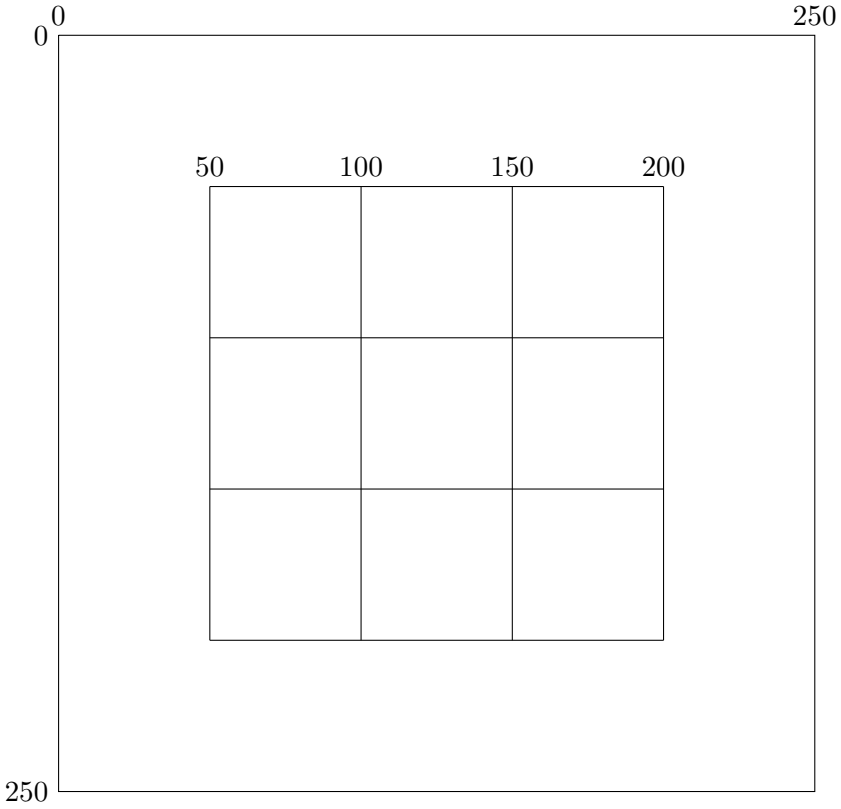
TP10 - Le jeu du morpion

Documentation : `https://github.com/kitao/pyxel/blob/main/docs/README.fr.md`

I Partie graphique



Voici le code minimal pour travailler avec la bibliothèque `pyxel`.

```
import pyxel

def update():
    pass
def draw():
    pass

pyxel.init(250, 250) #crée une fenêtre de 250 pixels par 250 pixels

pyxel.run(update, draw)
```

Q1. Ecrire une fonction `dessiner_quadrillage()` qui dessine le quadrillage du morpion.
Pour tester cette fonction, l'appeler dans la fonction `draw()`.

Q2. Dans ce quadrillage, on devra dessiner des croix ou des ronds.

On admet que le joueur 1 dessine des ronds et le joueur 2 des croix.

Ecrire une fonction `dessiner_symbole(x, y, j)` qui a pour paramètres `(x,y)` les coordonnées du centre de la case, `j` prenant les valeurs 1 ou 2, et qui dessine un rond ou une croix (suivant la valeur de `j`) centrés en `(x,y)`.

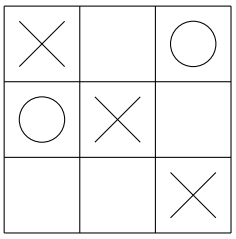
Par exemple `dessiner_symbole(75, 75, 1)` dessine un rond dans la case en haut à gauche.

II Modélisation de la grille

II.1 La variable globale `grille`

On utilise une liste contenant 3 sous-listes pour représenter l'état de la grille. Chaque sous-liste contient les éléments 0, ou 1, ou 2.

`grille = [ [2, 0, 1], [1, 2, 0], [0, 0, 2] ]` correspond à la grille :



Q3. Définir la variable globale `grille`. *On la place juste avant la fonction* ***`run`****.*

Q4. Faire apparaître la souris en écrivant l'instruction `pyxel.mouse(True)` juste après `init`.

II.2 Dessiner la grille

Le jeu se joue avec la souris. Ainsi pour ne pas voir les traces de la souris, on doit écrire dans la fonction `draw()`, appelée 30 fois par seconde, une fonction `dessiner_grille()` qui :

- efface la fenêtre graphique ;
- dessine le quadrillage ;
- met les croix et les ronds.

Q5. Ecrire la fonction `dessiner_grille()`.

Déroulement de la fonction :

— On efface la fenêtre graphique ;

— On dessine le quadrillage ;

— On parcourt la liste `grille` :

  — Si l'élément de `grille` est 1 :

    — on calcule les coordonnées du centre de la case en fonction des indices de `grille` ;

    — on met une croix ;

  — Si l'élément de `grille` est 2 :

    — on calcule les coordonnées du centre de la case en fonction des indices de `grille` ;

    — on met un rond.

Tester cette fonction en l'appellant dans `draw()` et en modifiant la variable `grille`.

III Le jeu

III.1 Mise à jour de la grille

A chaque clic de souris valide, on doit modifier la grille et changer de joueur.

On utilise une variable globale `joueur` qui prend uniquement deux valeurs 1 ou 2.

- si `joueur` vaut 1 : c'est au joueur 1 de joueur ;
- si `joueur` vaut 2 : c'est au joueur 2 de joueur.

Q6. Définir la variable globale `joueur` et l'initialiser. *On la place après* `grille`.

La fonction `btn(touche)` prend en paramètre `touche` un entier représentant une touche et renvoie `True` si la touche est appuyée et `False` sinon.

On peut voir la liste des touches ici :
`https://github.com/kitao/pyxel/blob/main/python/pyxel/__init__.pyi`

Chaque touche est codée par un entier. Pour plus de lisibilité, on stocke cet entier dans une variable. Par exemple :

— La touche `MOUSE_BUTTON_LEFT` vaut 20004.

— La touche `MOUSE_BUTTON_RIGHT` vaut 20006.

Q7. Ecrire une fonction `mise_a_jour_grille(numero)` qui a pour paramètre le numéro du joueur et qui, à **chaque clic gauche** de souris *(qu'on supposera valide)* :

- modifie `grille`
- renvoie le numéro du joueur suivant.

Tester en écrivant :

```
def update():
  global joueur
  joueur = mise_a_jour_grille(joueur)
def draw():
  dessiner_grille()
```

III.2 Validité du clic

A chaque clic de souris valide, la grille est modifiée.
Donc la grille ne doit pas être modifiée :

- si on clique en dehors de la grille ;
- si on clique sur une case déjà remplie.

Q8. Ecrire une fonction `clic_valide()` qui renvoie `True` si le clic est valide et `False` sinon.
Rajouter l'appel de cette fonction dans la fonction `mise_a_jour_grille()`

```
def mise_a_jour_grille():
  if pyxel.btn(pyxel.MOUSE_BUTTON_LEFT):
    if clic_valide() :
      ....
```

III.3 Numéro du gagnant

La grille permet de déterminer le numéro du gagnant.

Q9. Ecrire une fonction `gagnant()` qui renvoie 1 si le joueur 1 gagne, 2 si le joueur 2 gagne et 0 sinon.

III.4 Fin de partie

Une partie est finie :

- s'il y a un gagnant
- ou si la grille est complètement remplie

Q10. Ecrire une fonction `fin_de_partie()` qui renvoie `True` si la partie est finie et `False` sinon. Compléter le code :

```
def draw():
  dessiner_grille()
  if fin_de_partei():
    #code pour afficher qui gagne
```

Q11. Modifier le programme de sorte que le joueur 1 joue contre l'ordinateur. Celui-ci jouera de façon aléatoire.

Q12. Modifier le programme de sorte que l'ordinateur ne joue plus de façon aléatoire.