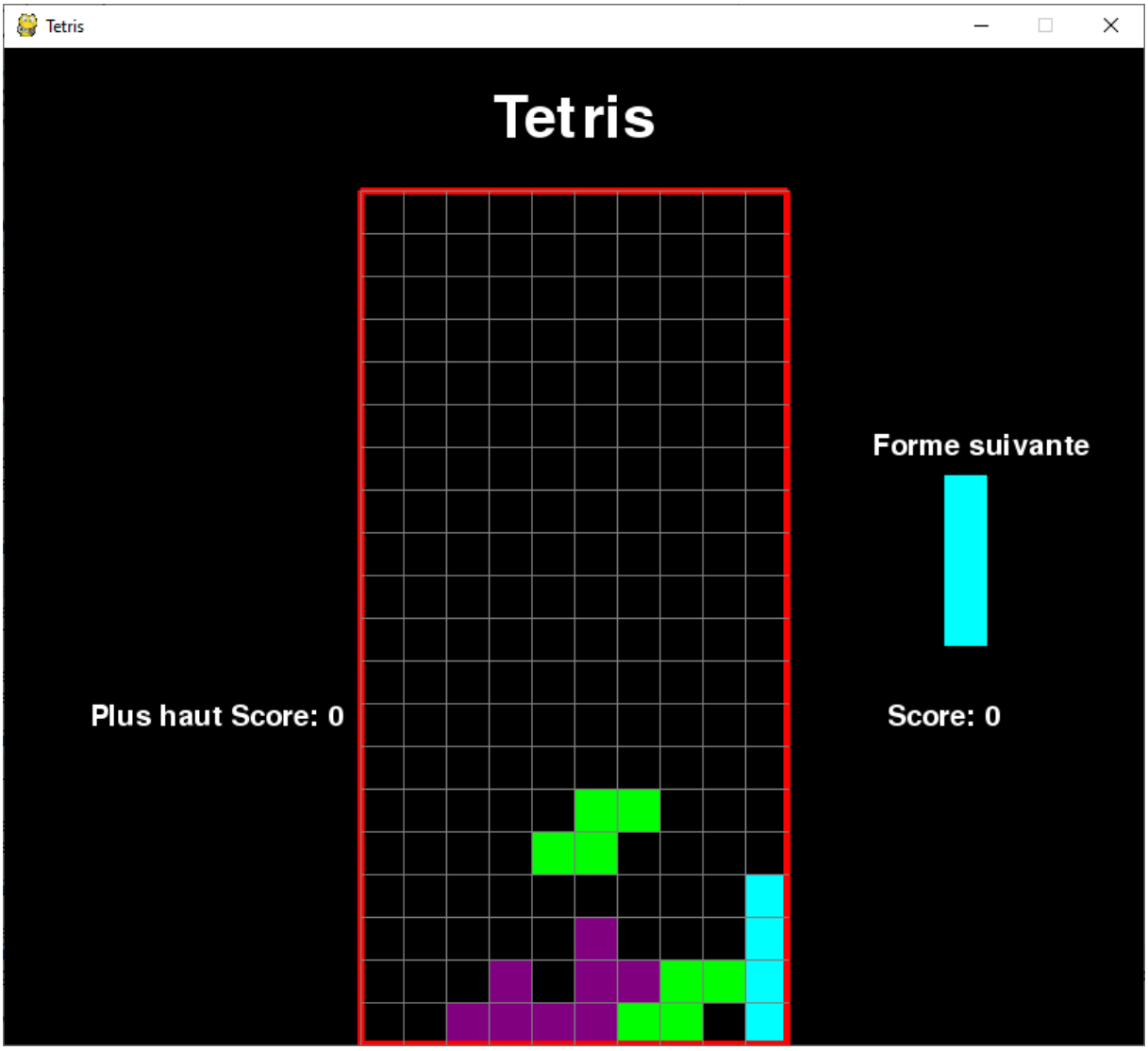
Tétris 1

Le jeu



Tetris met le joueur au défi de réaliser des lignes complètes en déplaçant des pièces de formes différentes, les tétriminos, qui défilent depuis le haut jusqu'au bas de l'écran.

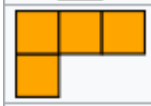
Les tétriminos peuvent être **déplacés, retournés** pour s'imbriquer au mieux les uns dans les autres.

Les lignes complétées disparaissent tout en rapportant des points et le joueur peut de nouveau remplir les cases libérées.

Le jeu n'a pas de fin : le joueur perd la partie lorsqu'un tétrimino reste bloqué en haut.

Il doit donc résister le plus longtemps à la chute continue des tétriminos, afin de réaliser le meilleur score.

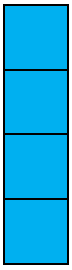
Les Tétriminos :

Forme	Appellation
	Tétrimino I
	Tétrimino O
	Tétrimino T
	Tétrimino L
	Tétrimino J
	Tétrimino Z
	Tétrimino S

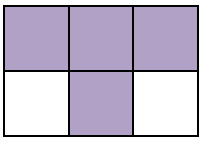
Analyse du programme à réaliser

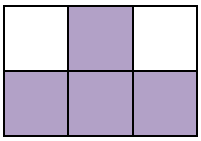
Dans ce premier programme, nous allons :

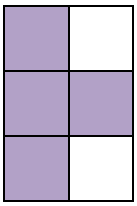
1. Concevoir l'interface du jeu, tel que présentée ci-dessus.
2. Réfléchir à la façon de représenter en mémoire les différents tétriminos, sachant que ceux-ci peuvent être basculés et donc prendre une allure différente : par exemple le tétrimino L peut être affiché horizontalement ou verticalement.

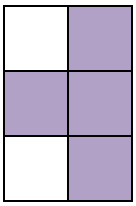


Le tétrimino T peut être représenté sous 4 positions différentes :









3. Afficher un à un tous les tétriminos en dessous du texte « Forme suivante »

L'interface

La zone dans laquelle les tétriminos tombent, est constituées d'une grille de 10 X 20 carrés.

Si un carré fait 30 pixels, cette zone doit avoir pour dimensions :

Hauteur = 30 X 20 = 600 pixels

Largeur = 30 X 10 = 300 pixels

Nous définissons les variables globales suivantes :

```
# Variables globales
nb_colonnes = 10
nb_lignes = 20
largeur_ecran = 800
hauteur_ecran = 700
taille_bloc = 30
#---------------------------------------
largeur_zoneJeu = taille_bloc * nb_colonnes
hauteur_zoneJeu = taille_bloc * nb_lignes
#coordonnées du coin en haut à gauche de la zone de Jeu
top_left_x = (largeur_ecran - largeur_zoneJeu) // 2
top_left_y = hauteur_ecran - hauteur_zoneJeu
```

Fonction affichant la grille dans la zone de jeu

```
def afficher_grille(surface):
    sx = top_left_x
    sy = top_left_y
    #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))
```

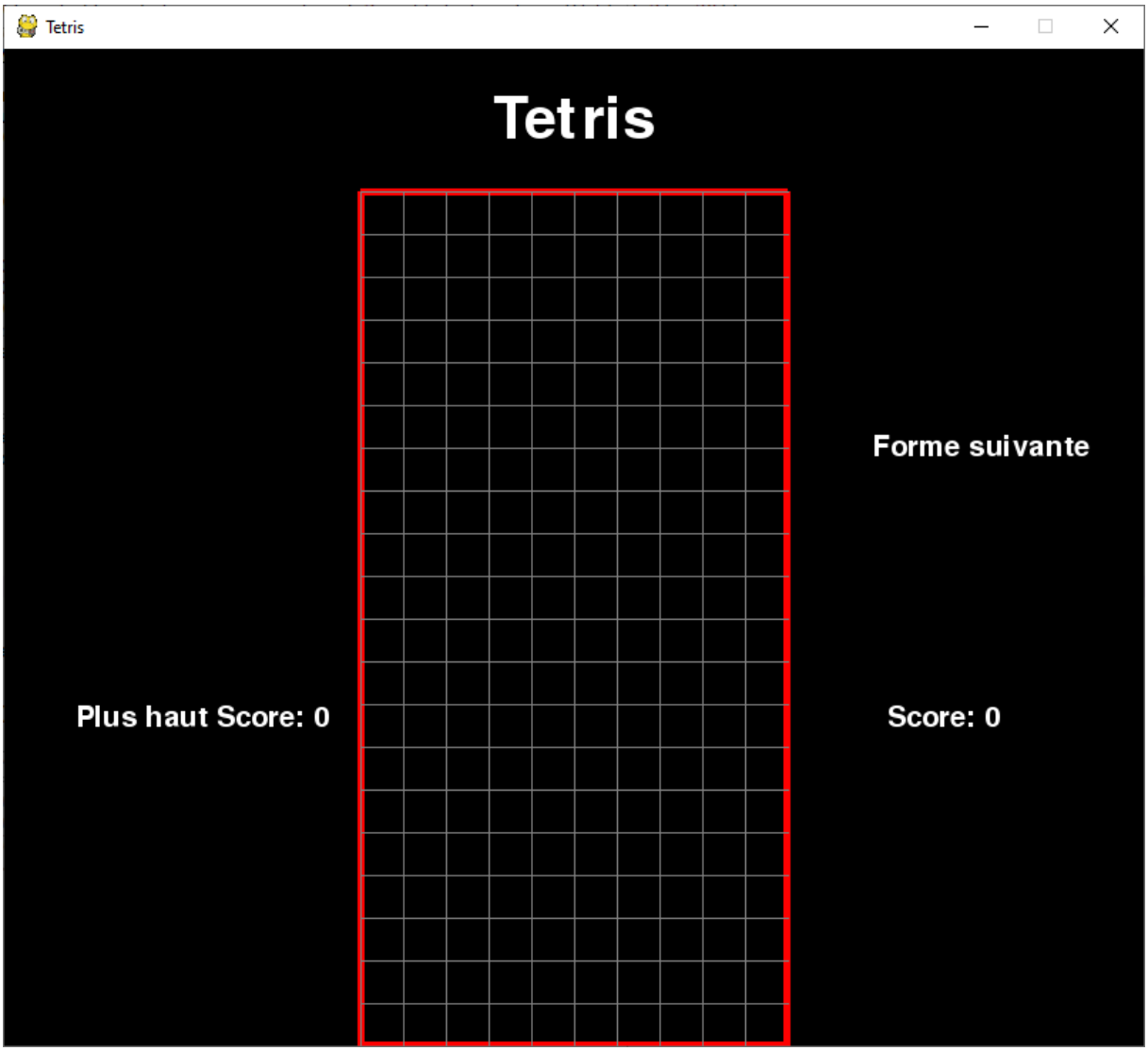
Fonction affichant les éléments de la fenêtre

```
def afficher_fenetre(surface, score="0", max_score = "0"):
    #mettre toute la fenêtre en noir
    surface.fill((0, 0, 0))
    #Titre : Tétris
    font = pygame.font.SysFont('comicsans', 60)
    label = font.render('Tetris', 1, (255, 255, 255))
    surface.blit(label, (top_left_x + largeur_zoneJeu / 2 - (label.get_width() / 2), 30))
    #Texte: forme suivante
    font = pygame.font.SysFont('comicsans', 30)
    label = font.render('Forme suivante', 1, (255,255,255))
    sx = top_left_x + largeur_zoneJeu + 50
    sy = top_left_y + hauteur_zoneJeu/2 - 100
    surface.blit(label, (sx + 10, sy - 30))
    # Texte: score
    label = font.render('Score: ' + str(score), 1, (255,255,255))
```

```
    surface.blit(label, (sx + 20, sy + 160))
    # Texte : max score
    label = font.render('Plus haut Score: ' + max_score, 1, (255,255,255))
    sx = top_left_x - 200
    sy = top_left_y + 200
    surface.blit(label, (sx , sy + 160))
    #afficher la grille de la zone de jeu
    afficher_grille(surface)
```

Test :

```
#=========================================================
ecran = pygame.display.set_mode((largeur_ecran, hauteur_ecran))
pygame.display.set_caption('Tetris')
pygame.font.init()
afficher_fenetre(ecran)
pygame.display.update()
run = True
while run:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False
            pygame.quit()
```



Définir les tétriminos

Chaque forme d'un tétrimino peut être définie de la façon suivante :

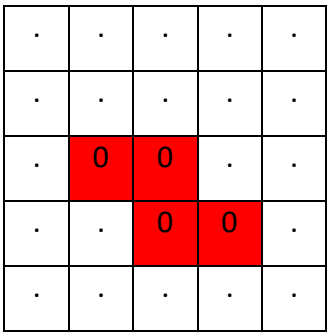
Tétrimino S : forme 1

.	.	.	.	.
.	.	.	.	.
.	.	0	0	.
.	0	0	.	.
.	.	.	.	.

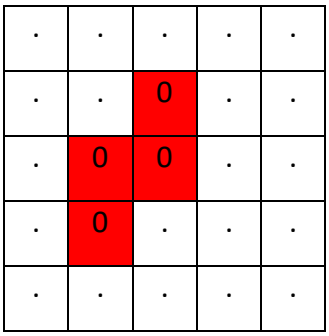
Tétrimino S : forme 2

.	.	.	.	.
.	.	0	.	.
.	.	0	0	.
.	.	.	0	.
.	.	.	.	.

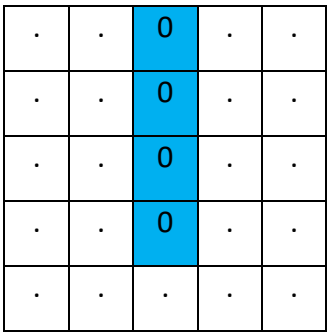
Tétrimino Z : forme 1



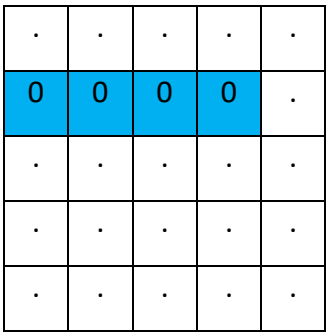
Tétrimino Z : forme 2



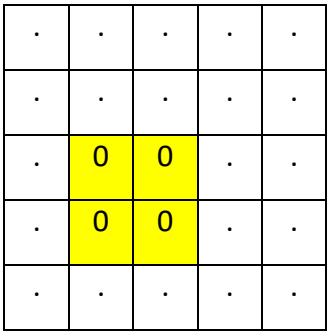
Tétrimino I: forme 1



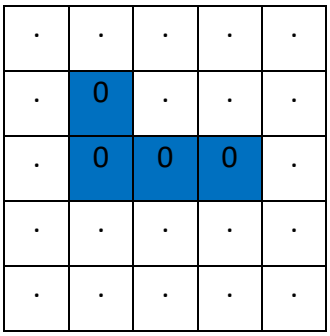
Tétrimino I: forme 2



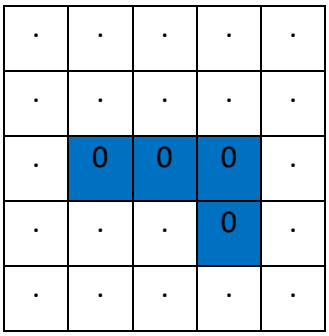
Tétrimino 0: Une seule forme



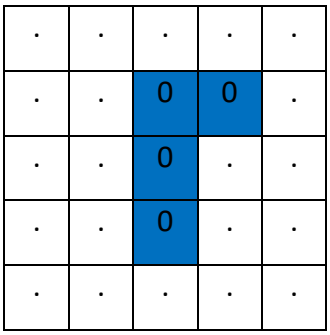
Tétrimino J : forme 1



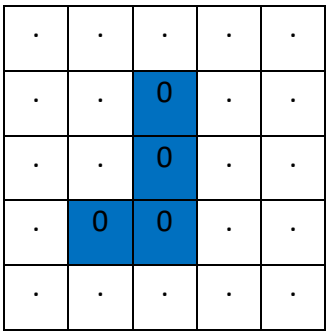
Tétrimino J : forme 3



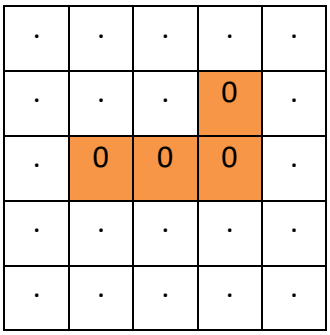
Tétrimino J : forme 2



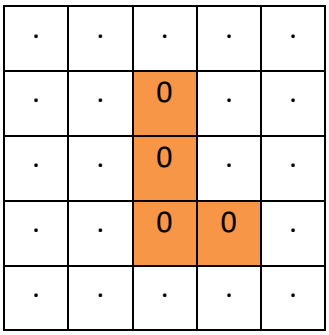
Tétrimino J : forme 4



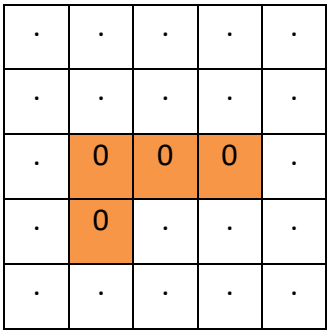
Tétrimino L : forme 1



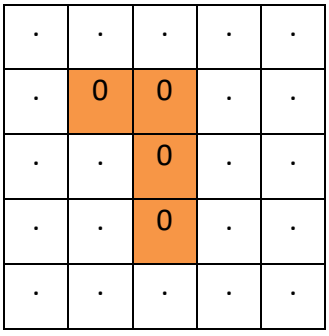
Tétrimino L : forme 2



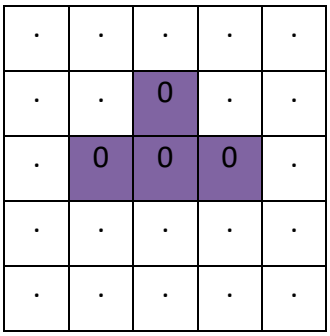
Tétrimino L : forme 3



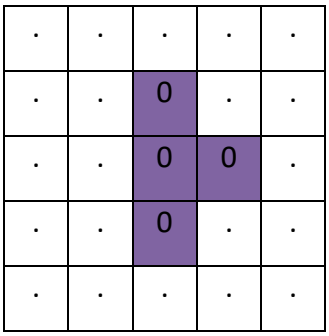
Tétrimino L : forme 4



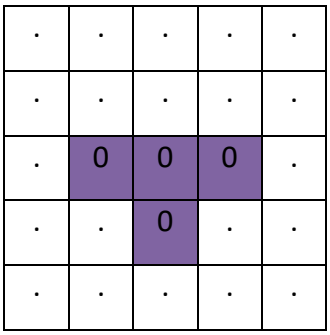
Tétrimino T : forme 1



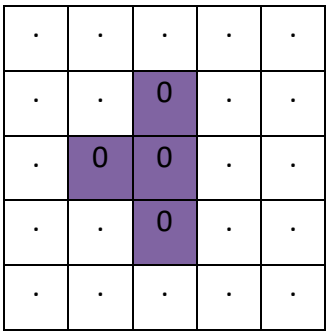
Tétrimino T : forme 2



Tétrimino T : forme 3



Tétrimino T : forme 4



Cela se traduit dans le programme par une liste pour chaque tétrino, comportant des sous-listes, chacune définissant une des formes.

Chaque élément d'une sous-liste définit une ligne de la forme.

```
T = [['.....',  '..0..', '.000.',  '.....', '.....'], ['.....', '..0..', '..00.', '..0..', '.....'],
    ['.....', '.....', '.000.', '..0..', '.....'], ['.....', '..0..', '.00..', '..0..', '.....']]
```

- Placer toutes ces définitions, en dessous des variables globales.
- Pour compléter la définition des tétrinos, nous ajoutons deux listes :

```
#liste des tous les tétrinos

formes = [S, Z, I, O, J, L, T]

#couleur de chaque tétrino, dans l'ordre indiqué par la liste « formes »

couleurs_forme = [(0, 255, 0), (255, 0, 0), (0, 255, 255), (255, 255, 0), (255, 165, 0), (0, 0, 255), (128, 0, 128)]
```

Afficher un tétrino en utilisant sa définition

Nous allons commencer par définir une classe Piece, qui représentera un tétrino à afficher :

```
class Piece(object):
    def __init__(self, x, y, tetrimino):
        self.x = x
        self.y = y
        self.tetrimino = tetrimino
        self.couleur = couleurs_forme[formes.index(tetrimino)]
        self.rotation = 0
```

Les paramètres x, y représentent les coordonnées de départ de la forme.

tetrimino représente le tétrimino à dessiner (S, T, …)

Couleur est la couleur de ce tétrimino

Rotation définit la forme du tétrimino à dessiner : elle peut prendre les valeurs de 0 à 3

La fonction affichant le tétrino suivant, sous le texte « forme suivante »

Définir une constante NOIR = [0,0,0] en début de programme

```
def afficher_formeSuivante(piece, surface):
    sx = top_left_x + largeur_zoneJeu + 50
    sy = top_left_y + hauteur_zoneJeu/2 - 100
   #on efface la forme affichée précédemment
    pygame.draw.rect(surface, NOIR, (sx , sy , 4*taille_bloc, 4*taille_bloc), 0)
    #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):
            #on colorie un petit carré de 30 pixels si colonne = 0
            if colonne == '0':
                pygame.draw.rect(surface, piece.couleur, (sx + j*taille_bloc, sy + i*taille_bloc, taille_bloc, taille_bloc), 0)
```

Comment fonctionne enumerate :

```
my_list = ['apple', 'banana', 'grapes', 'pear']
for c, value in enumerate(my_list):
    print(c, value)
```

Sortie:

0 apple

1 banana

2 grapes

3 pear

Test :

```
#=========================================================
ecran = pygame.display.set_mode((largeur_ecran, hauteur_ecran))
pygame.display.set_caption('Tetris')
pygame.font.init()
afficher_fenetre(ecran)
run = True
rotation = -1
while run:
    rotation+=1
```

```
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False
            pygame.quit()
    if rotation < 4:
        for i in range(7):
            piece_suivante = Piece(5, 0, formes[i])
            piece_suivante.rotation=rotation
            afficher_formeSuivantepiece_suivante, ecran)
            pygame.display.update()
            pygame.time.wait(2000)
    else:
        run = False
        pygame.quit()
```

Nous affichons toutes les formes de tous les tétraminos.