



RAPPORT FINAL DU PROJET SUR :

Jeu TETRIS

Réalisé par :
Katia ALLACHE
M'balou SACKO

Encadré par :
M. AMIR NAKIB

Promotion 2021 - 2022

Table des matières

Table des matière	1
Liste des Figures	2
Liste des Tableaux	2
1 Introduction :	3
2 Principe du jeu :	3
3 Réalisation du jeu :	4
3.1 La morphologie des Tetrominos:	4
3.2 Création du décor:	6
3.3 Création des pièces:	7
3.4 Déplacement de la pièce:	9
4 Gestion de la rotation, déplacements latéraux et la descente pour les tetrominos :	9
4.1 Gestion de la rotation:	9
4.2 Déplacements latéraux:	10
4.3 Gestion de la descente:	11
5 Rendu final :	12
6 Conclusion :	13
Références	14

Table des figures

1	Image de l'écran d'un jeu TETRIS. [1]	3
2	Schéma des Tetrominos. [2]	4
3	Le décor de TETRIS	7
4	Les pièces existantes du jeu Tetris	8
5	Illustration de déplacement d'une pièce	9
6	Illustration d'une rotation d'un Tetrominos. [3]	10
7	Version final du jeu TETRIS	13

Liste des tableaux

1	Liste des Tetrominos existants.	5
---	---------------------------------	---

1 Introduction :

Nous devons réaliser un programme pour un projet dans le cadre de la formation en Conception et Analyse des Algorithmes (CAA), le projet était libre. Nous avons codé ces projets par groupes de deux . On utilise le langage de programmation Python que nous avons appris à utiliser au cours de l'année.

Pour écrire le code nous nous servons de l'anaconda, et aussi également la bibliothèque Pygame qui permet une meilleure gestion de l'interface graphique.

Notre groupe est composé de SACKO M'Balou et ALLACHE Katia . Et le thème de notre projet est de réaliser le jeu très connu Tetris.

Tetris est un jeu vidéo entre arcade et puzzle , conçu par Alekseï Pajitnov en juin 1984. Le succès devient planétaire et toutes les consoles qui suivront posséderont leur version de Tetris. Il fait partie des jeux les plus addictifs de l'époque avec Pacman. Ce jeu consiste à compléter des lignes de carrés en plaçant des pièces tombantes composées chacune de quatre carrés afin de réaliser le plus haut score possible [4].

2 Principe du jeu :

Le jeu consiste, grâce à seulement quatre touches, à orienter une pièce à droite ou à gauche, pendant que cette dernière tombe vers le bas de la grille. On peut aussi la faire tourner sur elle-même ou accélérer sa descente. Une fois la pièce bloquée, on ne peut plus modifier sa position et une autre pièce arrive en haut du décor. Et au grand désarroi du joueur, ce n'est jamais celle qu'il attend ! Ses ennemis sont les trous qu'il laisse au fur et à mesure de ses empilements maladroits.

Cependant, lorsqu'une ligne est complète, elle disparaît et les lignes supérieures descendent alors d'un étage, laissant au joueur un espoir de survie supplémentaire ! Le jeu s'arrête lorsqu'une nouvelle pièce est bloquée dès son arrivée [5].

La figure 1 représente l'image de l'écran du jeu TETRIS :

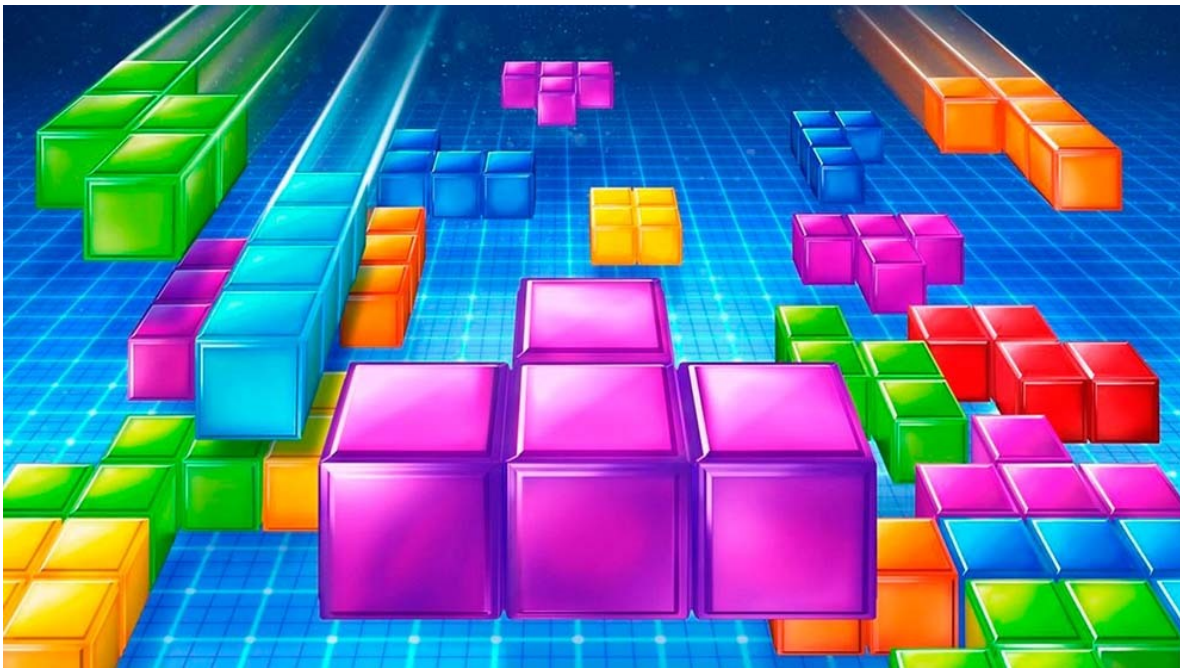


FIG. 1 – Image de l'écran d'un jeu TETRIS. [1]

En se servant de ces liens [5, 6, 7], on a pu avoir plus de détails dans les parties suivantes (Sec.3 et Sec.4).

3 Réalisation du jeu :

3.1 La morphologie des Tetrominos :

Tout d'abord, nous avons déclaré des constantes pour les couleurs et nous les avons stockées dans une liste de couleurs. Le fond noir est associé à la valeur 0, et murs gris à la valeur 1.

Le code suivant permet de définir les couleurs :

```
NOIR = (0, 0, 0)
VERT = (0, 255, 0)
ROUGE = (255, 0, 0)
BLEU = (0, 0, 255)
GRIS = (128,128,128)
CYAN = (0,255,255)
JAUNE = (255,255,0)
ORANGE= (255,150,0)
VERT = (0,255,255)
MAUVE = (180,80,255)
LCou = [ NOIR, GRIS, CYAN, JAUNE, MAUVE, ORANGE, BLEU,
ROUGE, VERT ]
```

Il existe 7 tétraminoes (pièces) dans ce jeu, c'est-à-dire, tous les groupes que l'on peut créer à partir de quatre carrés. Pour simplifier nos algorithmes, nous allons utiliser les combinaisons de carrés qui s'inscrivent dans une grille 3x3. Ainsi, nous aurons les pièces suivantes dans notre Jeu. La figure 2 représente le schéma des Tetrominos.

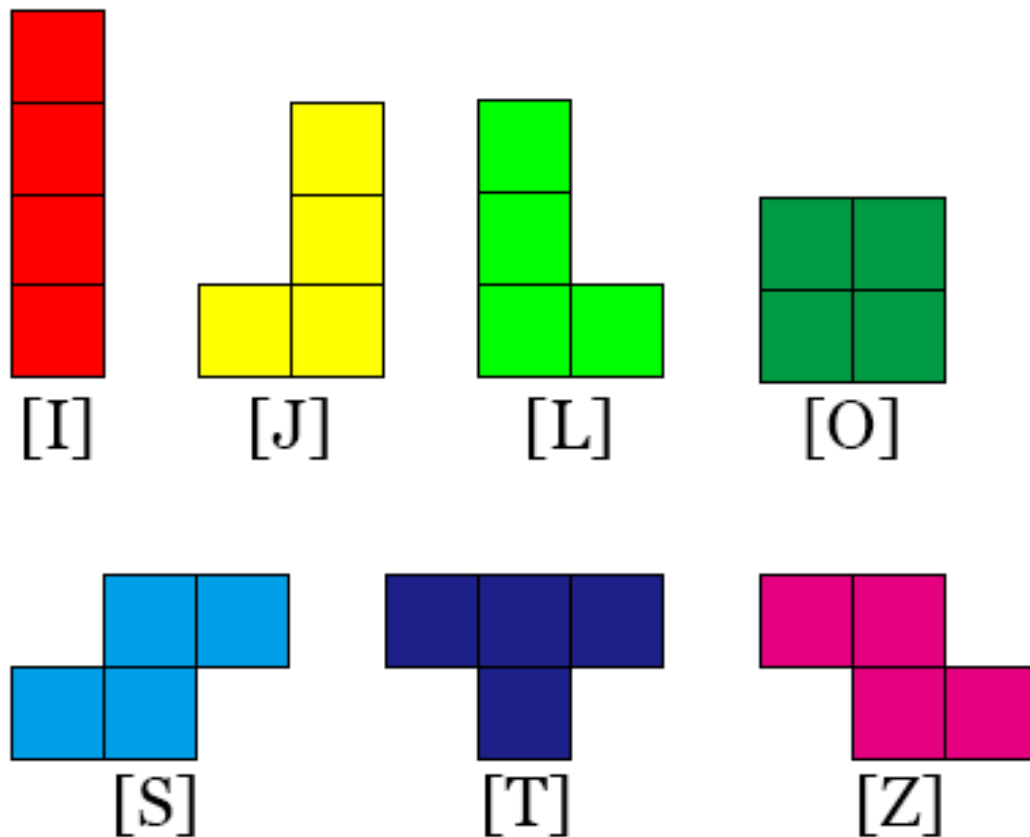

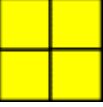
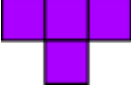

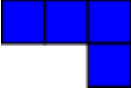
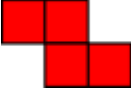
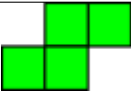


FIG. 2 – Schéma des Tetrominos. [2]

Le tableau (Table 1) représente la liste des pièces existantes dans un jeu de TETRIS.

Forme	Appellation	Autres Appellations	Construction
	Tétrimino I	<i>Bâton</i> , <i>ordinaire</i> , <i>barre</i> , <i>long</i>	Quatre carrés alignés
	Tétrimino O	« Carré », « bloc »	Méta-carré de 2x2
	Tétrimino T	« Té »	Trois carrés en ligne et un carré sous le centre
	Tétrimino L	« L » , « lambda »	Trois carrés en ligne et un carré sous le côté gauche.
	Tétrimino J	« L inversé », « gamma »	Trois carrés en ligne et un carré sous le côté droit.
	Tétrimino Z	« Biais »	TMéta-carré de 2x2, dont la rangée supérieure est glissée d'un pas vers la gauche.
	Tétrimino S	« Biais inversé »	Méta-carré de 2x2, dont la rangée supérieure est glissée d'un pas vers la droite.

TAB. 1 – *Liste des Tetrominos existants.*

Ensuite, toutes les tetrominos ont été stockées dans des listes de listes de 3x3 éléments. Une valeur nulle correspond à une case vide et une valeur non nulle indique une case pleine ainsi que sa couleur. L'ensemble des pièces est stocké dans une liste nommée LTetros.

Le code suivant correspond aux pièces existantes :

```

TetrosI = [[0,2,0],[0,2,0],[0,2,0]]
TetrosT = [[0,0,0],[4,4,4],[0,4,0]]
TetrosO = [[3,3,0],[3,3,0],[0,0,0]]
TetrosL = [[0,0,0],[5,5,5],[5,0,0]]
TetrosJ = [[0,0,0],[6,6,6],[0,0,6]]
TetrosZ = [[7,7,0],[0,7,7],[0,0,0]]
TetrosS = [[0,8,8],[8,8,0],[0,0,0]]
LTetros = [ TetrosI,TetrosT,TetrosO,TetrosL,TetrosJ,TetrosZ,TetrosS]

```

Les variables d'état du Programme sont présentées dans le code ci-dessous. La Variable idpièce indique l'indice de la pièce courante dans la liste LTetros. Ainsi, le jeu démarre avec une pièce de type 1 en forme T. Les variables px, py et rot indique la position en x et y de la pièce dans la grille, ainsi que sa rotation 0 pour 0 degré, 1 pour 90 degrés à droite et ainsi de suite. Nous avons créé les valeurs KeyDown, KeyUp, KeyLef et KeyRigth, pour stocker l'état des événements précédent avec des touches fléchées soit il est enfoncé ou non. Le but est de pouvoir détecter les appuis sur ces touches.

Ce code représente la déclaration des variables d'états :

```
idpiece = 1
px = 6
py = 0
rot = 0
KEYDown = 0
KeyUp = 0
KeyLeft = 0
KeyRight = 0
```

3.2 Création du décor :

Pour l'affichage du décor de fond on a utilisé le code ci-dessous. Les valeurs sont stockées dans une liste de listes intitulées DECOR. Ainsi, Len (DECOR) correspond au nombre de lignes, et Len (DECOR[0]) au nombre de colonnes du jeu. En écrivant DECOR[y] [x], on accède donc à l'indice de couleur pour la case de coordonnées (x,y).

L'origine du décor (0,0) est positionnée en haut à gauche de l'écran. La variable LCASE définit la largeur d'une case en pixels, nous l'avons fixée à 20.

Pour dessiner la grille entièrement, nous utilisons une double boucle en x et y. Nous dessinons un carré plein grâce à la première fonction draw.rect(), puis les bords noirs avec le deuxième appel.

Le code ci-dessous représente l'affichage du décor du fond :

```
LCASE = 20
def AfficheDecors():
    for y in range(len(DECOR)):
        for x in range(len(DECOR[0])):
            xx = x * LCASE
            yy = y * LCASE
            id = DECOR[y][x]
            pygame.draw.rect(screen,LCoul[id],(xx,yy,LCASE,LCASE))
            pygame.draw.rect(screen,NOIR,(xx,yy,LCASE,LCASE),1)
```

Le décor est stocké dans une grille de 15 cases de large pour 18 cases de haut. Comme la largeur des cases fait 20 pixels, nous avons donc une fenêtre de taille 300x360 pixels. Nous définissons une Constante LIGNE VIDE composée de 2 colonnes sur la gauche et sur la droite, qui marquent les bords avec des cases grises, donc de code couleur associé 1. Les 11 cases centrales vides sont remplies avec la valeur 0. Le décor est défini comme une liste de 18 lignes. Le code suivant représente le décor :

```
LIGNE_VIDE = [1,1] + [0] * 11 + [1] * 2
DECOR = []
for i in range(16):
    DECOR.append(LIGNE_VIDE.copy())
DECOR.append([1]*15)
DECOR.append([1]*15)
```

Les 16 premières sont des lignes vides, et les 2 dernières sont remplies de 1. Ainsi lorsque l'on veut accéder à la case (x,y) du decor, nous allons devoir écrire DECOR [y] [x]. Pour créer les 16 lignes vides, nous utilisons la liste

LIGNEVIDE que nous copions grâce à la fonction copy(). Ceci est très important, car chaque ligne doit être indépendante !

La figure 3 représente le décor avec une grille de 15 cases de large et de 18 cases de haut. Les deux lignes du bas servent de fond et bloquent les pièces lors de leur descente.

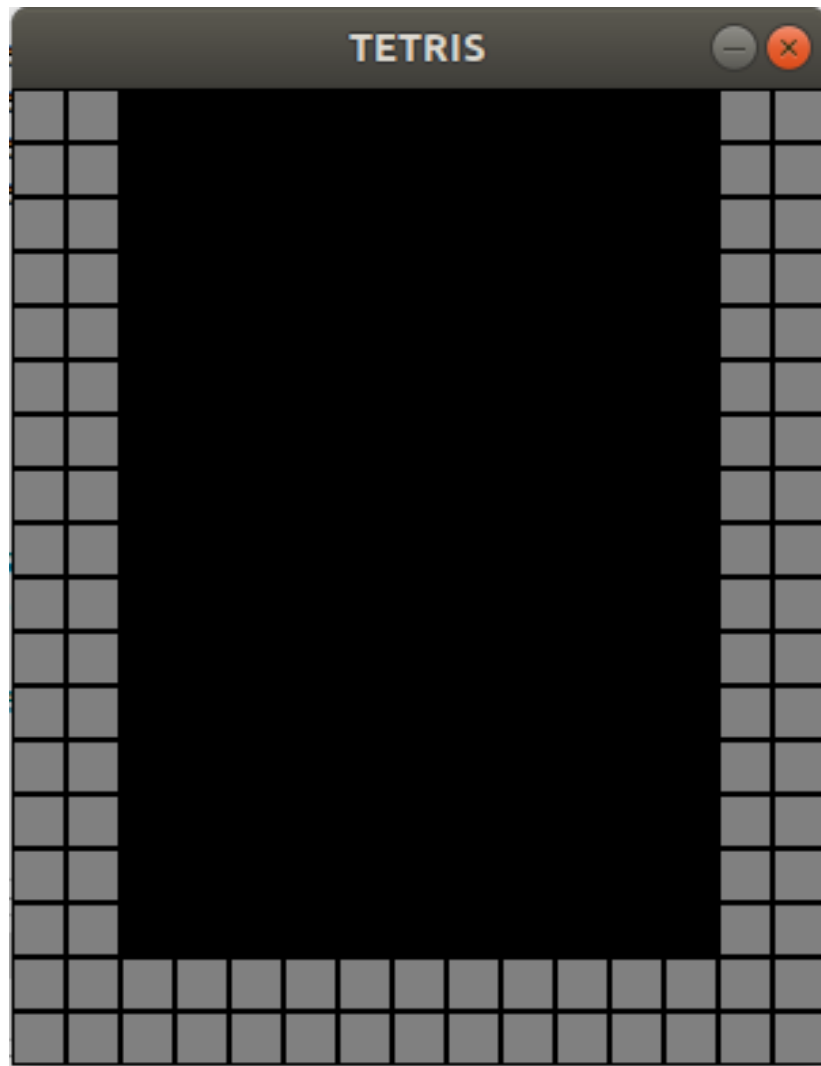


FIG. 3 – *Le décor de TETRIS*

3.3 Création des pièces :

Après avoir dessiné le décor, nous affichons la pièce courante par-dessus. Pour cela, la variable globale `idpiece` nous indique le type de pièce dans la liste globale `LTetro`. Ainsi, en écrivant `LTetro[idpiece]`, on récupère la forme de la pièce dans une liste de listes 3×3 . Comme nous avons formaté toutes nos pièces en 3×3 , nous utilisons une double boucle avec les indices `dx` et `dy` allant de 0 à 2.

Nous connaissons ainsi la coordonnée (dx, dy) de la case courante relativement à la position de la pièce. Nous connaissons aussi la position de la pièce dans le repère de la grille grâce aux deux variables globales : (px, py) . Ainsi, pour connaître la position de la case courante dans le repère de la grille il faut faire l'opération : $(px, py) + (dx, dy)$.

Nous parcourons ainsi l'intégralité des cases de la pièce, et lorsqu'un indice de couleur non nul est trouvé, nous dessinons un rectangle plein à la position souhaitée dans le décor.

Le code suivant représente affichage de toutes les pièces existantes :

```
def AffPiece():
    P = LTetro[idpiece]
    P = Rotn(P,rot)
    for dx in range(3):
        for dy in range(3):
            c = P[dy][dx]
            if c != 0:
                idcoul = int(c)
                xx = (px+dx) * LCASE
                yy = (py+dy) * LCASE
                pygame.draw.rect(screen,LCoul[idcoul],(xx,yy,LCASE,LCASE))
```

La figure 4 représente les pièces existantes dans notre jeu Tetris :

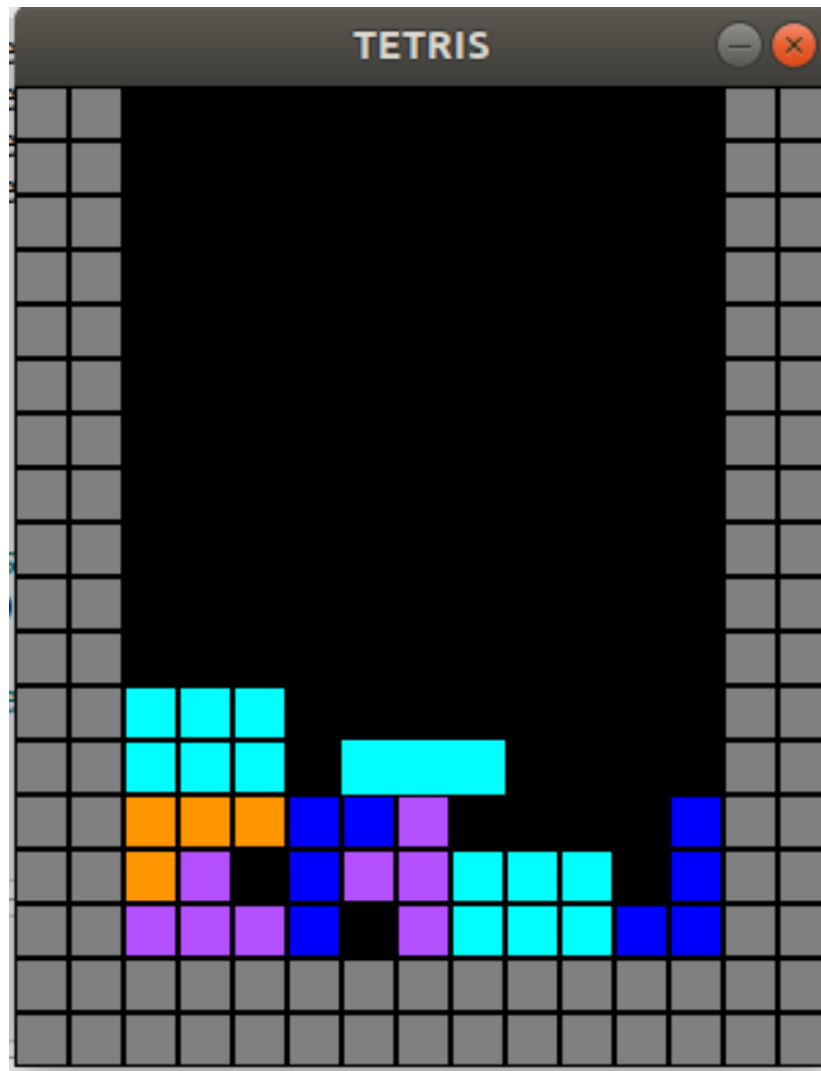


FIG. 4 – Les pièces existantes du jeu Tetris

3.4 Déplacement de la pièce :

Nous déplaçons la pièce courante avec une technique particulière. Nous utilisons la variable comptage qui comptabilise le nombre d'affichages effectués et cela se produit toutes les 1,33 seconde.

Pour appliquer ce déplacement, on appui sur (KRIGHT) pour déplacer la pièce à droite, (KLEFT) pour déplacer la pièce à gauche, (KUP) pour faire tourner la pièce de 90° et (KDOWN) pour faire descendre la pièce jusqu'en bas. La figure 5 illustre le déplacement d'une pièce dans le jeu TETRIS.

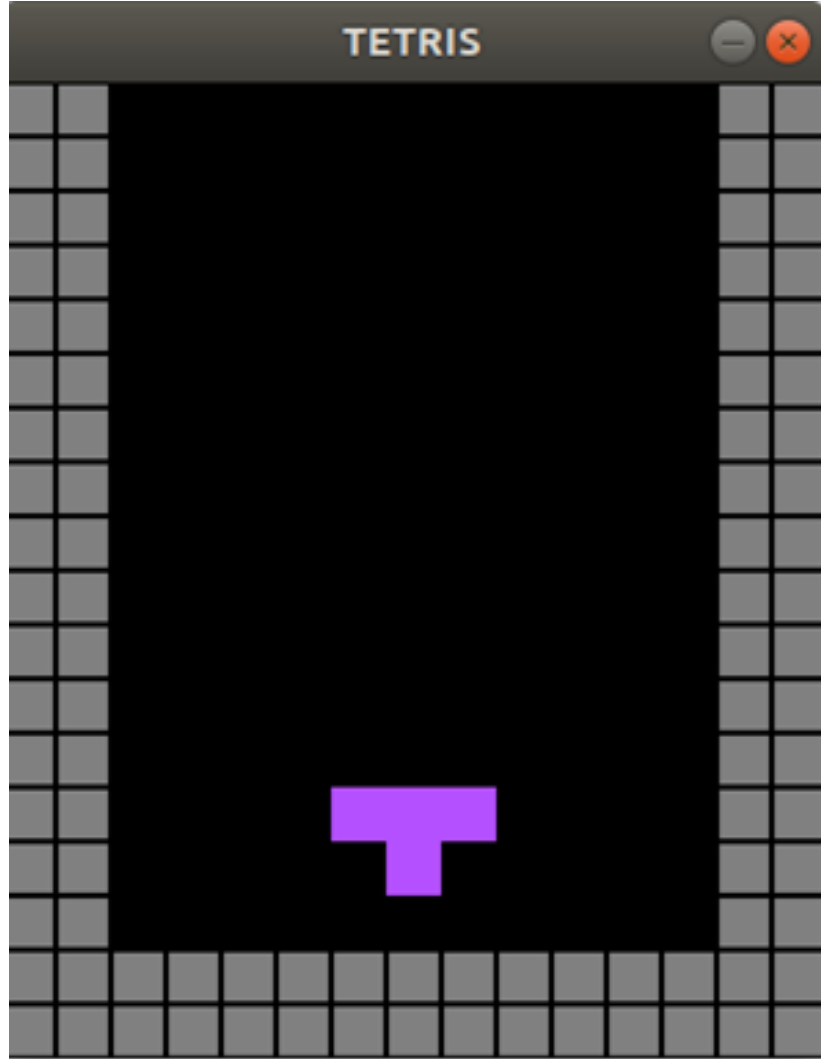


FIG. 5 – Illustration de déplacement d'une pièce

4 Gestion de la rotation, déplacements latéraux et la descente pour les tetrominos :

4.1 Gestion de la rotation :

On se focalise sur la gestion de la rotation de la pièce courante. À chaque appui sur la touche (KUP), la valeur de rot s'incrémente de 1 et la variable globale prend la nouvelle valeur de rot. Pour que cette variable globale soit comprise entre 0 et 3, on la divise par 4.

Avec la fonction Rot90Droite(P) qui, à partir d'une pièce 3 x 3, permet de tourner une pièce de 90° dans le sens des aiguilles d'une montre où La pièce Tetros correspond à une liste de listes, afin de créer une nouvelle pièce on va l'initialiser à partir d'une liste de listes contenant des 0 ou on va appliquer

une fonction `copy.deepcopy()` sur la pièce Tetros actuelle. Et ensuite, on suit les mêmes étapes pour la rotation à droite de 90°.

La figure 6, explicite un exemple de rotation avec une pièce P en entrée à gauche et la pièce R à droite. On note que dans tous les cas, la case centrale de la pièce ne change pas de position dans le carré 3 x 3.

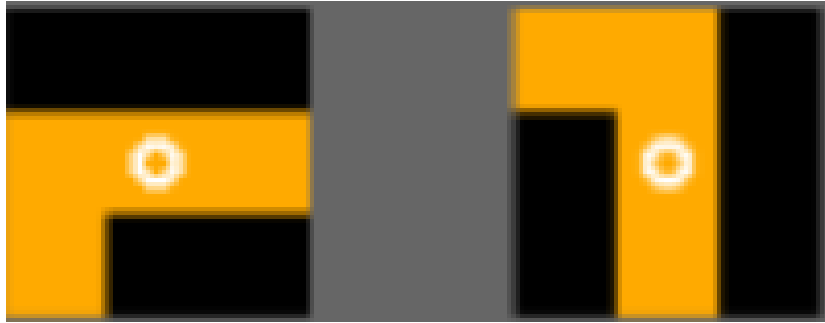


FIG. 6 – Illustration d'une rotation d'un Tetrominos. [3]

Dans cette fonction `Rot90Droite(P)` on a utilisé 2 options :

- Option 1 : On a écrit une instruction pour chacune des 8 cases, par exemple, pour la case 1 en haut à gauche : `R[0][0] = P[2][0]` et pour la case 02 : `R[0][1] = P[1][0]`.

- Option 2 : On a dressé une liste des positions des huit cases des bords, en tournant dans le sens des aiguilles d'une montre `L = [(0,0), (1,0), (2,0), (2,1), (2,2), (1,2), (0,2), (0,1)]`. Ainsi, on a créé une boucle for avec un indice `i` allant de 0 à 7, et on a mis la case à la position `R[i]` qui doit être initialisée avec la case `L[(i-2) % 8]`. Le code ci-dessous determine la gestion de la rotation pour les pièces du jeu TETRIS :

```
def Rot90Droite(P):
    R = copy.deepcopy(P) pos = [(0,0),(1,0),(2,0),(2,1),(2,2),(1,2),(0,2),(0,1)]
    R[1][1] = P[1][1]
    for i in range(8):
        x,y = pos[i]
        v = P[y][x]
        x,y = pos[(i+2)]
        R[y][x] = v
    return R
```

La fonction `Rotn(P, nb)` qui calcule la pièce P après nb rotations. Pour faire nb rotation sur une pièce P on a :

- Initialisé la pièce 3 x 3 sous forme d'une liste de listes, ensuite utilisé la fonction `copy.deepcopy` pour cloner la pièce P parce que quand la variable nb vaut 0, il n'y aura aucune rotation effectuée et c'est la copie de la pièce initiale qui sera retournée.

- Pour effectuer autant de rotations nécessaires, nous utilisons la fonction `Rot90Droite(P)`.

Et enfin, on modifie la fonction `AffPiece` pour qu'elle tienne compte de la variable globale `rot`. On remarque que lorsqu'on appuie sur la touche [Flèche en haut] du clavier, la pièce courante tourne. Le code suivant montre ce déroulement :

```
def Rotn(P,nb):
    R = copy.deepcopy(P)
    for i in range(nb):
        R = Rot90Droite(R)
    return R
```

4.2 Déplacements latéraux :

Pour la détection des collision on crée une fonction `DetectCollision` qui détermine suivant une pièce, une rotation et une position (x,y) données, s'il y a collision avec le decor ou non pour cela :

- On fait la rotation sur la pièce pour obtenir sa bonne orientation, puis on crée d'une double boucle d'indices dx et dy afin de parcourir les cases de la pièce, en fin on fait la comparaison de chaque case (dx,dy) de la pièce avec la case (x+dx,y+dy) du décor. Si les deux cases sont non vides, alors il y a collision, et on retourne vrai dans ce cas.

- Lors de l'appui sur les touches [Flèche à gauche] et [Flèche à droite] du clavier, on vérifie d'abord que la future place de la pièce n'est pas en collision avec le décor. Si aucune collision n'est détectée, alors on modifie la position de la pièce en faisant : $px -= 1$. Le code suivant représente la detection de collision avec le décor :

```
def DetectCollision(idP,rot,x,y):
    P = Rotn(LP[idP],rot)
    for dx in range(3):
        for dy in range(3):
            vdecor = DECOR[y+dy][x+dx]
            vpiece = P[dy][dx]
            if vdecor != 0 and vpiece != 0 :
                return True
    return False
```

4.3 Gestion de la descente :

Soit la fonction FusionDecor qui, suivant une pièce, une rotation et une position (x, y) données, fixe cette pièce dans le décor. Cette fonction est comparable à la fonction DetectCollision.

La pièce courante descend automatiquement d'une ligne. Lorsque la pièce est susceptible de descendre, on examine si sa position future produit une collision, si c'est le cas, elle ne doit pas descendre donc elle sera stoppée par la fonction FusionDéco qui permet de figer la pièce. Après une nouvelle pièce se génère. Le code suivant représente la fusion du décor :

```
def DetectCollision(idP,rot,x,y):
    P = Rotn(LP[idP],rot)
    for dx in range(3):
        for dy in range(3):
            vdecor = DECOR[y+dy][x+dx]
            vpiece = P[dy][dx]
            if vdecor != 0 and vpiece != 0 :
                return True
    return False

def FusionDecor(idP,rot,x,y):
    P = Rotn(LP[idP],rot)
    for dx in range(3):
        for dy in range(3):
            if P[dy][dx] != 0 :
                DECOR[y+dy][x+dx] = P[dy][dx]
    NextPiece()
    RemoveLignes()
```

La fonction `NextPiece` permet d'initialiser une nouvelle pièce. Et c'est grâce au package `random` qu'on pourrait choisir une pièce au hasard. Sa position sera forcément sur la ligne 0 et au milieu de la grille, c'est-à-dire, à l'abscisse 6. En revanche, on peut choisir sa rotation aléatoirement. Le prochain code illustre l'initialisation :

```
def NextPiece():
    global idpiece,px,py,rot
    idpiece = random.randint(0,len(LP)-1)
    px = 6
    py = 0
    rot = random.randint(0,3)
```

Après la fusion d'une pièce avec le décor, il se peut qu'une ou plusieurs lignes pleines apparaissent pur cela on crée La fonction `RemoveLigne` qui retirera l'ensemble des lignes pleines du décors. Le code suivant montre cela :

```
def RemoveLignes():
    nb = len(DECOR)
    for i in range(2,nb):
        p = nb-i-1
        if not 0 in DECOR[p]: ligne pleine
        DECOR.pop(p)
        while len(DECOR) != nb:
            DECOR.insert(0,LIGNE_VIDE.copy())
```

On a parcouru les lignes du décor et si on a pu détecter une ligne pleine, on la retire. Cependant, si on a parcouru les lignes à partir de la ligne 0, on prend le risque. En effet, si on doit retirer la ligne 0 par exemple, la ligne 1 va prendre sa place, mais celle-ci correspondra alors à la ligne anciennement numéro 2, et on n'aura jamais testé la ligne 1, ce qui représente une erreur. Pour éviter ce type d'erreur, lorsque on retire des éléments dans une liste pendant un parcours, il faut partir de la fin. Ainsi, on a dans le sens des indices décroissants, le retrait d'un élément ne modifie les indices restant à parcourir.

- La boucle `for` avec un indice partant de 0 jusqu'à 15 compris. Les deux dernières lignes en bas ne doivent pas être traitées.

- On a fait un calcul pour trouver la valeur de l'indice qui parcourt les lignes en sens inverse, c'est-à-dire de l'indice 15 à l'indice 0.

- Puis on a testé la ligne associé à ce nouvel indice pour savoir si elle est pleine :

- Pour cela, il suffit de détecter si une valeur 0 est présente dans la ligne courante.

- Si la ligne est pleine, on l'a retiré grâce à la fonction `DECOR.pop(index)`.

- Une fois le parcours terminé, des lignes ont pu être supprimées. Ainsi, tant que le nombre de lignes dans la liste `DECOR` est insuffisant, on a rajouté des lignes vides à l'indice 0 grâce à la fonction `DECOR.insert(0,LIGNE_VIDE.copy())`.

5 Rendu final :

La version suivante est la version finale du jeu Tetris! Sur notre projet, la pièce a été placée sur la gauche et une ligne complète a été créée, puis retirée. La figure 7 représente la version finale du jeu Tetris :

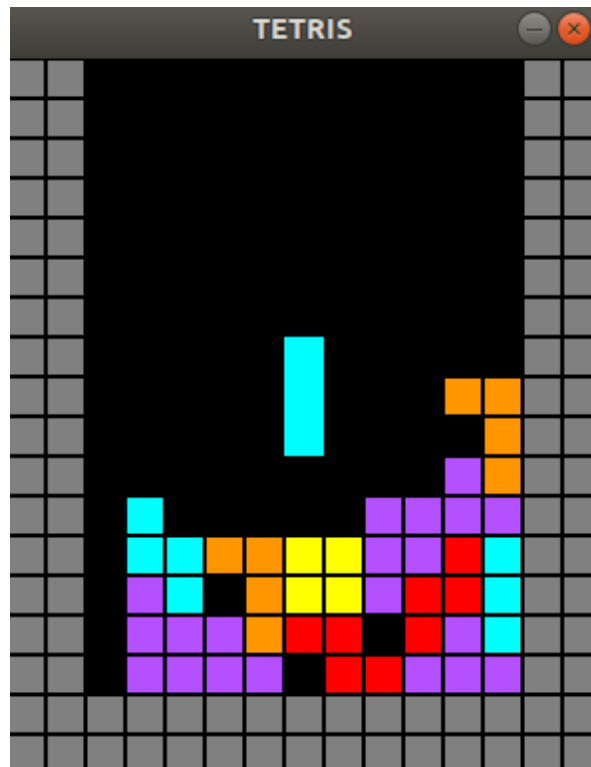


FIG. 7 – *Version final du jeu TETRIS*

6 Conclusion :

Tous comme le temps, les gens, les goûts changent, mais pas les jeux. Certains jeux vidéo sont entrés directement dans l'histoire et, malgré le temps qui passe, gardent leur charme intact. Un exemple avant tout ? Le « Tetris », qui a traversé des dizaines de plates-formes, de dispositifs et de jeux depuis les années 80, a réussi à conquérir des utilisateurs de tous âges.

Le jeu TETRIS est disponible pour presque toutes les consoles de jeux vidéo et systèmes d'exploitation informatiques, ainsi que sur des appareils tels que les téléphones mobiles.

Tous simplement, Le jeu Tetris, c'est de marquer autant de points que possible en effaçant les lignes horizontales des blocs. Le joueur doit tourner, se déplacer et laisser tomber les Tetriminos qui tombent à l'intérieur de la Matrice (terrain de jeu). Les lignes sont effacées lorsqu'elles sont remplies de blocs et n'ont pas d'espaces vides. Au fur et à mesure que les lignes sont dégagées, le niveau augmente et les Tetriminos tombent plus rapidement, ce qui rend le jeu progressivement plus difficile. Si les blocs atterrissent au-dessus du haut du terrain de jeu (décor), le jeu se termine.

Enfin, un conseil pour vous, afin d'avoir le plus grand score, vous devrez mettre vos compétences organisationnelles et votre endurance à l'épreuve en franchissant autant de lignes que possible, sinon, si vous empilez les Tetriminos trop haut, le jeu sera terminé !

Je vous invite à vous amuser avec le jeu TETRIS !

Références

- [1] <https://www.marca.com/esports/2019/03/31/5ca0062f22601d3c5f8b456d.html>. consultée le 28/12/2021.
- [2] <https://gigazine.net/news/20191116-tetris-algorithm/>. consultée le 28/12/2021.
- [3] <https://gamedev.stackexchange.com/questions/17974/how-to-rotate-blocks-in-tetris>. consultée le 28/12/2021.
- [4] <https://strategywiki.org/wiki/Tetris>. consultée le 22/12/2021.
- [5] <https://fr.wikipedia.org/wiki/Tetris>. consultée le 22/12/2021.
- [6] <https://www.bing.com/videos/search?q=Comment+programmer+un+TETRIS+> consultée le 22/12/2021.
- [7] <https://www.byteacademy.co/blog/tetris-pygame-python>. consultée le 22/12/2021.