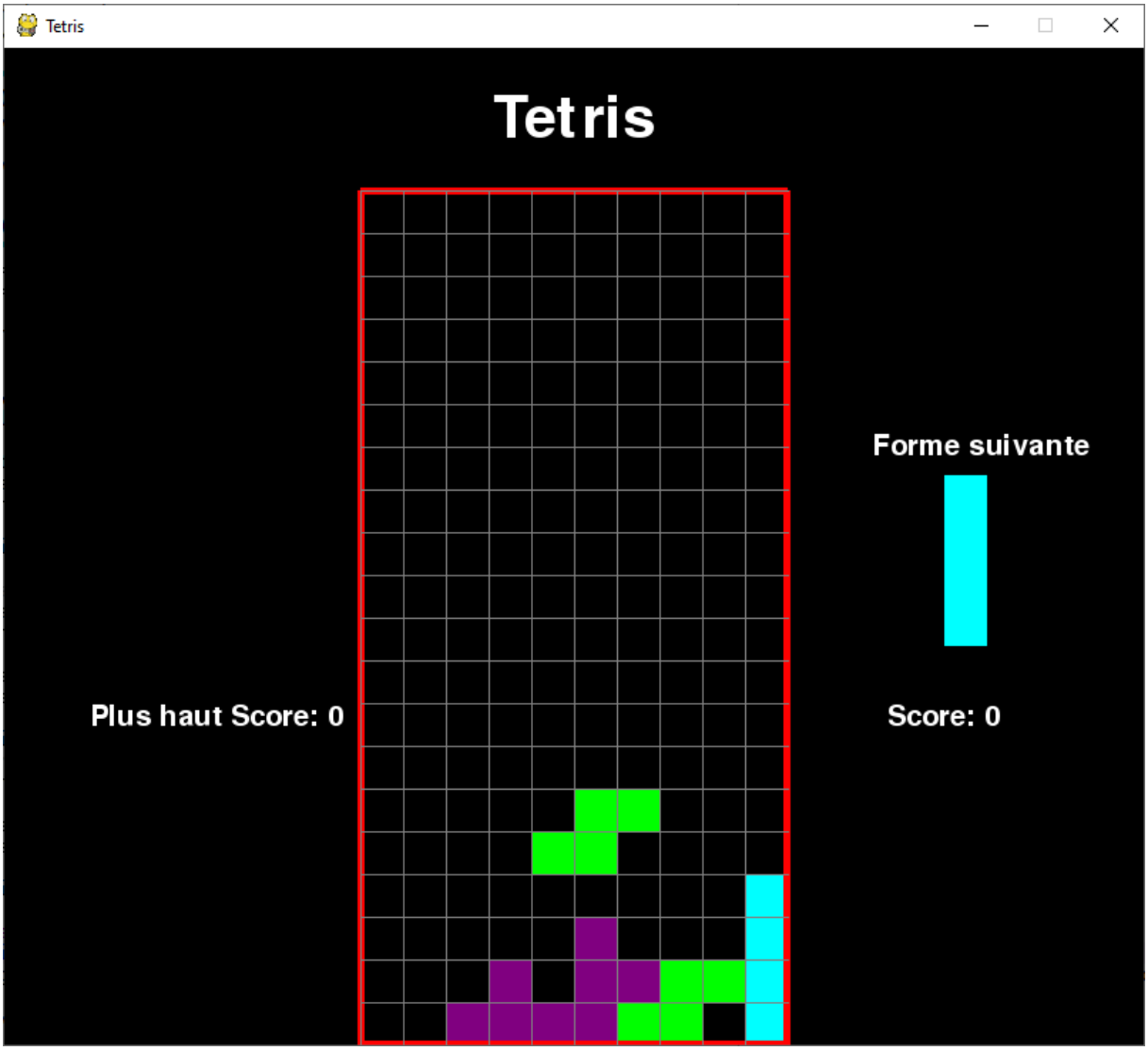
Tétris 2

Le jeu



Mémoriser l'emplacement des tétriminos dans la zone de jeu

Nous allons utiliser une liste multiple, contenant 20 listes de 10 éléments (lignes, colonnes).

Chaque élément d'une liste est un tuple indiquant la couleur de la pièce à la position correspondante.

Nous pourrons ainsi tracer facilement dans la zone de jeu, tous les petits carrés colorés.

Voir la documentation concernant la création de liste par « **compréhension** » : (partie 5.1.3)

https://docs.python.org/fr/3/tutorial/datastructures.html

Au cours du jeu, une piece tombe et la position de ses carrés colorés change.

Certaines pièces, arrivées au fond de la grille ne changent plus de place.

Nous allons mémoriser dans un « dictionnaire » la position des carrés qui ne changent plus de couleur et utiliser ce dictionnaire pour reconstituer la grille avant son affichage.

Voir la documentation concernant les « dictionnaires » : (partie 5.5)

https://docs.python.org/fr/3/tutorial/datastructures.html

```
def creer_grille(emplacement_bloque={}):
    #Création d'une grille vide constituée de nb_lignes listes
    #Chaque liste contient nb_colonnes tuples (0,0,0) = couleur noire
    grille = [[(0,0,0) for i in range(nb_colonnes)] for j in range(nb_lignes)]
    #On parcourt chaque position de la grille.
    #Si une position est une des clés du dictionnaire emplacement_bloque
    #on utilise la clé pour aller chercher la couleur du carré à cet emplacement
    #et on positionne cette couleur dans la grille.
    for i in range(len(grille)):
        for j in range(len(grille[i])):
            if (j, i) in emplacement_bloque:
                c = emplacement_bloque[(j,i)]
                grille[i][j] = c
    return grille
```

Afficher la grille

Nous modifions la fonction afficher_grille de façon à ce qu'elle affiche les carrés colorés et les lignes colorées.

La fonction utilise un nouveau paramètre : la grille contenant la couleur de chaque emplacement.

```
def afficher_grille(surface, grille):
    sx = top_left_x
    sy = top_left_y
    #afficher les carrés colorés
```

```
    for i in range(len(grille)):
        for j in range(len(grille[i])):
            pygame.draw.rect(surface, grille[i][j], (top_left_x + j*taille_bloc, top_left_y + i*taille_bloc, taille_bloc, taille_bloc), 0)
    #afficher le cadre rouge autour de la zone de jeu
    pygame.draw.rect(surface, (255, 0, 0), (top_left_x, top_left_y, largeur_zoneJeu, hauteur_zoneJeu), 5)
    #afficher les lignes grises de la grille dans la zone de jeu
    for i in range(nb_lignes):
        pygame.draw.line(surface, (128,128,128), (sx, sy + i*taille_bloc), (sx+largeur_zoneJeu, sy+ i*taille_bloc))
    for j in range(nb_colonnes):
        pygame.draw.line(surface, (128, 128, 128), (sx + j*taille_bloc, sy),(sx + j*taille_bloc, sy + hauteur_zoneJeu))
```

Test 1

Nous passons le paramètre « grille » à la fonction afficher_fenetre, afin que cette fonction puisse le passer à afficher_grille.

```
def afficher_fenetre(surface,grille, score="0", max_score = "0"):
…
    #afficher la grille de la zone de jeu
    afficher_grille(surface, grille)
```

Partie principale :

```
#==========================================================
ecran = pygame.display.set_mode((largeur_ecran, hauteur_ecran))
pygame.display.set_caption('Tetris')
pygame.font.init()
#Nous créons un dictionnaire avec 4 clés et 4 valeurs. Une clé = (colonne, ligne)
#les carrés de coordonnées (0, 19) à (3,19) sont des carrés bloqués en bleu
```

emplacement_bloque=dict([((0,19), (0,0,255)), ((1,19), (0,0,255)), ((2,19), (0,0,255)),((3,19), (0,0,255))])

#Nous créons la grille à afficher

grille = creer_grille(emplacement_bloque)

afficher_fenetre(ecran,grille)

pygame.display.update()

run = True

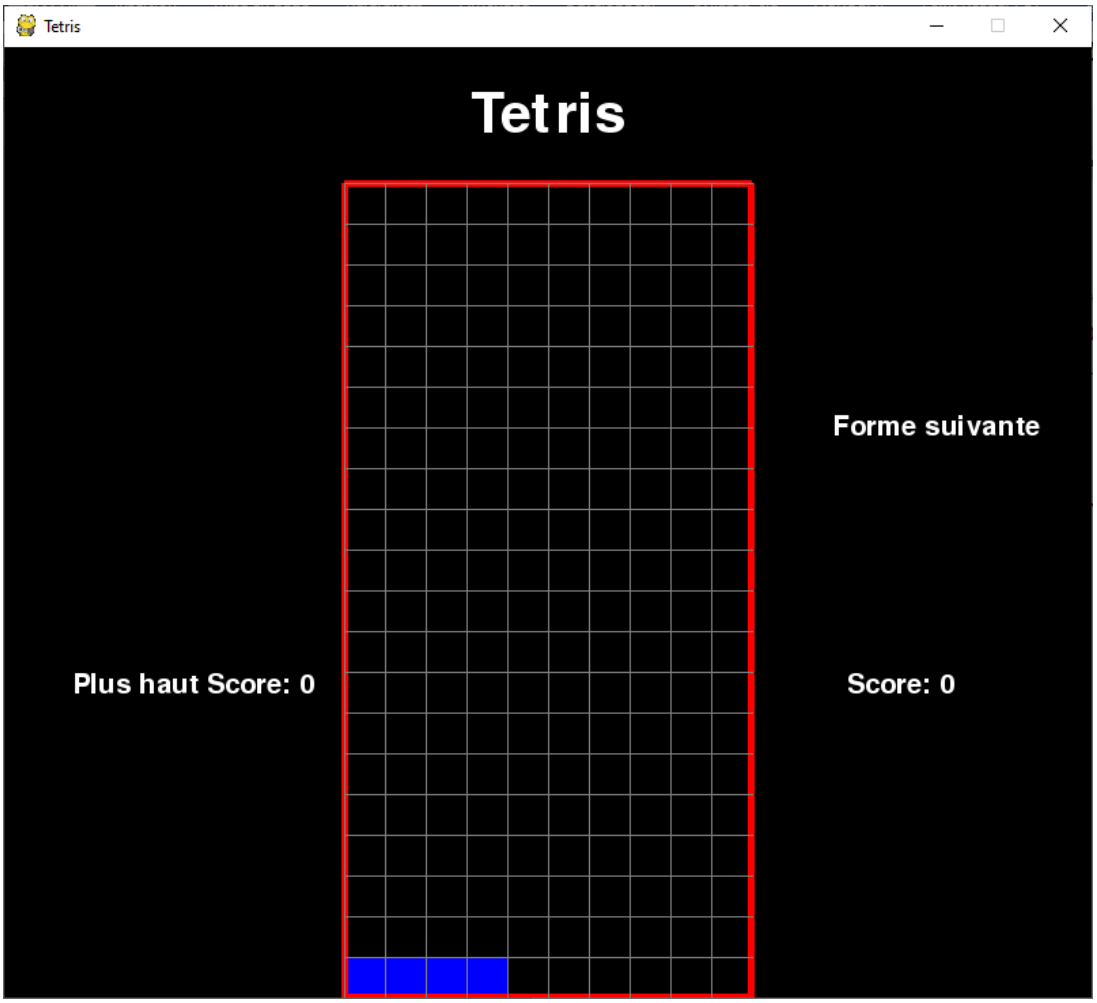
while run:

  for event in pygame.event.get():

    if event.type == pygame.QUIT:

      run = False

      pygame.quit()



Choix du tétrimino à faire tomber

Le choix du prochain tétrimino à faire tomber doit être aléatoire.

```
def choix_piece():
    return Piece(5, 0, math.random.choice(formes))
```

x = 5, car la bordure rouge fait 5 pixels.

Comment afficher la pièce qui tombe

Chaque pièce est représentée par une liste multiple, contenant toutes ses formes possibles.

Nous devons donc, en fonction de la valeur « rotation » d'une pièce, extraire de la liste, la forme à afficher et traduire cette forme en une liste de position des carrés colorés.

Une fois les positions des carrés colorés trouvées, il suffira d'ajouter ces positions dans la grille à afficher.

```
def convertir_forme(piece):
    positions = []
    #la forme à afficher suivant la valeur de la rotation :
    #il faut trouver la liste définissant cette forme
    forme = piece.tetrimino[piece.rotation % len(piece.tetrimino)]
    #la forme est une liste contenant des lignes formées de points et de 0
    #i = numéro de la ligne dans la forme.
    #ligne = i ième chaine de caractères composées de . et de 0, de forme
    for i, ligne in enumerate(forme):
        #row = liste formée avec les caractères (point ou 0) de la ligne.
        row = list(ligne.strip())
        #j = numéro caractère dans la liste row
        #colonne = j ième caractère de la liste row : 0 ou .
        for j, colonne in enumerate(row):
            if colonne == '0':
                positions.append((forme.x + j, forme.y + i))
```

```
    #on centre la pièce et on la place de façon à ce quelle démarre en haut de la grille
    for i, pos in enumerate(positions):
        positions[i] = (pos[0] - 2, pos[1] - 4)
    return positions
```

Test 2

```
#=========================================================
ecran = pygame.display.set_mode((largeur_ecran, hauteur_ecran))
pygame.display.set_caption('Tetris')
pygame.font.init()
emplacement_bloque={}
#choix de la piece à faire tomber
piece_enCours = choix_piece()
run = True
clock = pygame.time.Clock()
while run:
    pygame.time.delay(50)
    clock.tick(2)
    #créer une nouvelle grille, ne contenant que les emplacements bloqués
    grille = creer_grille(emplacement_bloque)
  for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False


  #faire descendre la pièce d'une ligne
    piece_enCours.y+=1
    #tester si la pièce est arrivée au fond
    if piece_enCours.y> 19:
        run=False
```

```
    #Récupérer les positions des cases colorées de la pièce
    position_piece = convertir_forme(piece_enCours)
    #Ajouter la couleur de ces cases dans la grille à afficher
    for i in range(len(position_piece)):
        x, y = position_piece[i]
        if y > -1:
            grille[y][x] = piece_enCours.couleur
    #Mettre à jour la fenêtre
    afficher_fenetre(ecran,grille)
    pygame.display.update()

pygame.quit()
```

Faire tourner ou déplacer les pièces lorsqu'elles tombent

Nous allons utiliser les flèches pour déplacer ou faire tourner une pièce.

Flèche haut → faire tourner la pièce

Flèche bas → faire descendre la pièce d'une ligne

Flèche gauche → déplacer la pièce d'une colonne à gauche

Flèche droite → déplacer la pièce d'une colonne à droite

Test 3

Ajoutons simplement la prise en compte de ces flèches dans la boucle de jeu

```
for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT:
                piece_enCours.x -= 1
                if piece_enCours.x < 0:
                    piece_enCours.x += 1
```

```
if event.key == pygame.K_RIGHT:
    piece_enCours.x += 1
    if piece_enCours.x > 9:
        piece_enCours.x -= 1
if event.key == pygame.K_DOWN:
    piece_enCours.y += 1
    if piece_enCours.y > 19:
        piece_enCours.y -= 1
if event.key == pygame.K_UP:
    piece_enCours.rotation += 1
    if piece_enCours.rotation>3:
        piece_enCours.rotation = 0
```

On remarquera que la prise en compte des flèches est un peu poussive.

En effet à chaque tour de boucle, nous arrêtons le programme pendant un certain temps :

```
pygame.time.delay(50)
```

Il faut donc trouver un moyen de mettre en œuvre un chronomètre qui ne servira qu'à faire tomber la pièce, sans bloquer le programme afin qu'il puisse prendre en compte l'appui sur les flèches sans retard.

Compteur personnalisé

La fonction pygame.time.set_timer(Evenement, millisecondes), place l'évènement dans la file des évènement, toutes les millisecondes.

L'évènement peut être un évènement personnalisé que l'on peut définir de la façon suivante :

CHUTE_PIECE = pygame.USEREVENT + 1

Puis pour créer le chronomètre :

pygame.time.set_timer(CHUTE_PIECE, 1000)

Il suffit ensuite de traiter l'évènement CHUTE_PIECE dans la boucle de jeu, de la même façon que l'on traite les autres évènements : QUIT, KEYDOWN etc.

La partie principale devient :

```
#=========================================================
ecran = pygame.display.set_mode((largeur_ecran, hauteur_ecran))
pygame.display.set_caption('Tetris')
pygame.font.init()
emplacement_bloque={}
piece_enCours = choix_piece()
run = True
#évènement personnalisé
CHUTE_PIECE = pygame.USEREVENT + 1
#créer un chronomètre qui déclenche l'évènement toute les seconde
pygame.time.set_timer(CHUTE_PIECE, 1000)
while run:
    #créer une nouvelle grille, ne contenant que les emplacements bloqués
    grille = creer_grille(emplacement_bloque)
    #-------------------------------------------------------------
    #Prendre en compte les évènements
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False
        #Tester si l'évènement CHUTE_PIECE a été déclenché.
        if event.type == CHUTE_PIECE:
            #faire descendre la pièce d'une ligne
            piece_enCours.y+=1
            #tester si la pièce est arrivée au fond
            if piece_enCours.y> 19:
                run=False
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT:
```

```
                piece_enCours.x -= 1
                if piece_enCours.x < 0:
                    piece_enCours.x += 1
            if event.key == pygame.K_RIGHT:
                piece_enCours.x += 1
                if piece_enCours.x > 9:
                    piece_enCours.x -= 1
            if event.key == pygame.K_DOWN:
                piece_enCours.y += 1
                if piece_enCours.y > 19:
                    piece_enCours.y -= 1
            if event.key == pygame.K_UP:
                piece_enCours.rotation += 1
                if piece_enCours.rotation>3:
                    piece_enCours.rotation = 0
    #---------------------------------------------------------------------
    #Récupérer les positions des cases colorées de la pièce
    position_piece = convertir_forme(piece_enCours)
    #Ajouter la couleur de ces cases dans la grille à afficher
    for i in range(len(position_piece)):
        x, y = position_piece[i]
        if y > -1:
            grille[y][x] = piece_enCours.couleur
     #Mettre à jour la fenêtre
    afficher_fenetre(ecran,grille)
    pygame.display.update()

pygame.quit()
```