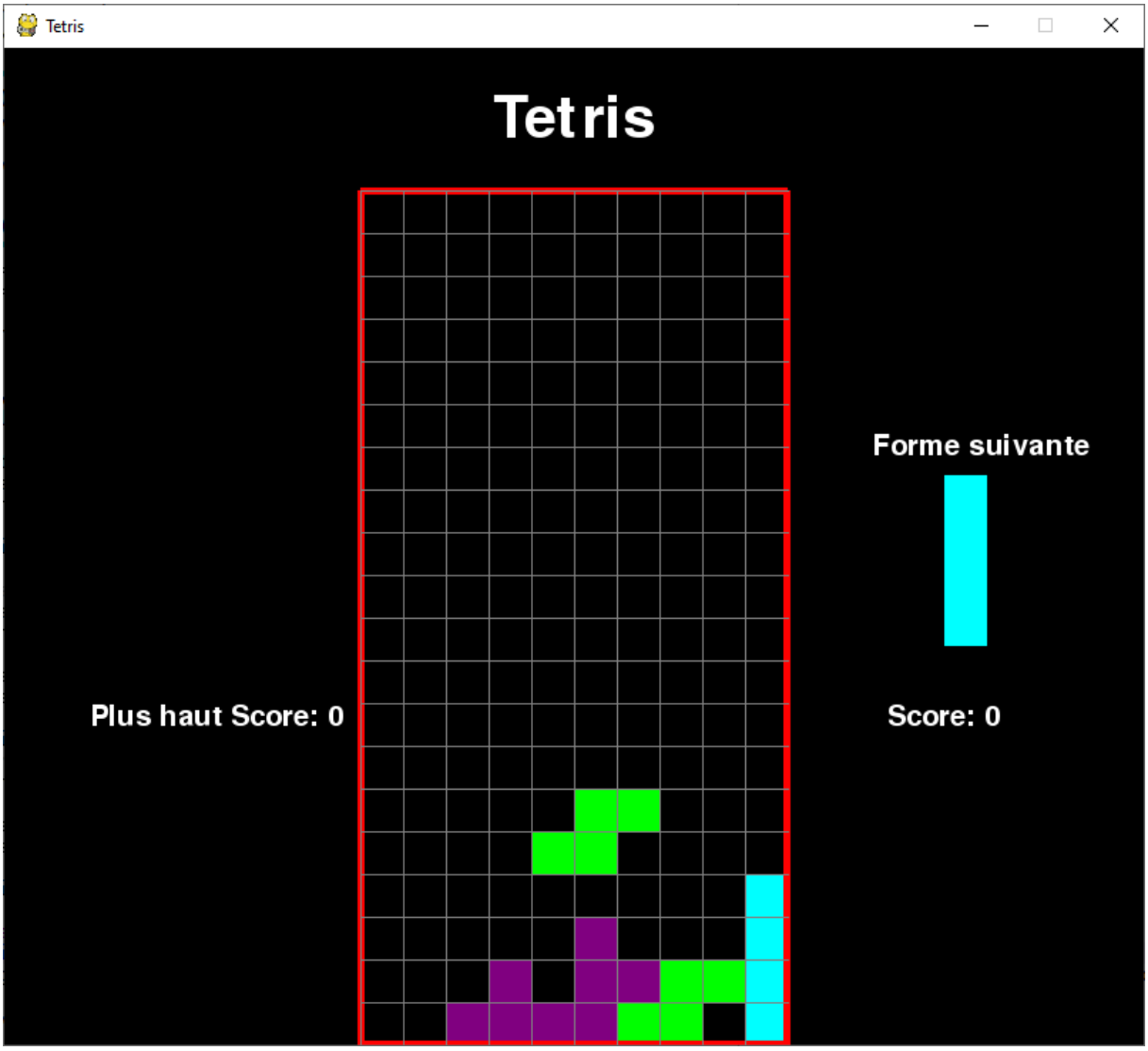
# Tétris 4

## Le jeu




Il ne nous reste plus qu'à fignoler le jeu :

1. Gagner des points lorsqu'on complète une ligne
2. Afficher la pièce suivante à chaque renouvellement de pièce
3. Tester si le joueur a perdu
4. En fin de partie enregistrer le score dans un fichier pour pouvoir le retrouver au démarrage d'une autre partie.
5. Prévoir plusieurs niveaux : la vitesse s'accroit à chaque niveau.

## Gagner des points et afficher la pièce suivante

Nous allons utiliser une variable **piece_suivante** que nous initialisons avant la boucle de jeu ainsi qu'une variable score:

```
#============================================================

ecran = pygame.display.set_mode((largeur_ecran, hauteur_ecran))

pygame.display.set_caption('Tetris')

pygame.font.init()

emplacement_bloque={}

piece_enCours = choix_piece()

piece_suivante = choix_piece()

changer_piece = False

score = 0

run = True

#évènement personnalisé

CHUTE_PIECE = pygame.USEREVENT + 1

#créer un chronomètre qui déclenche l'évènement toute les 2 secondes

pygame.time.set_timer(CHUTE_PIECE, 2000)
```

Le score sera augmenté à chaque fois que l'on remplit une ligne.

A la fin de la boucle de jeux :

```
  #Changer de piece

   if changer_piece:

        #Mémoriser les cases occupées par la piece bloquée

        for pos in position_piece:

           p = (pos[0], pos[1])

           emplacement_bloque[p] = piece_enCours.couleur

        #choisr une autre pièce

        piece_enCours = piece_suivante

        piece_suivante=choix_piece()
```

```
            changer_piece = False
            score+=effacer_ligne(grille, emplacement_bloque)
        #Mettre à jour la fenêtre
        afficher_fenetre(ecran,grille,score)
        #afficher la piece suivante
        afficher_formeSuivante(piece_suivante, ecran)
        pygame.display.update()
```

## Tester si le joueur à perdu

Le joueur perd si la dernière piece bloquée se trouve sur la ligne 0 (en haut).

```
def tester_finPartie(positions):
    for pos in positions:
        x, y = pos
        if y < 1:
            return True
    return False
```

### Test

En fin de boucle de jeu, nous ajoutons :

```
if tester_finPartie(emplacement_bloque):
            print("fin partie")
            pygame.display.update()
            pygame.time.delay(1500)
            run = False
```

## Enregistrer son score et le retrouver en début de partie

Soit max_score une variable qui enregistre le dernier score saugegardé dans un fichier score.txt.

Ce fichier est lu en début de partie :

```
def lire_score():
    with open('scores.txt', 'r') as f:
        lines = f.readlines()
        score = lines[0].strip()
    return score
```

Avant la boucle de jeu on écrit :

```
max_score = lire_score()
```

En fin de partie, on n'enregistre le score obtenu, que s'il est supérieur à max_score

```
def enregistrer_score(nscore, max_score):
    with open('scores.txt', 'w') as f:
        if int(nscore) > int(max_score):
            f.write(str(nscore))
        else:
            f.write(str(max_score))
```

Pour afficher le score max, il faut rajouter ce paramètre à l'affichage de la fenêtre :

**#Mettre à jour la fenêtre**

afficher_fenetre(ecran,grille,score, max_score)

Dès que l'on sort de la boucle de jeu

```
enregistrer_score(score,max_score)
```

pygame.quit()

## Test

Créez un fichier score.txt avec la valeur 0 dans le même dossier que le programme.

Et jouez jusqu'à ce que votre score dépasse 0. Arrêtez le programme et vérifier que le noveau score max a bien été enregistré.

## Page de démarrage et de fin de jeu

Avant de démarrer le jeu, nous affichons une page permettant au joueur de se préparer :

```
#=========================================================
ecran = pygame.display.set_mode((largeur_ecran, hauteur_ecran))
pygame.display.set_caption('Tetris')
pygame.font.init()
#page d'instructions
font = pygame.font.SysFont("comicsansms", 40)
#faut'il encore afficher une page
afficher_instructions = True
# -------- Boucle d'affichage de la première page -----------
while afficher_instructions:
    ecran.fill((0,0,0))
    texte_image = font.render("Cliquez pour démarrer", True, BLANC)
    ecran.blit(texte_image, [150, 200])
    pygame.display.update()
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            afficher_instructions = False
            pygame.quit
        if event.type == pygame.MOUSEBUTTONDOWN:
            afficher_instructions = False
```

En fin de partie :

```
if tester_finPartie(emplacement_bloque):
            ecran.fill((0,0,0))
            texte_image = font.render("Vous avez perdu", True, BLANC)
            ecran.blit(texte_image, [200, 200])
            pygame.display.update()
```

```
pygame.time.delay(5000)

run = False
```

## Implémenter plusieurs niveaux de jeu

Le jeu devient difficile lorsque les pièces tombent vite. Nous allons donc jouer sur la vitesse de chute, pour créer plusieurs niveaux.

La page d'introduction sera maintenant :


Nous avons quatre boutons, qui doivent permettre de définir la vitesse du chronomètre qui gère la chute des pièces. Lorsque la souris passe au dessus de l'un des boutons, sa couleur éclaircit. Lorsqu'on clique sur le bouton, une fonction est appelée qui règle la vitesse du chronomètre et indique le nom du fichier où l'on doit chercher et enregistrer le score maximum obtenu à ce niveau.

Nous définissons de nouvelles variables globales :

```
niveau = 2000
fichier_score ="score_niveau1.txt"
```

Nous définissons de nouvelles constantes de couleurs :

```
BLANC = (255, 255, 255)
VERT = (0,204,0)
ROUGE = (255,0,0)
ROUGE_CLAIR = (254,180,180)
VERT_CLAIR = (33,255,33)
BLEU = (0,0,255)
BLEU_CLAIR=(0,176,240)
VIOLET = (153,0,204)
VIOLET_CLAIR = (232,167,255)
```

La fonction permettant de créer un bouton aura 9 paramètres :

ecran : la surface où il faut écrire

msg : le texte à placer sur le bouton

x, y : coordonnées du point en haut à gauche du bouton

w : largeur du bouton

h : hauteur du bouton

ic : couleur normale du bouton

ac : couleur du bouton lorsqu'il est survolé par la souris

action : fonction à exécuter lorsqu'on clique sur le bouton

```
def button(ecran, msg,x,y,w,h,ic,ac, action=None):
    mouse = pygame.mouse.get_pos()
    click = pygame.mouse.get_pressed()
    #si la souris survole le bouton, on lui met sa couleur claire
    if x+w > mouse[0] > x and y+h > mouse[1] > y:
        pygame.draw.rect(ecran, ac,(x,y,w,h))
```

```
            #Si on a cliqué sur le bouton, on exécute l'action
            if click[0] == 1 and action != None:
                action()
        else:
            #si la souris est en dehors du bouton, on lui met sa couleur normale
            pygame.draw.rect(ecran, ic,(x,y,w,h))
        #Ecriture du texte sur le bouton
        pygame.font.init()
        font = pygame.font.SysFont("comicsansms",20)
        texteSurface = font.render(msg, True, BLANC)
        texteRectangle = texteSurface.get_rect()
        texteRectangle.center = ( (x+(w/2)), (y+(h/2)) )
        ecran.blit(texteSurface, texteRectangle)
```

Les actions possibles : chacune définisse une vitesse et un fichier de score

```
def jouer_niveau1():
    global niveau, fichier_score
    niveau = 1500
    fichier_score = "score_niveau1.txt"

def jouer_niveau2():
    global niveau, fichier_score
    niveau = 1000
    fichier_score = "score_niveau2.txt"

def jouer_niveau3():
    global niveau, fichier_score
    niveau = 500
    fichier_score = "score_niveau3.txt"
```

```
def jouer_niveau4():
    global niveau, fichier_score
    niveau = 250
    fichier_score = "score_niveau4.txt"
```

La fonction gérant l'affichage de la page d'introduction :

```
def intro_jeu(ecran):
   intro = True
   ecran.fill(NOIR)
   while intro:
      for event in pygame.event.get():
         if event.type == pygame.QUIT:
            pygame.quit()
            quit()
         if event.type == pygame.MOUSEBUTTONDOWN:
            #On a cliqué sur un bouton: si on a cliqué en dehors
            #d'un bouton, c'est le niveau 1 qui est lancé
            intro = False
      #affichage du texte
      pygame.font.init()
      font = pygame.font.SysFont("comicsansms", 40)
      texteSurface = font.render("Cliquez sur un bouton de niveau", True, BLANC)
      texteRectangle = texteSurface.get_rect()
      texteRectangle.center = ((largeur_ecran/2),(hauteur_ecran/2)-150)
      ecran.blit(texteSurface, texteRectangle)
      #création des boutons
      button(ecran,"Niveau 1",150,(hauteur_ecran/2),100,50,VERT,VERT_CLAIR,
jouer_niveau1)
```

```
    button(ecran,"Niveau 2",280,(hauteur_ecran/2),100,50,ROUGE,ROUGE_CLAIR,
jouer_niveau2)
    button(ecran,"Niveau 3",410,(hauteur_ecran/2),100,50,BLEU,BLEU_CLAIR,
jouer_niveau3)
    button(ecran,"Niveau 4",540,(hauteur_ecran/2),100,50,VIOLET,VIOLET_CLAIR,
jouer_niveau4)
    pygame.display.update()
```

Les modifications à apporter à la partie principale :

```
#============================================================
ecran = pygame.display.set_mode((largeur_ecran, hauteur_ecran))
pygame.display.set_caption('Tetris')
pygame.font.init()
#page d'instructions
intro_jeu(ecran)
#------------------------------------------------------
emplacement_bloque={}
piece_enCours = choix_piece()
piece_suivante = choix_piece()
etc.
```

Les fonctions de lecture et écriture du score : nous nous assurons dans la fonction de lecture que le fichier de score existe bien. S'il n'existe pas on le crée et on y inscrit 0.

```
def lire_score():
  try:
    with open(fichier_score, 'r') as f:
      lines = f.readlines()
      score = lines[0].strip()
    return score
```

```
    except IOError:
        with open(fichier_score, 'w') as f:
            f.write("0")
            return "0"

def enregistrer_score(nscore, max_score):
    with open(fichier_score, 'w') as f:
        if int(nscore) > int(max_score):
            f.write(str(nscore))
        else:
            f.write(str(max_score))
```

## Avec de la musique :

```
#=========================================================
ecran = pygame.display.set_mode((largeur_ecran, hauteur_ecran))
pygame.display.set_caption('Tetris')
pygame.font.init()
#page d'instructions
intro_jeu(ecran)
pygame.mixer.init()
pygame.mixer.music.load("Tetris.mp3")
pygame.mixer.music.play(-1)
```

**Télécharger et installer : Tetris.mp3 dans le dossier du programme**