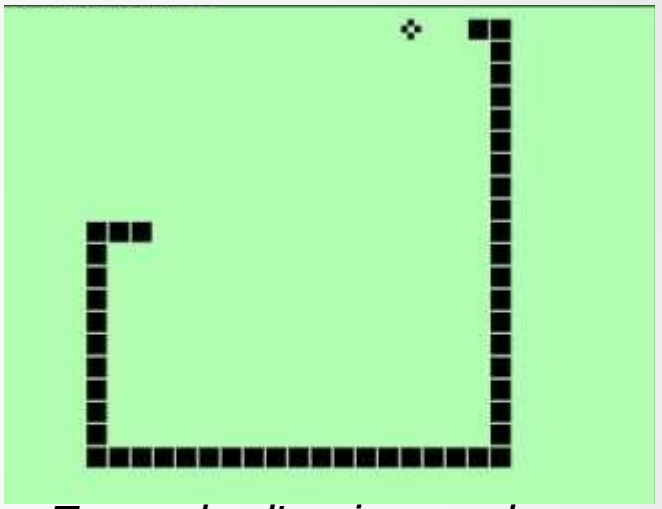
Projet I.S.N:Snake

Image du menu du jeu

I\ Introduction:

A\ Pourquoi avons-nous choisi ce projet:

- mini-jeu amusant
- Réalisable simplement avec Python



Exemple d'un jeu snake

B\ Principe du jeu snake:

- Le joueur déplace un serpent dans quatre directions.
- Il doit manger les fruit pour agrandir le serpent.
- Pour ne pas mourir il ne faut pas toucher les bords de l'écran ni que le serpent se mange lui-même.

III\ Début du projet:

- Recherche de différents codes de snake en python
- Compréhension globale des codes
- Mais difficultés à comprendre entièrement les codes
    - Écrit en anglais
    - Noms de variables compliqués

A\ Répartition des tâches:

- Étapes du codes définies par un étudiant en maths-informatique
- Création du code du jeu par Loïc et moi
- Création du menu par Kévin

Utilisation de pygame, bibliothèque multiplate-forme, pour le menu

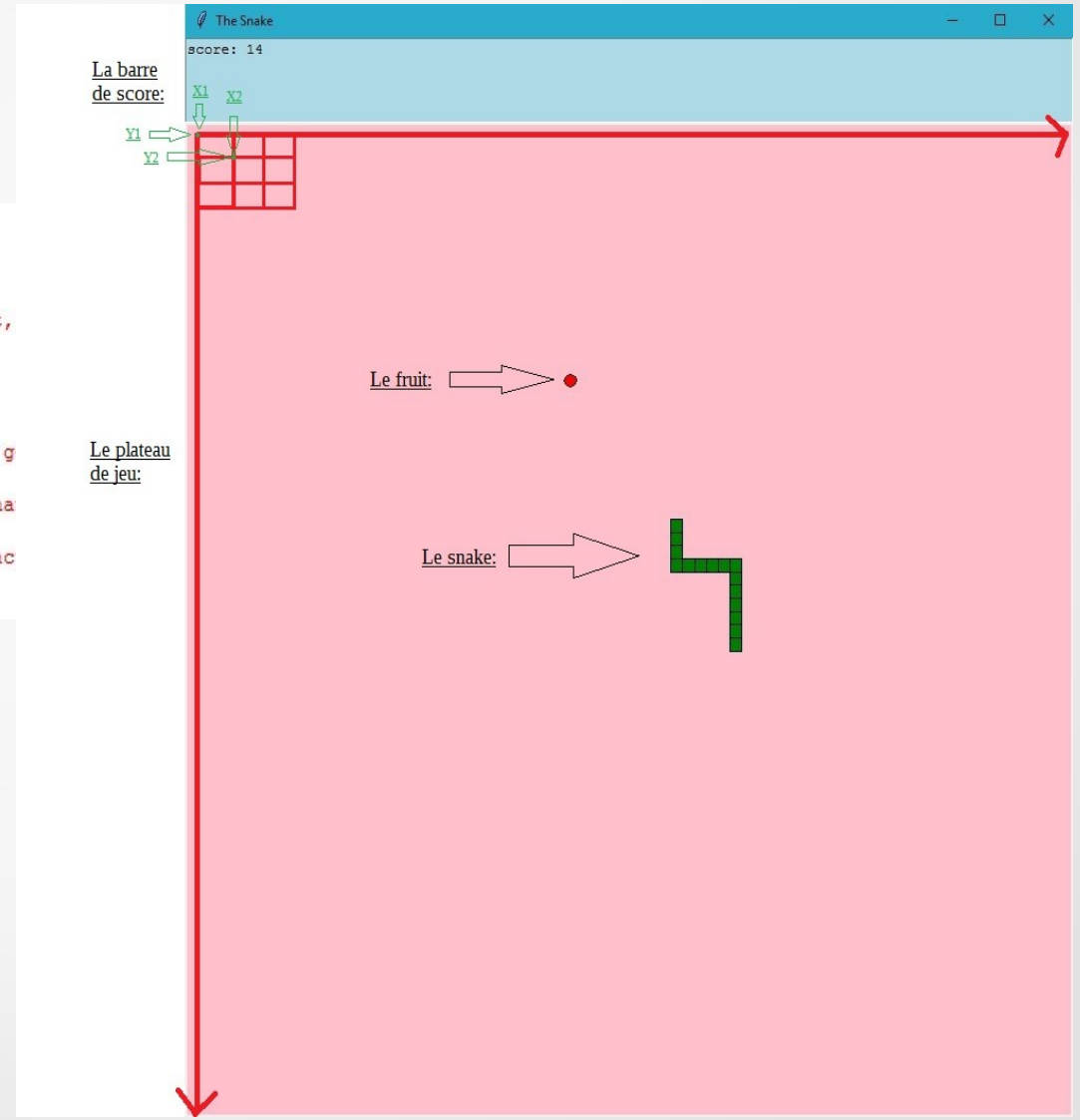
IV\ Réalisation du projet:

A\ Interface de la fenêtre de jeu:

```
#On récupère les dimensions de l'écran
hauteur = fenetre.winfo_screenheight()
largeur = fenetre.winfo_screenwidth()

#On convertie les données de la hauteur (H) et de la largeur (L) en int,
#et on modifie les dimensions voulues
H = str(int(hauteur/1.1))
L = str(int(largeur/2))

#On applique la taille voulue à la fenêtre grâce à l'appel de fonction g
# w, h sont respectivement la largeur et la hauteur
# X et Y sont les coordonnées du point d'origine relativement au coin ha
# le x et les deux + sont des séparateurs
# pour insérer les variables, on les concatène avec des chaînes de carac
fenetre.geometry(L + "x" + H + "+0+0")
```



B\ Créations du serpent et du fruit:

```
#Fonction qui détermine la taille des cases du plateau et qui les colore en vert pour symboliser le serpent
def remplir_case (x, y):

    #On défini les coordonnées (origine_caseX1; origine_caseY1) du point en haut à gauche de la case
    #et (origine_caseX2;origine_caseY2) du point en bas à droite de la case
    OrigineCaseX1 = x * LargeurCase
    OrigineCaseY1 = y * HauteurCase
    OrigineCaseX2 = OrigineCaseX1 + LargeurCase
    OrigineCaseY2 = OrigineCaseY1 + HauteurCase

    #remplissage du rectangle
    Plateau.create_rectangle(OrigineCaseX1, OrigineCaseY1, OrigineCaseX2, OrigineCaseY2, fill="green")

#On renvoie une case aléatoire
def case_aleatoire():

    AleatoireX = randint(0, NombreDeCases - 1)
    AleatoireY = randint(0, NombreDeCases - 1)

    return (AleatoireX, AleatoireY)

# affichee le serpent, l'argument étant la liste snake
def dessine_serpent(snake):

    #tant qu'il y a des cases dans snake
    for case in snake:

        # on récupère les coordonées de la case
        x, y = case
        # on colorie la case
        remplir_case(x, y)

#########################################################################################################################

#On retourne le chiffre 1 si la case est dans le snake, 0 sinon
def etre_dans_snake(case):

    if case in SNAKE:
        EtreDedans = 1
    else:
        EtreDedans = 0

    return EtreDedans

#On renvoie un fruit aléatoire qui n'est pas dans le serpent
def fruit_aleatoire():

    # choix d'un fruit aléatoire
    FruitAleatoire = case_aleatoire()

    # tant que le fruit aléatoire est dans le serpent
    while (etre_dans_snake(FruitAleatoire)):
        # on prend un nouveau fruit aléatoire
        FruitAleatoire = case_aleatoire
```

C\ Déplacement du serpent:

```
#Ces quatres fonctions permettent le déplacement dans quatres directions du serpent
#elles mettent à jour les coordonées du mouvement
def left_key(event):
    global MOUVEMENT
    MOUVEMENT = (-1, 0)

def right_key(event):
    global MOUVEMENT
    MOUVEMENT = (1, 0)

def up_key(event):
    global MOUVEMENT
    MOUVEMENT = (0, -1)

def down_key(event):
    global MOUVEMENT
    MOUVEMENT = (0, 1)

# indique les fonctions à appeler suite à une pression sur les flèches (ne fonctionne que si la fenêtre a le focus)
fenetre.bind("<Left>", left_key)
fenetre.bind("<Right>", right_key)
fenetre.bind("<Up>", up_key)
fenetre.bind("<Down>", down_key)
```

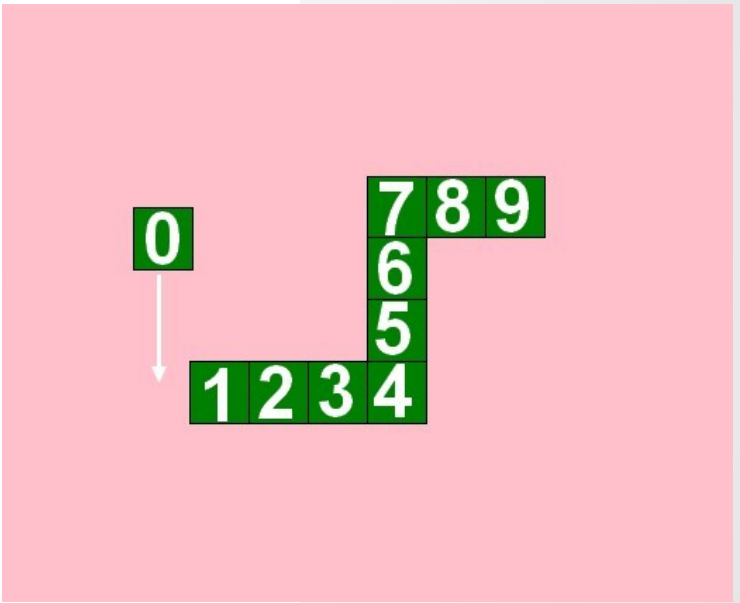
D\ Agrandissement du serpent:

```
# met à jour le snake
def mise_a_jour_snake():

    global SNAKE, FRUIT

    # on récupère les coordonées de la tête actuelle
    (AncienneTeteX, AncienneTeteY) = SNAKE[0]
    # on récupère les valeurs du mouvement
    MouvementX, MouvementY = MOUVEMENT
    # on calcule les coordonées de la nouvelle tête
    NouvelleTete = (AncienneTeteX + MouvementX, AncienneTeteY + MouvementY)
    # on vérifie si on a perdu
    serpent_mort(NouvelleTete)
    # on ajoute la nouvelle tête
    SNAKE.insert(0, NouvelleTete)

    # si on mange un fruit
    if NouvelleTete == FRUIT:
        # on génère un nouveau fruit
        FRUIT = fruit_aleatoire()
        # on met à jour le score
        mise_a_jour_score()
    # sinon
    else:
        # on enlève le dernier élément du serpent (c'est-à-dire: on ne grandit pas)
        SNAKE.pop()
```



E\ Fin de la partie:

```
# met à jour la variable PERDU indiquant si on a perdu
def serpent_mort(NouvelleTete):

    global PERDU

    NouvelleTeteX, NouvelleTeteY = NouvelleTete

    # si le serpent se mange lui-même (sauf au démarrage, c'est-à-dire: sauf quand MOUVEMENT vaut (0, 0))
    # OU si on sort du canvas
    if (etre_dans_snake(NouvelleTete) and MOUVEMENT != (0, 0)) or NouvelleTeteX < 0 or NouvelleTeteY < 0 or NouvelleTeteX >= NombreDeCases or NouvelleTeteY >= NombreDeCases:
        # alors, on a perdu
        PERDU = 1
```

```
# réinitialise les variables pour une nouvelle partie
def reinitialiser_jeu():

    global SNAKE, FRUIT, MOUVEMENT, SCORE, PERDU

    # serpent initial
    SNAKE = [case_aleatoire()]
    # fruit initial
    FRUIT = fruit_aleatoire()
    # mouvement initial
    MOUVEMENT = (0,0)
    # score initial
    SCORE = 0
    # variable perdu initiale (sera mise à1 si le joueur perd)
    PERDU = 0
```

Fonction principale:

```
# fonction principale
def tache():

    # on met à jour l'affichage et les événements du clavier
    fenetre.update
    fenetre.update_idletasks()
    # on met à jour le snake
    mise_a_jour_snake()
    # on supprime tous les éléments du plateau
    Plateau.delete("all")
    # on redessine le fruit
    dessine_fruit()
    # on redessine le serpent
    dessine_serpent(SNAKE)

    # si on a perdu
    if PERDU:
        # on efface la barre des scores
        Barre.delete(0.0, 3.0)
        # on affiche perdu
        Barre.insert(END, "Perdu avec un score de " + str(SCORE))
        # on prépare la nouvelle partie
        reinitialiser_jeu()
        # on rappelle la fonction principale
        fenetre.after(70, tache)
    # sinon
    else:
        # on rappelle la fonction principale
        fenetre.after(70, tache)
```

V\ Conclusion:

A\ Améliorations possibles du projet:

- mettre le jeu en pause

Faire varier le jeu et augmenter la difficulté

- ajouter des fruits malus
- augmenter la vitesse a partir d'un certain score