



PRESENTATION

PROJET JEU SNAKE C

Yasmine afellat
Cassy koa



SOMMAIRE



- **Installation de l'environnement C# sur Windows**
 - Installation de **WinGW**
 - Sélection des composants à installer
 - Vérification de l'installation
 - Rajout de MinGW au Path
- **Compilation du code C#**
 - Préparation
 - Compilation du code
 - Exécution du programme
- **Génération des pommes**
 - Rajout de 3 pommes
 - Affichage des pommes
 - Augmentation de la taille du serpent
- **Modification demandé**
 - Arrêt du jeu quand le serpent atteint les parois
 - Sauvgarde de la partie
 - Affichage de la partie Sauvgarder
 - Changement de vitesse

Installation de l'environnement C# sur Windows

- Installation de WinGW

Téléchargement de MinGW sur le site officiel : [Installation de WinGW](#)

Téléchargez le gestionnaire d'installation "**mingw-get-setup.exe**".

- Sélection des composants à installer

Package	Class	Installed Version	Repository Version	Description
<input type="checkbox"/> mingw-developer-toolkit	bin		2013072300	An MSYS Installation for MinGW Developers (meta)
<input checked="" type="checkbox"/> mingw32-base	bin	2013072200	2013072200	A Basic MinGW Installation
<input type="checkbox"/> mingw32-gcc-ada	bin		6.3.0-1	The GNU Ada Compiler
<input type="checkbox"/> mingw32-gcc-fortran	bin		6.3.0-1	The GNU FORTRAN Compiler
<input checked="" type="checkbox"/> mingw32-gcc-g++	bin	6.3.0-1	6.3.0-1	The GNU C++ Compiler
<input checked="" type="checkbox"/> mingw32-gcc-objc	bin	6.3.0-1	6.3.0-1	The GNU Objective-C Compiler
<input checked="" type="checkbox"/> msys-base	bin	2013072300	2013072300	A Basic MSYS Installation (meta)

- mingw32-base : Il s'agit du compilateur GCC (GNU Compiler Collection) pour C
- mingw32-gcc-g++ : Il s'agit du compilateur GCC pour C++.
- msys-base : Il fournit un environnement en ligne de commande amélioré qui est utile pour les opérations système.

- **Rajout de MinGW au Path**

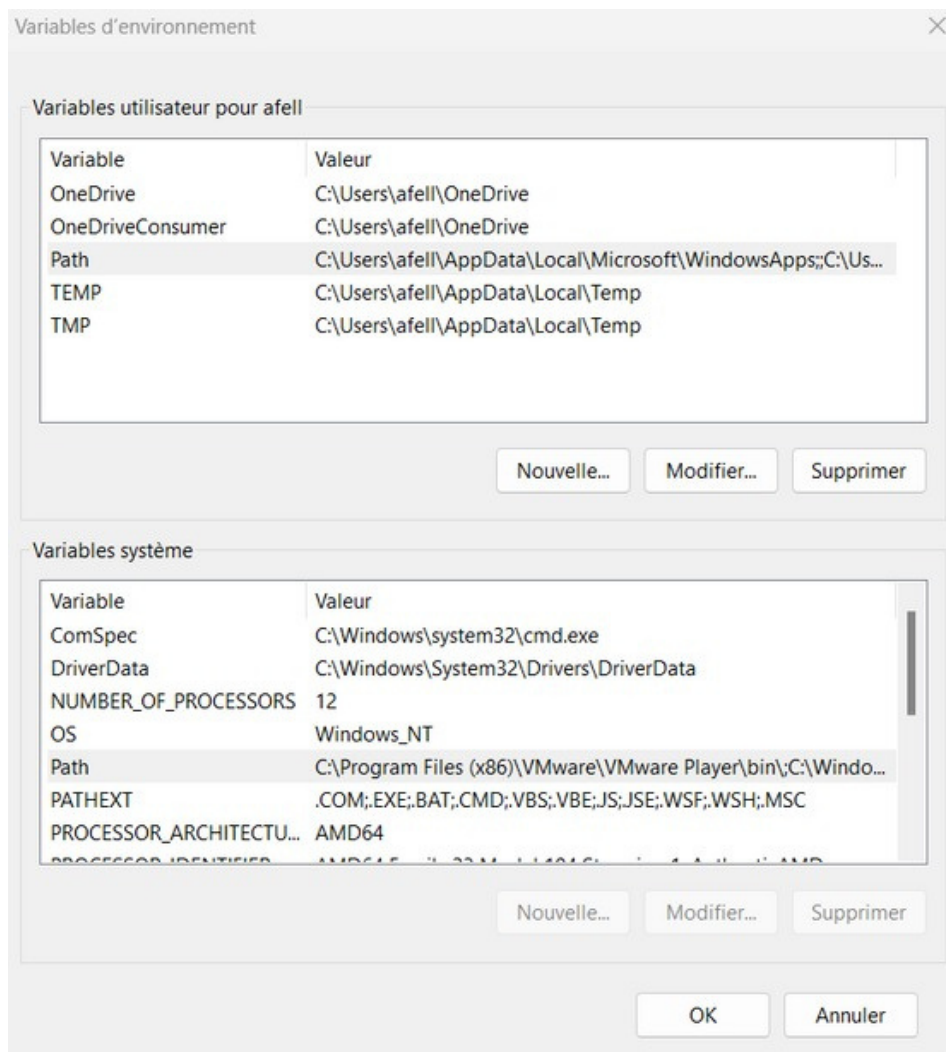
Une fois l'installation terminée, on doit ajouter le chemin vers le répertoire bin de **MinGW** à votre variable d'environnement **PATH**. Cela permettra d'exécuter les compilateurs et autres outils MinGW depuis n'importe quel répertoire dans l'invite de commandes.

Cliquez avec le bouton droit sur "**Ce PC**" ou "Poste de travail" et sélectionnez "**Propriétés**".

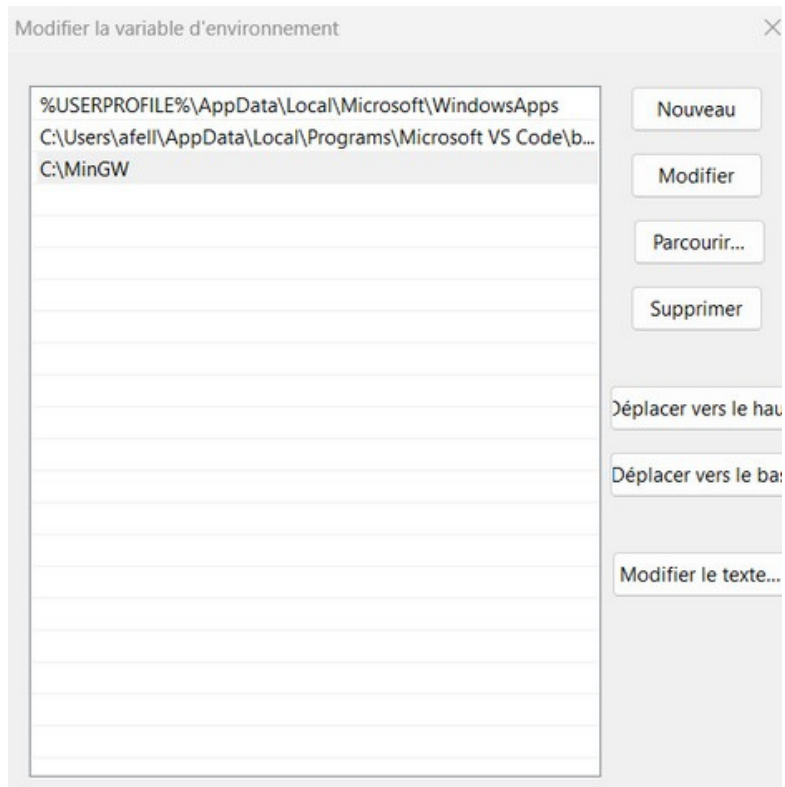
Cliquez sur "**Paramètres système avancés**" (sur le côté gauche).

Cliquez sur "**Variables d'environnement**".

Dans la section "**Variables système**", recherchez et sélectionnez "**Path**", puis cliquez sur "**Modifier**".



- Cliquez sur "**Nouveau**" et ajoutez le chemin **C:\MinGW\bin**



• Vérification de l'installation

```
gcc --version
```

On doit avoir ce résultat : Téléchargement réussi

```
gcc (MinGW.org GCC Build-2) 9.2.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

```
Microsoft Windows [version 10.0.22621.2283]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\afell>echo %PATH%
C:\Program Files (x86)\VMware\VMware Player\bin\;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Windows\System32\OpenSSH\;C:\Program Files\Git\cmd;C:\Users\afell\AppData\Local\Microsoft\WindowsApps;C:\Users\afell\AppData\Local\Programs\Microsoft VS Code\bin;C:\MinGW\bin;

C:\Users\afell> gcc --version
gcc (MinGW.org GCC-6.3.0-1) 6.3.0
Copyright (C) 2016 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

C:\Users\afell>
```

Compilation du code C#

Placez le fichier source C

Mettez votre fichier source C (par exemple, mon_code.c) dans un dossier de votre choix.

Accédez au dossier

Utilisez la commande `cd` pour accéder au dossier contenant votre code. Par exemple, si votre code est dans `C:\chemin\vers\le\dossier`, tapez :

```
cd C:\chemin\vers\le\dossier
```

Compilation du code

Utilisez la commande `gcc` pour compiler votre code C. Voici la commande avec des explications :

```
gcc mon_code.c -o mon_executable
```



gcc : C'est le compilateur que nous utilisons.

mon_code.c : C'est le nom de votre fichier source.

-o mon_executable : Cela indique à GCC de créer un fichier exécutable avec le nom `mon_executable`.

Exécution du programme

```
mon_executable.exe
```

Nom	Modifié le	Type	Taille
 snake	06/10/2023 17:37	Fichier source C	4 Ko
 snaky	06/10/2023 17:38	Application	44 Ko

Après exécution du code on obtient ce résultat

```
#####  
#                                     #  
#                                     #  
#                                     #  
#                                     #  
#                                     #  
#                                     #  
#                                     #  
#                                     #  
# F                                   #  
#                                     #  
#                                     #  
#          0                         #  
#                                     #  
#                                     #  
#                                     #  
#                                     #  
#                                     #  
#                                     #  
#                                     #  
#                                     #  
#                                     #  
#####  
Score:0  
C:\Users\afell\Desktop\sna>
```

Génération des pommes

```
void Setup()
{
    gameover = 0;
    dir = STOP;
    x = width / 2;
    y = height / 2;
    fruitx = rand() % width;
    fruity = rand() % height;
    score = 0;
}
```

Dans ce code, la génération de la pomme se passe dans la fonction void Setup().

- **rand()** est une fonction en C qui génère un nombre aléatoire.
- **%** est l'opérateur modulo, il retourne le reste de la division de deux nombres. Par exemple, 5 % 3 donne 2, car 3 rentre deux fois dans 5 avec un reste de 2.
- **rand() % width** génère un nombre aléatoire entre 0 et width - 1. Cela signifie que fruitx (la coordonnée x de la pomme) sera comprise entre 0 et width - 1.
- De manière similaire, **rand() % height** génère un nombre aléatoire entre 0 et height - 1, donc fruity (la coordonnée y de la pomme) sera comprise entre 0 et height - 1.

- **Rajout de 3 pommes**

- Ajoutez trois nouvelles variables globales pour stocker les coordonnées x et y des trois pommes :

```
// Variables globales
int gameOver;
int x, y, pomme1X, pomme1Y, pomme2X, pomme2Y, pomme3X, pomme3Y score;
int tailX[100], tailY[100];
int nTail;
enum eDirection { STOP = 0, LEFT, RIGHT, UP, DOWN};
enum eDirection dir;
```

- Modifiez la fonction Setup() pour initialiser les positions des trois pommes de manière aléatoire :

```
void Setup()
{
    // Initialisation du jeu
    gameOver = 0;
    dir = STOP;
    x = LARGEUR / 2;
    y = HAUTEUR / 2;
    fruitX = rand() % LARGEUR;
    fruitY = rand() % HAUTEUR;
    score = 0;
}
```

Après modification

```
do {
    pomme1X = rand() % width;
    pomme1Y = rand() % height;
} while ((pomme1X == x && pomme1Y == y));

do {
    pomme2X = rand() % width;
    pomme2Y = rand() % height;
} while ((pomme2X == x && pomme2Y == y) || (pomme2X == pomme1X && pomme2Y == pomme1Y));

do {
    pomme3X = rand() % width;
    pomme3Y = rand() % height;
} while ((pomme3X == x && pomme3Y == y) || (pomme3X == pomme2X && pomme3Y == pomme2Y) || (pomme3X == pomme1X && pomme3Y == pomme1Y));
score = 0;
```

```

do {
pomme1X = rand() % width;
pomme1Y = rand() % height;

} while ((pomme1X == x && pomme1Y == y) );

do {
pomme2X = rand() % width;
pomme2Y = rand() % height;

} while ((pomme2X == x && pomme2Y == y) || (pomme2X == pomme1X
& pomme2Y == pomme1Y));

do {
pomme3X = rand() % width;
pomme3Y = rand() % height;

} while ((pomme3X == x && pomme3Y == y) || (pomme3X == pomme2X &&
pomme3Y == pomme2Y)|| (pomme3X == pomme1X && pomme3Y ==
pomme1Y) );

score = 0;
}

```

- Affichage des pommes

Pour cela il faudra modifier la fonction **Draw()** pour afficher les trois pommes

```

// Affichage de la premiere pomme
if (i == apple1y && j == apple1x)
    printf("*");
// Affichage de la deuxieme pomme
else if (i == apple2y && j == apple2x)
    printf("*");
// Affichage de la troisieme pomme
else if (i == apple3y && j == apple3x)
    printf("*");

```

- **Augmentation de la taille du serpent**

```
if (count == 3)
{
    score += 10;
    nTail++;
}
```

Pour cela on a mis en place une variable count qui est initialisée à 0 et elle s'incrémente à chaque fois que le serpent mange une pomme quand celle-ci atteint 3 qui est le nombre total des pommes à manger **nTail++**; ce qui veut dire que la taille du serpent augmente d'un cran

- **Arrêt du jeu quand le serpent atteint les parois**

La partie du code responsable du comportement du serpent quand il touche les parois est la suivante :

```
if(x >= LARGEUR) x = 0; else if(x < 0) x = LARGEUR - 1;
if(y >= HAUTEUR) y = 0; else if(y < 0) y = HAUTEUR - 1;
```

dans ce cas on a :

Quand le serpent touche les parois il réapparaît de l'autre côté

Du coup pour réaliser la condition demandée dans le tp il faudra rajouter **gameOver = 1**; ce qui mettra fin à la partie

```
if(x >= LARGEUR) gameOver = 1; else if(x < 0) gameOver = 1;
if(y >= HAUTEUR) gameOver = 1; else if(y < 0) gameOver = 1;
```

• Sauvegarde de la partie

```
void Sauvegarde(char* nomFichier) {  
    FILE *fichier = fopen(nomFichier, "w");  
    if (fichier != NULL) {  
        // Écrivez les données de sauvegarde ici  
        fprintf(fichier, "%d %d %d %d %d %d %d %d %d %d %d %d",  
            gameOver, x, y, pomme1X, pomme1Y, pomme2X, pomme2Y, pomme3X, pomme3Y, score,  
            for(int i = 0; i < nTail; i++) {  
                fprintf(fichier, " %d %d", tailX[i], tailY[i]);  
            }  
        fclose(fichier);  
    }  
}
```

```
case 'c':  
    Sauvgarde(); // Appel de la fonction de sauvegarde  
    gameOver = 1;  
    break;  
}
```

Les modifications que on a rajouter au code sont :

- l’affichage d’une phrase qui informe que on peut sauvegarder la partie en cliquant sur la touche c du clavier
- le rajout du case ‘c’ qui détecte le click sur la touche c et exécute la fonction qui sauvegarde la partie

• Recharger la partie

```
void Sauvegarde(char* nomFichier) {
    FILE *fichier = fopen(nomFichier, "w");
    if (fichier != NULL) {
        // Écrivez les données de sauvegarde ici
        fprintf(fichier, "%d %d %d %d %d %d %d %d %d %d %d %d",
            gameOver, x, y, pomme1X, pomme1Y, pomme2X, pomme2Y, pomme3X, pomme3Y, score,
            nTail, nTail);

        for(int i = 0; i < nTail; i++) {
            fprintf(fichier, " %d %d", tailX[i], tailY[i]);
        }

        fclose(fichier);
    }
}
```

- **void Sauvegarde(char* nomFichier):** C'est la déclaration de la fonction Sauvegarde. Cette fonction prend un argument nomFichier, qui est un pointeur vers une chaîne de caractères (une chaîne de caractères est utilisée pour stocker le nom du fichier dans lequel vous souhaitez sauvegarder l'état du jeu).
- **FILE* fichier = fopen(nomFichier, "wb");** Cette ligne ouvre le fichier
- **fprintf(fichier, "%d %d %d %d %d %d %d %d %d %d %d %d", ...);** Cette ligne utilise fprintf pour écrire les variables du jeu dans le fichier. Chaque %d dans la chaîne de format représente un emplacement où une variable entière doit être insérée. Les variables sont énumérées après la chaîne de format, dans l'ordre correspondant. Par exemple, %d correspond à une variable de type int.
- **fclose(fichier);** Cette ligne ferme le fichier après avoir écrit les données. Cela libère les ressources associées au fichier.

```
void Charger(char* nomFichier) {
    FILE *fichier = fopen(nomFichier, "r");
    if (fichier != NULL) {
        fscanf(fichier, "%d %d %d %d %d %d %d %d %d %d %d %d",
            &gameOver, &x, &y, &pomme1X, &pomme1Y, &pomme2X, &pomme2Y, &pomme3X, &pomme3Y,
            &nTail, &nTail);

        for(int i = 0; i < nTail; i++) {
            fscanf(fichier, " %d %d", &tailX[i], &tailY[i]);
        }

        fclose(fichier);
    }
}
```

- **Vitesse du jeu**

```
int vitesse = 150;  
int vitesse2 = 10;
```

- On initialise la vitesse par défaut à 150 ms et on crée une constante vitesse2 à 10 ms
- On met à jour la variable vitesse directement dans la fonction **Input()**

```
case 'r':  
    vitesse = vitesse2;  
    break;  
}
```

Dans la boucle principale, on utilise la variable **vitesse** pour déterminer le temps

```
printf("Game Over!\n");  
while (1) {  
    if (_kbhit()) {  
        switch (_getch()) {  
            case 'x':  
                return 0; // Quitter en appuyant sur 'x'  
                break;  
            case 'r':  
                vitesse = vitesse2;  
                break;  
            default:  
                break;  
        }  
    }  
}
```

Merci pour votre attention