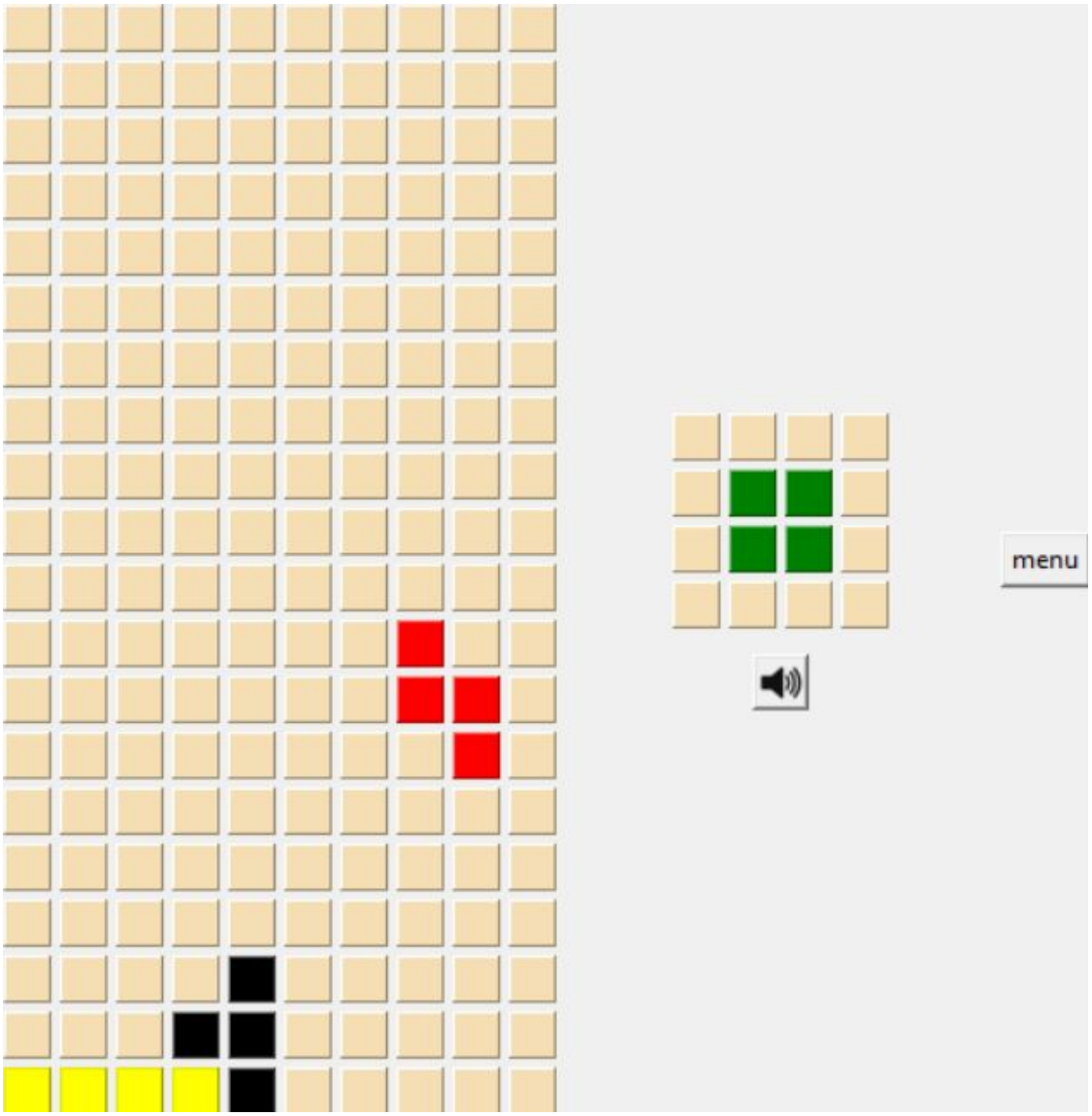
Tétris



DELAS Carla TS2

CLAVEAU Armand TS2

I. Présentation

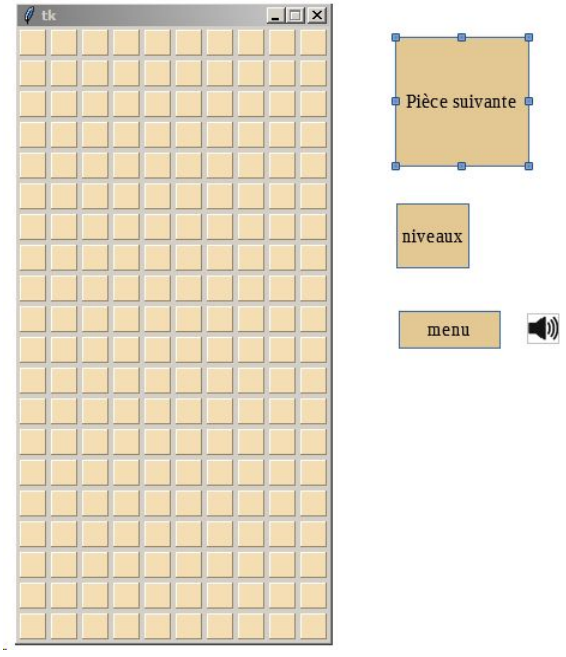
A, Comment rendre un tétris basique plus divertissant ?

Nous voulons coder en python un jeu divertissant et connu de tous. Ce jeu est accessible aux plus jeunes comme aux plus âgées. Ce thème possède de nombreuse possibilité de modifications. Nous pourrions ainsi le modifier à notre image.

Nous avons utilisé le python et le pygame afin de créer notre programme. Pour cela nous nous sommes appuyés sur l'idée d'un tétris basique.

B, objectifs

- Créer différentes pièces
- Accélération des pièces avec la touche descendre
- Visibilité de la pièce suivante
- Augmentation du NIVEAU toutes les 10 lignes remplies
    - accélération de la vitesse
- Musique
- Couleurs différentes pour chaques pièces
- Notice
- échanger la pièce actuelle avec la pièce suivante



C, Répartition des tâches

date	Carla	Armand
04/03/2019	Aperçu du projet Mise en place des objectifs	
11/03/2019	Commencer la rédaction du dossier-projet, Initiation à pygame	
18/03/2019	Musique	Niveau, augmentation vitesse,
25/03/2019	Couleur piece	Arrêt et départ musique avec pygame
01/04/2019	Couleur piece	musique et image associé au bouton
08/04/2019	Fin Couleur	
29/04/2019	notice	interface du menu
06/05/2019	bouton notice	affichage pièce suivante
13/05/2019	correction de bug d'affichage	

II. Réalisation

Avant de commencer à coder, nous avons regardé en détail le code déjà existant pour le comprendre et se l'approprier. Puis nous avons établit les objectifs.

A. Ma première partie

Dans ma première partie, je devais ajouter des niveaux pour augmenter la vitesse de descente des pièces, à chaque niveau. J'ai d'abord gérer la descente de la pièce à la pression de la touche "flèche du bas". Pour cela j'ai d'abord voulu ajouter des unités à la vitesse de descente. Et j'ai finis par comprendre que ce serait plus pratique s'il y avait une vitesse de descente rapide et une autre de descente normale, celle au long de la partie sans utiliser la touche "flèche du bas". il suffisait alors, quand la flèche du bas est enfoncée, de changer la vitesse de descente en celle de descente rapide, et quand le bouton est relâché de changer la vitesse du jeu en une vitesse courante.

Ensuite j'ai plus qu'as conté le nombre de lignes qui sont complété, et toutes les 10 lignes j'ajoute une unité à une variable, qui correspond au niveau, et rajouter à la vitesse courante quelques unité. (Le fait de compter le niveau n'est pas utile pour le moment mais pourrait être affiché)

B. Ma deuxième partie

Cette partie a était la plus compliqué à réaliser. Je devais créer un bouton qui gère la musique, et sur ce bouton insérer une image de son ouvert ou fermé. On a donc utilisé pygame pour la musique. Il a d'abord fallu s'habituer à pygame, puis à l'insérer dans un code contenant Tkinter. J'ai donc créé un bouton avec Tkinter puis je lui ai donné la commande pour lancer une fonction qui teste si la musique est allumé puis l'éteint et l'inverse. Ensuite les difficultés sont arrivées quand j'ai essayé de mettre l'image sur le bouton. Je n'arrivais pas à faire changer l'image sur le bouton. Une fois le programme lancé, l'image sur le bouton apparaissait puis disparaissait et le bouton se bloquait. Après quelques changements l'erreur a disparu. Mais lors de l'insertion de cette partie du code dans le code complet l'erreur est réapparu. On a donc cherché avec Carla mais l'aide du professeur a était essentiel. Finalement le bouton n'était pas initialisé au bon endroit.

C. Ma troisième partie

Ma troisième partie consiste à créer un espace à droite de la zone de jeux pour insérer un menu, ajouter le bouton du son et afficher la pièce suivante. J'ai donc créé un canvas que j'ai positionné à droite de la grille de jeu. J'ai ensuite ajouté le bouton du son en bas de ce canvas. Après, j'ai copier une partie du code déjà existant pour créer 16 canvas dans un canvas plus grand pour y placer la pièce suivante du jeu. J'ai donc créé une deuxième grille mais cette fois de 4 sur 4. j'ai ensuite recopié les fonctions qui permettent :

- d'afficher la pièce dans la grille
- d'effacer la pièce de la grille
- d'actualiser la grille

puis j'ai modifié les tuples pour qu'ils correspondent à la grille menu. J'ai ensuite fait afficher la pièce au moment de faire apparaître celle dans la grille de jeu et effacer juste avant. La grille s'actualise à chaque fois que l'on efface ou affiche une pièce.

Dans la grille d'affichage de la pièce suivante, il y avait une erreur d'affichage, la piece était souvent à moitié effacé. J'ai donc résolu ce problème en modifiant la fonction effacée. Au lieu de remettre à 0 seulement les cases qui correspondent à la pièce, je remet toute la grille a 0.

Une fois cette étape terminée, j'ai créé un ordre de sélection des pièces pour connaitre la pièce suivante. J'ai d'abord créé une liste pour placer la pièce en jeu et celle au rang +1. Mais j'ai trouvé plus simple de créer des tuples correspondant à la pièce en jeu et celle d'après. Puis lors de la sélection de la nouvelle pièce je décale la pièce du rang +1 à celle en jeux et je choisi aléatoirement dans la liste des pièces possibles la pièce du rang +1.

D. Ma dernière partie

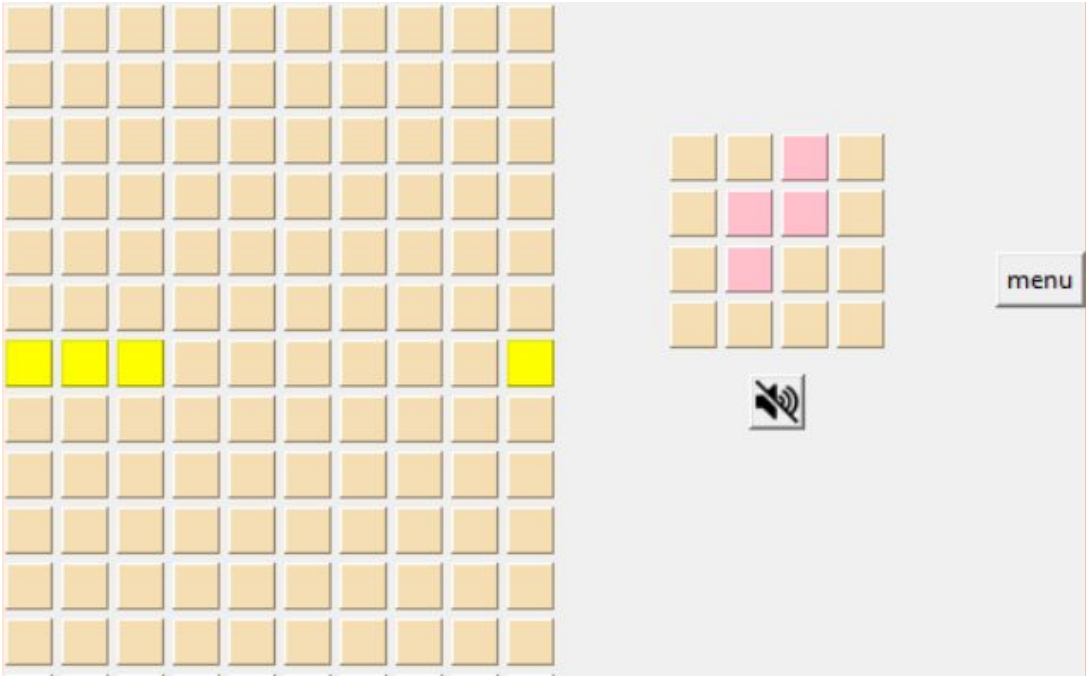
Pour cette dernière partie, je devais faire changer la piece en jeu avec celle au rang +1, affiché dans le menu. J'ai donc créé une fonction qui se lance quand on appui sur la bar d'espace. Cette fonction échange les tuples définie précédemment comme la pièce en jeu et la pièce au rang +1. Pour cela je me sers d'un tuple de stockage, que j'ai nommé pièce au rang +2. J'ai ensuite effacé les deux pièces de leur grille, pour enfin les afficher.

Le changement de pièce fonctionnait mais si la pièce était trop près du bord et que l'autre déborde au moment du changement, le programme plante. J'ai donc eu l'idée de créer une fonction qui vérifie si la pièce au rang +1 à la place de la pièce en jeu est dans le cadre. Si cette fonction n'est pas vraie alors le changement ne se fait pas.

III. Bilan

Le code de chacun était mis en commun à toute les séances, pour que l'on puisse se rendre compte des erreurs et de l'avancement.

Le résultats est un jeu jouable, mais il reste quelques erreur que nous n'avons pas eu le temps de corriger, par exemple lors du changement de pièce, la pièce de 4 blocs de long, si elle est couchée, apparaît à l'extérieur de la grille de jeu mais la fonction ne la détecte pas et le jeu ne plante pas. Elle se pose la moitié d'un côté et l'autre de l'autre de la grille de jeu. Comme sur la capture d'écran suivante.



Aucune autre erreur n'as était trouvée pour le moment.

Pour améliorer le jeu, il serait plutôt bien d'afficher le niveau auquel on est, et de faire une interface de lancement et de fin du jeu. On pourrait aussi faire des statistiques des parties.

Le fait de travailler en groupe m'as obligé à expliquer clairement ce que je voulais et ce que je faisait, pour aider mais aussi pour ne pas faire des parties de code incompatible. J'était aussi surement plus motivé car le travaille que j'apportais pouvais pénalisé le groupe si je ne le faisait pas correctement.

IV. Annexe:

Pour que notre projet fonctionne il faut un fichier contenant le son du jeu, les images correspondantes au son et celle correspondante à la notice ; voici un lien pour télécharger le fichier contenant le code et les images et le son, plus le code de base du quel nous somme parti : https://drive.google.com/drive/folders/1Vt4GYjHnkgD2jqXVKdHCDf26trMnVHUB?usp=sharing

code :

```
from tkinter import *
from time import sleep
from random import *
import pygame


def defpiece():
	global piece,tab_couleur,piecen,piecenn,Piecen1,Piecen2

tab_couleur=['wheat','red','green','yellow','pink','black','#FFFFFF','#FFA500','#90
EE90']


	piece1 = [[(0,1,0,0),(0,1,1,0),(0,0,1,0),(0,0,0,0)], # 0 1 0 0
		  [(0,0,0,0),(0,0,1,1),(0,1,1,0),(0,0,0,0)],     # 0 1 1 0
		  [(0,1,0,0),(0,1,1,0),(0,0,1,0),(0,0,0,0)],     # 0 0 1 0
		  [(0,0,0,0),(0,0,1,1),(0,1,1,0),(0,0,0,0)]]     # 0 0 0 0

	piece2 = [[(0,0,0,0),(0,2,2,0),(0,2,2,0),(0,0,0,0)], # 0 0 0 0
		  [(0,0,0,0),(0,2,2,0),(0,2,2,0),(0,0,0,0)],     # 0 2 2 0
		  [(0,0,0,0),(0,2,2,0),(0,2,2,0),(0,0,0,0)],     # 0 2 2 0
		  [(0,0,0,0),(0,2,2,0),(0,2,2,0),(0,0,0,0)]]     # 0 0 0 0

	piece3 = [[(0,3,0,0),(0,3,0,0),(0,3,0,0),(0,3,0,0)], # 0 3 0 0
		  [(0,0,0,0),(3,3,3,3),(0,0,0,0),(0,0,0,0)],     # 0 3 0 0
		  [(0,3,0,0),(0,3,0,0),(0,3,0,0),(0,3,0,0)],     # 0 3 0 0
		  [(3,3,3,3),(0,0,0,0),(0,0,0,0),(0,0,0,0)]]     # 0 3 0 0

	piece4 = [[(0,0,4,0),(0,4,4,0),(0,4,0,0),(0,0,0,0)], # 0 0 4 0
		  [(0,0,0,0),(0,4,4,0),(0,0,4,4),(0,0,0,0)],     # 0 4 4 0
		  [(0,0,4,0),(0,4,4,0),(0,4,0,0),(0,0,0,0)],     # 0 4 0 0
```

```
                        [(0,0,0,0),(0,4,4,0),(0,0,4,4),(0,0,0,0)]]     # 0 0 0 0

        piece5 = [[(0,5,0,0),(0,5,5,0),(0,5,0,0),(0,0,0,0)], # 0 5 0 0
                        [(0,0,0,0),(0,0,5,0),(0,5,5,5),(0,0,0,0)],     # 0 5 5 0
                        [(0,0,0,5),(0,0,5,5),(0,0,0,5),(0,0,0,0)],     # 0 5 0 0
                        [(0,5,5,5),(0,0,5,0),(0,0,0,0),(0,0,0,0)]]     # 0 0 0 0

        piece6 = [[(0,0,6,0),(0,0,6,0),(0,6,6,0),(0,0,0,0)], # 0 0 6 0
                        [(0,0,0,0),(0,6,0,0),(0,6,6,6),(0,0,0,0)],     # 0 0 6 0
                        [(0,6,6,0),(0,6,0,0),(0,6,0,0),(0,0,0,0)],     # 0 6 6 0
                        [(0,0,0,0),(0,6,6,6),(0,0,0,6),(0,0,0,0)],] # 0 0 0 0

        piece7 = [[(0,8,0,0),(0,8,0,0),(0,8,8,0),(0,0,0,0)], # 0 8 0 0
                        [(0,0,0,0),(0,8,8,8),(0,8,0,0),(0,0,0,0)],     # 0 8 0 0
                        [(0,8,8,0),(0,0,8,0),(0,0,8,0),(0,0,0,0)],     # 0 8 8 0
                        [(0,0,0,0),(0,0,0,8),(0,8,8,8),(0,0,0,0)]]     # 0 0 0 0

#       piece = [[(1,1,1,1),(1,1,1,1),(0,0,0,0),(0,0,0,0)],    # 1 1 1 1
#                       [(1,1,0,0),(1,1,0,0),(1,1,0,0),(1,1,0,0)],     # 1 1 1 1
#                       [(1,1,1,1),(1,1,1,1),(1,1,1,1),(1,1,1,1)],     # 1 1 1 1
#                       [(1,1,1,1),(1,1,1,1),(1,1,1,1),(1,1,1,1)]]     # 1 1 1 1

        ListePiece = [piece1,piece2,piece3,piece4,piece5,piece6,piece7]

        try:
                if piece == 0:
                        piece = 0
                else:
                        piece = Piecen1
        except NameError:
                piece = choice(ListePiece)
#si piece n'est pas définit choisi une piece au hasard sinon transforme la piece en
la suivante

        Piecen1 = choice(ListePiece)
#choisi une piece au hasard pour n+1



#-------------------------------------------------------------------------------
def initialisation():
        """ Creation du tableau de canevas 20 lignes 10 colonnes et de sa matrice
                associee.Definition d'une piece et de ses 2 positions."""
        global
tab_can,tab_can2,tab_grille,tab_grille2,coord_courante,vitesse,vitesse_courante,nbr
_ligne,niveau,test_notice,button,photo0,photo1,testmusique

        nbr_ligne = 0
#mise à 0 du compteur de ligne complété
        niveau = 1
#mise à 1 du niveau
```

```
	vitesse_courante = 300
	vitesse = vitesse_courante
#vitesse de descente de la pièce
	defpiece()

	test_notice = 0
#indicateur de la fenêtre notice
	tab_can=[]
	tab_grille=[]
	for i in range(20):
		tab_can.append([[]]*10)
		tab_grille.append([[]]*10)

	grille_jeu = Frame(fen)
	for ligne in range(20):
		for colonne in range(10):
			couleur = tab_couleur[0]
			tab_can[ligne][colonne] =
Canvas(grille_jeu,bg=couleur,height=20,

width=20,borderwidth=1,relief=RIDGE)
			tab_can[ligne][colonne].grid(row=ligne,column=colonne)
			tab_grille[ligne][colonne] = 0
	grille_jeu.grid(row=0,column=0)

#création d'une grille de 10 cases sur 20 qui correspond à l'espace de jeu.

#musique
	pygame.mixer.init()
	music = pygame.mixer.music.load("musique.ogg")
	pygame.mixer.music.play()
	pygame.mixer.music.set_volume(0.1)
#lance la musique , définit son volume
	testmusique = 1
#définit la variable musique comme allumée : 1

	photo0 = PhotoImage(file="soundoff.png")
	photo1 = PhotoImage(file="soundon.png")
#défini les image musique éteinte et allumé qui s'afficheron sur le bouton



	tab_can2=[]
	tab_grille2=[]
	for i in range(4):
		tab_can2.append([[]]*4)
		tab_grille2.append([[]]*4)

	grille_menu = Canvas(Menu)
	for ligne in range(4):
		for colonne in range(4):
```

```
                    couleur = tab_couleur[0]
                    tab_can2[ligne][colonne] =
Canvas(grille_menu,bg=couleur,height=20,

width=20,borderwidth=1,relief=RIDGE)
                    tab_can2[ligne][colonne].grid(row=ligne,column=colonne)
                    tab_grille2[ligne][colonne] = 0
      grille_menu.pack(side=TOP, padx=50, pady=5)

#création d'une grille de quatres cases sur quatres qui correspond à l'affichage de
la pièce suivante.

      button = Button(Menu, image=photo1,command = musique)
      button.pack(side=TOP, padx=5, pady=5)
#crée le bouton, lui assigne l'image allumé et la commande pour gérer la musique



      fen.bind('<Right>',droite)
      fen.bind('<Left>',gauche)
      fen.bind('<Up>',tourne)
      fen.bind('<Down>',vite)
      fen.bind('<KeyRelease-Down>',vite_relache)
      fen.bind('<space>',piece_change)
#associe les actions au touches du clavier

      maj_Grille()
      nouvelle_Piece()

#-------------------------------------------------------------------------------
def nouvelle_Piece():
      """ C'est par le biais de coord_courante que l'on va se deplacer
            dans la matrice soit horizontalement (gauche:dx=-1,droite:dx=1)
            soit verticalement (descente:dy=1).Sens represente la position de la
            piece dans le tableau contenant la piece courante."""
      global coord_courante,sens

      defpiece()
      coord_courante = [0,3]
      sens = 0
      sleep(1)
      if not verif_Deplacement(0,0,0):
            imprime_Piece(sens)
            print ("Perdu")
            fen.destroy()
      else:
            descente()
            imprime_PieceMenu()

#-------------------------------------------------------------------------------
def descente():
```

```
	"""On efface la piece et on teste si elle peut descendre toutes les 250ms.
	Dans le cas contraire on la reinscrit a sa position courante et on verifie
	si une ou plusieurs lignes se sont formees."""
	global vitesse

	efface_Piece()
	if verif_Deplacement(1,0,0):
		coord_courante[0] += 1
		imprime_Piece(sens)
		fen.after(vitesse,descente)
	else :
		imprime_Piece(sens)
		efface_PieceMenu()
		verif_Ligne()
		nouvelle_Piece()
#---------------------------------------------------------------------------
def vite(event):

	global vitesse

	vitesse = 10

def vite_relache(event):

	global vitesse,vitesse_courante

	vitesse = vitesse_courante

#---------------------------------------------------------------------------
def gauche(event):
	"""On efface la piece et on verifie si elle peut se deplacer vers la gauche.
	Dans le cas contraire on la reinscrit a sa position courante."""
	efface_Piece()
	if verif_Deplacement(0,-1,0):
		coord_courante[1] -=1
		imprime_Piece(sens)
	else:
		imprime_Piece(sens)

#---------------------------------------------------------------------------
def droite(event):
	"""On efface la piece et on verifie si elle peut se deplacer vers la droite.
	Dans le cas contraire on la reinscrit a sa position courante."""
	efface_Piece()
	if verif_Deplacement(0,1,0):
		coord_courante[1] += 1
		imprime_Piece(sens)
	else:
		imprime_Piece(sens)

#---------------------------------------------------------------------------
```

```
def tourne(event):
	"""On efface la piece et on verifie si on peut la faire tourner.
	Dans le cas contraire on la reinscrit a sa position courante."""
	global sens

	efface_Piece()
	if verif_Deplacement(0,0,1):
		sens += 1
		if sens == 4:
			sens = 0
		imprime_Piece(sens)
	else:
		imprime_Piece(sens)

#---------------------------------------------------------------------------
def efface_Piece():
	""" Avant n'importe lequel des deplacements on commence par effacer la piece
		de la matrice. Pour ceci on se place dans la matrice a la
coord_courante,
		et on boucle en ligne et en colonne dans la piece.(coord_courante
		represente le coin superieur gauche de la piece).Si la valeur de la
		piece dans une colonne donnee vaut zero on passe a la colonne suivante
		pour ne pas effacer les blocs preexistants.
		Dans le cas contraire son emplacement dans la matrice est mis a
zero."""
	for i in range(4):
		for j in range(4):
			if piece[sens][i][j] == 0:
				continue
			tab_grille[coord_courante[0]+i][coord_courante[1]+j] = 0

#---------------------------------------------------------------------------
def imprime_Piece(sens):
	"""Meme principe que efface_Piece(), cependant ici on passe les coordonnees
	   dans la matrice a la valeur de la piece."""
	for i in range(4):
		for j in range(4):
			if piece[sens][i][j] == 0:
				continue
			tab_grille[coord_courante[0]+i][coord_courante[1]+j] =
piece[sens][i][j]
	maj_Grille()

#---------------------------------------------------------------------------
def imprime_PieceMenu():

	for i in range(4):
		for j in range(4):
			if Piecen1[0][i][j] == 0:
				tab_grille2[i][j] = 0
			else:
```

```
                        tab_grille2[i][j] = Piecen1[0][i][j]
      maj_Grille2()
#--------------------------------------------------------------------------------
def piece_change(event):
      global Piecen1,Piecen2,piece,sens

      if verif_Changement():
            efface_Piece()
            efface_PieceMenu()
#efface la pice dans la grille de jeu et dans la grille de menu
            Piecen2 = Piecen1
            Piecen1 = piece
            piece = Piecen2
#échange la piece du jeu avec la suivante
            imprime_Piece(sens)
            imprime_PieceMenu()
#           print('possible')
#réaffiche la pièce suivante dans la grille de Menu
      else:
            print('impossbile')
#--------------------------------------------------------------------------------


def efface_PieceMenu():

      for i in range(4):
            for j in range(4):
                  tab_grille2[i][j] = 0
#--------------------------------------------------------------------------------
def verif_Deplacement(dy,dx,pivot):
      """La piece a ete effacee, on cherche maintenant a savoir si on peut la
         reinscrire vers la droite, vers la  gauche, vers le bas ou la faire
         tourner.Prenons l'exemple d'une translation vers la droite.
         On cherche d'abord a savoir si la piece ne sera pas en dehors de la
         matrice.Pour cela on se place a la coord_courante dans la matrice + le
         decalage(dx=1),et on boucle en ligne et en colonne dans la piece.
         Pour une ligne donnee,lorsque la coord_courante+dx+incrementation en
         colonne dans la piece depassent la limite de la matrice en largeur, on
         verifie la valeur de la colonne correspondante a cette incrementation
         dans la piece.Si elle vaut zero on poursuit l'incrementation, dans le cas
         contraire on renvoie un deplacement non possible a la methode appelante.
         Si la piece ne "deborde" pas, on multiplie la valeur de la coordonnee de
         la piece par celles de la matrice a la coordonnee correspondante d'apres
         la incrementations effectuees.Si le produit est non nul, un bloc est deja
         present donc on renvoie un deplacement non possible."""
      rotation = sens + pivot
      if rotation == 4 :
            rotation = 0
      for i in range(4):
            for j in range(4):
```

```
                    if coord_courante[1]+(dx+j) > 9 or coord_courante[1]+(dx+j) < 0
or coord_courante[0]+(dy+i) > 19:
                            if piece[rotation][i][j] != 0:
                                    return False
                            else:
                                    continue
                    if (piece[rotation][i][j] *
tab_grille[coord_courante[0]+(dy+i)] [coord_courante[1]+(j+dx)]) != 0:
                            return False
      return True

#---------------------------------------------------------------------------

def verif_Changement():
      for i in range(4):
            for j in range(4):
                    if coord_courante[1]+(j) > 9 or coord_courante[1]+(j) < 0 or
coord_courante[0]+(i) > 19:
                            if Piecen1[sens][i][j] != 0:
                                    #print('a')
                                    print(Piecen1)
                                    return False
                            else:
                                    #print('aa')
                                    return True
                    else:
                            #print('aaa')
                            return True

def verif_Ligne():

      global nbr_ligne,niveau,vitesse_courante,vitesse

      """Appelee apres chaque pose de piece, cette methode verifie par recursivite
      si au moins une ligne a ete construite.On se deplace ligne par ligne de par
      le bas dans la matrice.Par defaut la ligne est consideree complete.Pour
      chaque ligne on verifie chaque colonnes. Si la valeur de la matrice a cette
      coordonnee vaut zero la ligne n'est pas complete et on passe a la ligne
      suivante. Lorsqu'une ligne est complete on la supprime, on rajoute une ligne
      de "zero" en premier indice et on refait une verification de par le bas."""
      for ligne in range(19,-1,-1):
            ligne_complete = True
            for colonne in range(10):
                    if tab_grille[ligne][colonne] == 0:
                            ligne_complete = False

            if ligne_complete:
                    del tab_grille[ligne]
                    tab_grille[0:0] = [[0,0,0,0,0,0,0,0,0,0]]
                    nbr_ligne = int(nbr_ligne) + 1
                    print (nbr_ligne)
```

```
#             compte le nombre de ligne compléter
#             si > 9 remet le compteur à 0 et augmente le niveau et la vitesse
#             de descente des pieces

                  if nbr_ligne > 10 :
#                       print ("niveau up")
                        niveau = niveau + 1
                        vitesse_courante = vitesse_courante - 25
                        vitesse = vitesse_courante
                        print (vitesse)
                        nbr_ligne = 0


                  maj_Grille()
                  verif_Ligne()

#-------------------------------------------------------------------------------
def maj_Grille():
      """Mise a jour des canevas en fonction des couleurs referencees par les
      valeurs de la matrice."""
      for ligne in range(20):
            for colonne in range(10):
                  couleur = tab_couleur[tab_grille[ligne][colonne]]
                  tab_can[ligne][colonne].configure(bg=couleur)
                  #séléctione une case, prend sa valuer et définit sa couleur en
fonction.

      fen.update()

#-------------------------------------------------------------------------------
def maj_Grille2():

      for ligne in range(4):
            for colonne in range(4):
                  couleur = tab_couleur[tab_grille2[ligne][colonne]]
                  tab_can2[ligne][colonne].configure(bg=couleur)
                  #séléctione une case, prend sa valuer et définit sa couleur en
fonction.

      fen.update()

#-------------------------------------------------------------------------------
def musique():

      global testmusique

      if testmusique == 1:
            testmusique = 0
            #Test si la variable musique est sur 1, si elle l'est, la met sur 0
            pygame.mixer.music.pause()
```

```
                #met la musique en pause
                button.config(image=photo0)
                #change l'image du bouton musique sur l'image éteinte
                #print("stop")


        elif testmusique == 0:
                testmusique = 1
                #Test si la variable musique est sur 0, si elle l'est, la met sur 1
                pygame.mixer.music.unpause()
                #relance la musique
                button.config(image=photo1)
                #change l'image du bouton musique sur l'image allumée
                #print("unpause")
#-------------------------------------------------------------------------------
def nouvellefen ():
        global test_notice,fen1
        if test_notice == 0:
                fen1 = Toplevel()
                fen1.title("Notice")
                img_notice = PhotoImage(file='menuimage.png')
                label = Label(fen1,image=img_notice)
                label.image = img_notice
                label.grid()
                test_notice = 1


        else :
                fen1.destroy()
                test_notice = 0


fen = Tk()
fen.wm_geometry(newGeometry='+220+0')
fen.resizable(0,0)

Menu = Canvas(fen,width=200,height=500)
Menu.grid(row=0,column=1)

button2 = Button(fen, text='menu', command = nouvellefen)
button2.grid(row=0,column=2)




initialisation()
fen.mainloop()
```