Projet Python
Comme un air de Tetris

RÉALISÉ PAR GUERRIER TOM ET LECARPENTIER MAEL

L1-R PROMOTION 2022-2027

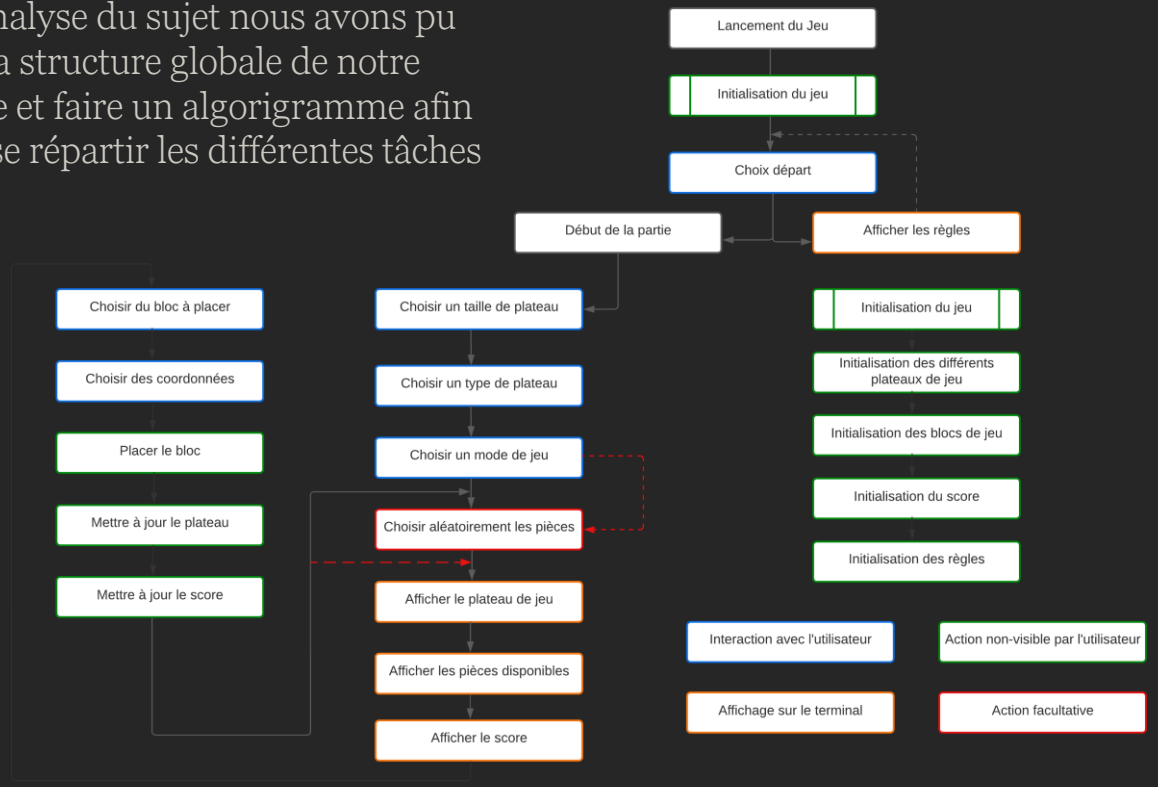
Sommaire :

Le Projet :

- Objectif : Le but est de réaliser une version modifiée du jeu Tetris en python. Le jeu devra permettre à l'utilisateur de choisir différents types de plateau, différentes tailles de plateau ainsi que différents mode de jeu. Le jeu suivra les règles d'un Tetris normal, à savoir chaque ligne ou colonne pleine se verra effacer, les blocs devront descendre lorsqu'une ligne est supprimée et le score augmentera à chaque bloc placé . La seule grande différence est que les blocs seront posés à l'aide d'une saisie de coordonnée. De plus, l'utilisateur pourra également sauvegarder sa partie et la rouvrir à n'importe quel moment.

Analyse du sujet et répartition des tâches :

Suite à l'analyse du sujet nous avons pu établir la structure globale de notre programme et faire un algorigramme afin de pouvoir se répartir les différentes tâches



Guerrier Tom
Création de l'interface utilisateur
Initialisation des règles
Placer le bloc, Mettre à jour le plateau et le score
Choix du bloc à placer et des coordonnées

Lecarpentier Mael
Initialisation des différents plateaux de jeu
Initialisation des blocs
Affichage du Plateau de jeu, des pièces et du score
Choix aléatoire des pièces

Nous avons décidé de se répartir les tâches de manières à ce que chacun d'entre nous puisse évoluer à son rythme sans attendre les résultats de l'autre afin de perdre le moins de temps possible

Les différentes fonctionnalités disponibles :

- L'utilisateur peut lire les règles du jeu ou jouer directement.
- Il peut choisir la taille et la forme de son plateau de jeu.
- Il a le choix entre 2 modes de jeu (un mode aléatoire qui remplace automatiquement la pièce placée sur le plateau et un autre mode fixe qui lui propose 3 pièces qui resteront les mêmes durant toute la partie).
- Il peut également choisir la pièce qu'il veut et la positionner sur le plateau à l'aide de coordonnées.
- Les lignes et colonnes pleines se suppriment automatiquement et un décalage vers le bas est effectué si une ligne est supprimée.
- Plus le score est important, plus la difficulté augmente en proposant des blocs de plus en plus grands et difficiles à placer.
- À la fin de la partie, l'utilisateur peut voir ces statistiques (nombre de pièces posées, son score, nombre le ligne et colonne remplies).

Les différentes fonctionnalités à ajouter :

- Permettre à l'utilisateur de pouvoir sauvegarder sa partie et la réouvrir lorsqu'il le souhaite.
- Pouvoir effectuer la rotation du bloc choisie.
- Créer un score board qui enregistre le score des anciennes parties.

Explication Cercle () :

```
20  def Cercle (Tplateau):
21      A = []
22      ligne = Tplateau + 4
23      colonne = Tplateau + 4
24
25      for i in range(ligne):
26          B = []
27          for j in range(colonne):
28              if (i == 0 or i == ligne - 1) and (j < 2 or j > Tplateau + 1):  # Decalage debut du plateau
29                  caractere = chr(32)
30              elif j == 0 and (i == 1 or i == Tplateau + 2):  # Decalage fin du plateau
31                  caractere = chr(32)
32              elif (i == 0 or i == Tplateau + 3) and j > 1 and j < Tplateau + 2:  # Remplissage ligne de caracteres minuscule
33                  caractere = chr(97 + j - 2)
34              elif (j == 0 or j == Tplateau + 3) and i > 1 and i < Tplateau + 2:  # Remplissage ligne de caracteres majuscule
35                  caractere = chr(65 + i - 2)
36              elif i == 1 or i == Tplateau + 2 and j > 0 and j < Tplateau + 4:  # Remplissage de la bordure haut et bas
37                  caractere = chr(61)
38              elif j == 1 or j == Tplateau + 2 and i > 1 and j < Tplateau + 3:  # Remplissage de la bordure droite et gauche
39                  caractere = chr(124)
40              elif i > 1 and i < 7 and j > 1 and j < 7 - i:  # Decalage haut gauche
41                  caractere = 0
42              elif i > 1 and i < 7 and j > Tplateau + i - 4:  # Decalage haut droit
43                  caractere = 0
44              elif i > Tplateau - 2 and j > 1 and j < i - Tplateau + 4:  # Decalage bas gauche
45                  caractere = 0
46              elif i > Tplateau - 2 and j > Tplateau - (i + 1) + Tplateau:  # Decalage bas droit
47                  caractere = 0
48              else:
49                  caractere = 1
50              B.append(caractere)
51          A.append(B)
```

Pour la création du plateau de jeu nous avons choisi d'utiliser une matrice 2D afin de pouvoir changer facilement les valeurs présentes dans le tableau. La première partie (de la ligne 28 à 39) consiste à créer le cadre du plateau, cette partie est la même pour les 3 formes. La deuxième partie (de la ligne 40 à 49) consiste à remplir l'intérieur du cadre, nous avons choisi par simplicité de remplir seulement les zones en dehors de la surface de jeu par des espaces et le reste de la matrice par des points qui représentent la grille de jeu. Cependant, cette partie est différentes pour chaque type de plateau.

Explication Choix_bloc () :

```python
def Choix_Bloc ():

    import random
    import numpy as np

    commun =    [[[3,3,3,3,3],[3,3,3,3,3],[3,3,3,3,3],[2,3,3,3,3],[2,2,3,3,3]],[[3,3,3,3,3],[3,3,3,3,3],[3,3,3,3,3],[3,2,3,3,3],[2,2,3,3,3]],
                [[3,3,3,3,3],[3,3,3,3,3],[3,3,3,3,3],[2,3,3,3,3],[2,2,2,3,3]],[[3,3,3,3,3],[3,3,3,3,3],[2,2,3,3,3],[3,2,3,3,3],[4,2,3,3,3]]]


    cercle =    [[[3,3,3,3,3],[2,2,2,2,3],[2,2,2,2,3],[2,2,2,2,3],[2,2,2,2,3]],[[3,3,3,3,3],[3,2,2,3,3],[2,2,2,2,3],[2,2,2,2,3],[4,2,2,3,3]],
                [[3,3,3,3,3],[2,3,3,2,3],[2,3,3,2,3],[2,3,3,2,3],[2,2,2,2,3]],[[3,3,3,3,3],[2,2,2,2,3],[3,3,3,2,3],[3,3,3,2,3],[4,3,3,2,3]],]

    # Utilisation de listes pour choisir le type et le rang des pieces

    Type = [1, 2]
    rang_commun = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19]
    rang_cercle = [0,1,2,3,4,5,6,7,8,9,10,11]

    # Choix aleatoire du type des pieces

    Tpiece1 = random.choice(Type)

    # Choix aleatoire du rang des pieces dans les listes 3D
    if Tpiece1 == 1:
        A = np.random.choice(rang_commun, 1, p=[0.25,0.25,0.25,0.25])
        Piece1 = commun[A[0]]
    else:
        A = np.random.choice(rang_cercle, 1, p=[0.25,0.25,0.25,0.25])
        Piece1 = cercle[A[0]]

    return Piece1
```

Ceci est un version réduite du code afin de simplifier l'explication

Nous avons utilisé des listes afin de déterminer le type (pièce commune ou spécifique) et le rang de chaque pièce (dans la liste 3D)

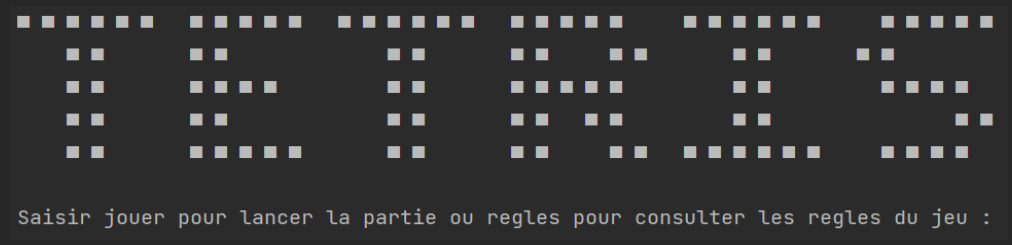
Afin de choisir aléatoirement les différents blocs à chaque tour, nous utilisons la librairie random. De plus, pour pouvoir augmenter la difficulté nous utilisons également la librairie numpy pour définir la probabilité que chaque bloc soit choisi (ligne 25 et 28). Lorsque nous disposons du rang de la pièce dans la liste 3D et de son type, il nous suffit d'ajouter la valeur de cette case à notre variable de Pièce (ligne 26 et 29)

Explication Place_bloc () :

```
170  def Place_bloc (ligne,colonne,plateau,Piece_choisie,score):
171      test = True
172      x = 0
173      y = 0
174      while (x < 1 or x > colonne - 2) : # Saisie des coordonnees horizontale
175          x = ord(str(input("Saisir une lettre minuscule : "))) - 95
176      while (y < 1 or y > ligne - 2) :  # Saisie des coordonnees verticale
177          y = ord(str(input("Saisir une lettre majuscule : "))) - 63
178
179      for k in range(4, -1, -1):  # Verification de l'emplacement choisie
180          for l in range(5):
181              if test == True:
182                  if Piece_choisie[k][l] == 2 and plateau[y][x] != 1:
183                      test = False
184                  else:
185                      x += 1
186          y -= 1
187          x -= 5
188      y += 5
189      if test == False:
190          print("Il n'y a pas la place necessaire pour poser votre piece au coordonees saisie !!")
191      else:
192          for k in range(4, -1, -1):  # Placement de la piece
193              for l in range(5):
194                  if Piece_choisie[k][l] == 2:
195                      plateau[y][x] = 2
196                      score += 1
197                  x += 1
198              y -= 1
199              x -= 5
200
201      return plateau,score
```

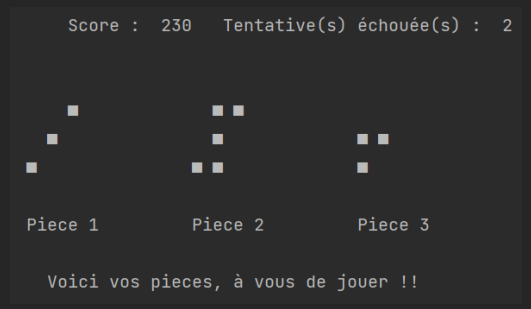
Dans la première partie (ligne 172 à 177), nous vérifions que les coordonnées saisies par l'utilisateur soient bien comprises dans le plateau, puis nous transformons les lettres en chiffres à l'aide de leur code ASCII. Dans la deuxième partie (ligne 179 à 185), nous vérifions s'il y a la place disponible pour placer notre pièce en parcourant en même temps notre pièce et notre matrice. Si il n'y a pas de place, le bloc n'est pas placé et un message d'erreur s'affiche pour signaler à l'utilisateur que les coordonnées saisies ne conviennent pas. Dans le cas où il y a la place, nous refaisons la même manipulation et lorsque l'on croise un bloc sur notre pièce, on modifie notre matrice principale. À savoir qu'un 0 signifie une place en dehors du plateau, un 1 signifie une place libre et un 2 signifie une place déjà occupée par un bloc

Interface utilisateur :



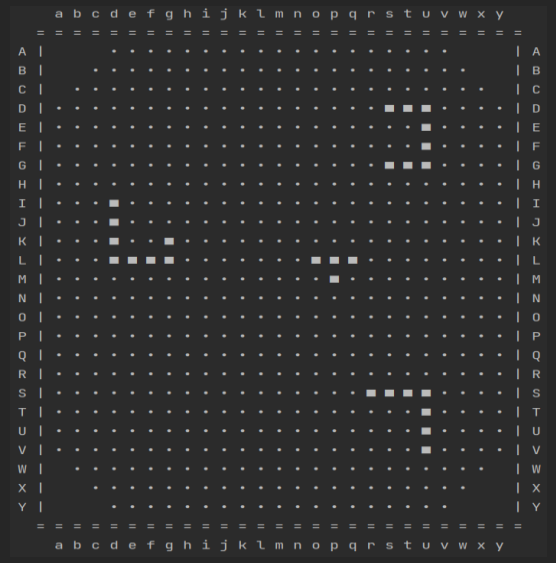
Début de partie

Fin de partie



Score board

Plateau : Triangle



Plateau : Cercle

Plateau : Losange

Conclusion

- Ce projet à été l'occasion de mettre en application nos compétences acquises en python sur un cas concret, de découvrir de nouvelles librairies utilisables sur Python (numpy et random) et d'aguerrir notre maîtrise de PyCharm.
- Nous avons mis l'accent sur notre coordination dès le début du projet pour nous permettre d'optimiser le temps de travail et la répartition de la charge. Cette organisation nous a permis de remplir le contrat dans les temps, toutefois nous nous étions fixé l'objectif de développer l'ensemble du projet, mais les tests fonctionnels ont pris plus de temps que prévu initialement et ont eu raison de nos ambitions.