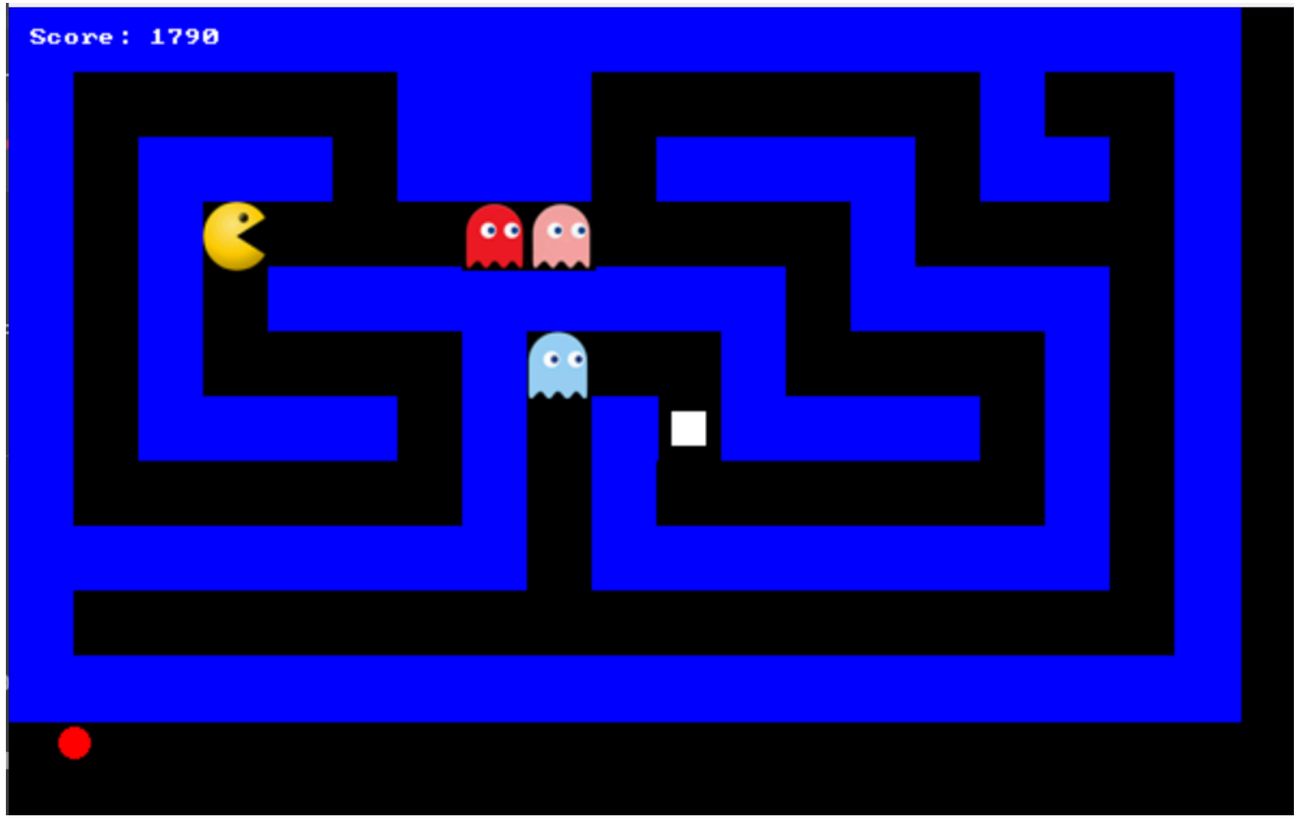



On peut voir le labyrinthe qui s’affiche ainsi que Pacman et les fantômes.
On aperçoit le score en haut et les vies en bas représentées par des ronds rouges.

J’ai décidé de faire un labyrinthe plus petit parce que je préférais le rendu en petite taille plutôt qu’en taille qui prenne toute la page




On peut voir ici que Pacman à commencé son chemin. Le score augmente de +10 dès qu’il mange un cube. Ensuite on peut également apercevoir qu’un fantôme la touché car on voit en bas une vie qui lui à été enlevée.




On peut voir ici que Pacman à réussi le premier niveau à savoir on a un message où il a été écrit “Félicitation un niveau à été réussi” et ensuite on passe au niveau 2 qui est celui-ci. Ensuite il y a un 3ème niveau.

A la fin du 3ème niveau on a un message de réussite complète du jeu.

```
color: makecol( r: 0,  g: 0,  b: 0)); // Nettoyer l'écran
 f: font,  str: "Niveau Terminé !",  x: SCREEN_W / 2,  y: SCREEN_H /
 f: font,  str: "Appuyez sur Entrée pour continuer.",  x: SCREEN_W
( r: 255,  g: 255,  b: 255),  bg: -1);

joueur appuie sur une touche
```

Ca c’est quand un niveau est terminé

```
void game_win() {
    clear_to_color( bitmap: screen,  color: makecol( r: 0,  g: 0,  b: 0)); // Nettoyer l'écran
    textout_centre_ex( bmp: screen,  f: font,  str: "Vous avez gagné !",  x: SCREEN_W / 2,  y: SCREEN_H
    textout_centre_ex( bmp: screen,  f: font,  str: "Félicitations !",  x: SCREEN_W / 2,  y: SCREEN_H /
    textout_centre_ex( bmp: screen,  f: font,  str: "Appuyez sur une touche pour quitter.",  x: SCREEN_
    readkey(); // Attendre une touche
    exit(0); // Quitter le jeu
}
```

Ca c’est quand le jeu est totalement terminé