Le jeu de la taupe sur Pyxel

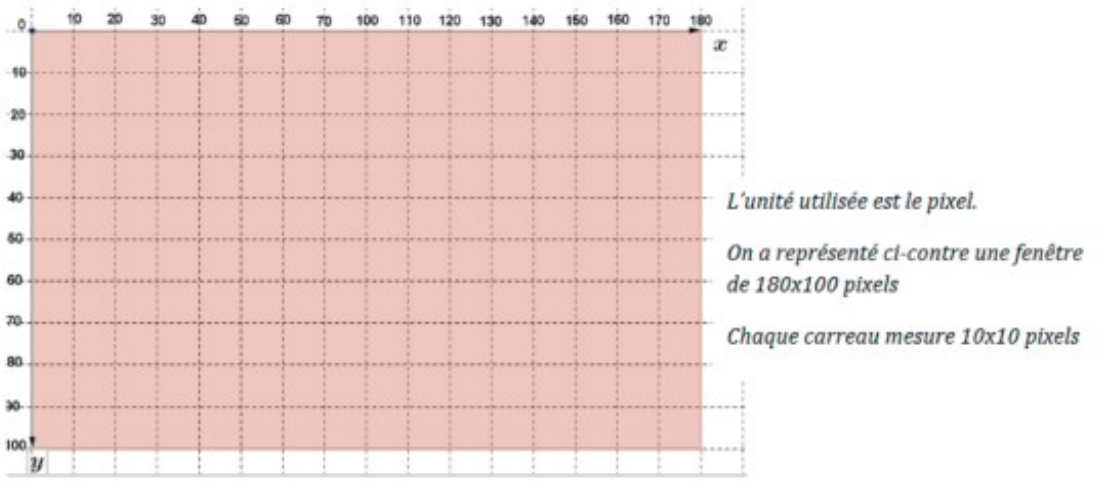
Nous allons coder un Jeu de la Taupe. C'est un jeu d'arcade dans lequel on fait apparaitre aléatoirement des objets sur une fenetre graphique. l'objectif étant de cliquer dessus le plus rapidement possible.

L'Environnement de conception est Pyxel. Malheureusement pyxel n'est pas installé sur Capytale. Si vous installez spyder ou Edupython,

vous pouvez installer le module pyxel ou sur https://github.com/kitao/pyxel/blob/main/docs/README.fr.md#comment-installer.

Le choix de Pyxel est motivé par le fait que la création d'interfaces graphiques y soit particulièrement aisée.

Vous pouvez aussi créer une session sur repl.it pour recuperer votre travail. ATTENTION faites **+Create** et choisir Pyxel :



Je ne conseille pas vraiment de travailler dessus, l'interface vidéo n'est pas agreable mais l'installation en local de pyxel n'est pas aisé.

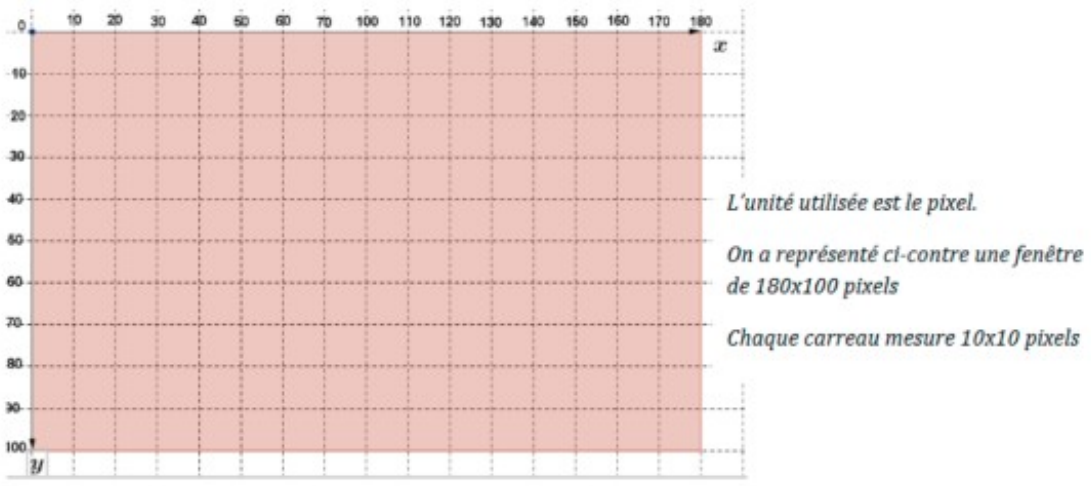
Vous pouvez avoir une documentation au lien suivant :

https://github.com/kitao/pyxel/blob/main/docs/README.fr.md#documentation-de-lapi

Le repérage dans la fenêtre

Par convention, le coin en haut à gauche de la fenêtre correspond aux valeurs x=0 et y=0.

Les valeurs x sont croissantes vers la droite et les valeurs y sont croissantes vers le bas.



Etape 1 : création de la fenêtre de jeu

- pyxel.init(128, 128, title="Tape Taupe") initialise le programme, comme par exemple la création d'une fenêtre de dimensions : 128x128 pixels
- La fonction `update()` est une boucle infinie qui gére la mise à jour des variables.
- La fonction `draw()` est une boucle infinie, qui permet l'actualisation de la fenêtre 30 fois par seconde.
- `pyxel.run(update, draw)` lance les deux fonctions.

Les références complètes des commandes disponibles sont répertoriées sur le site Documentation pyxel.

Copier le code ci-dessous et tester le pour faire apparaitre une fenetre graphique.

```
import pyxel

# taille de la fenetre 128x128 pixels
pyxel.init(128, 128, title="Tape Taupe")

# ==================================================
# == UPDATE
# ==================================================
def update():
    """mise à jour des variables (30 fois par seconde)"""
# ==================================================
# == DRAW
# ==================================================
def draw():
    """création des objets (30 fois par seconde)"""
    pyxel.cls(5) #Le fond est rempli avec la couleur numéro 5
```

```
pyxel.run(update, draw)
```

Pour la couleur

Vous avez le choix de 15 couleurs qui sont listés dans

```
pyxel.colors.to_list()
```

On peut changer la couleur avec `pyxel.colors.from_list()` en changeant la liste des nombres. par exemple en faisant

```
pyxel.colors.from_list(0x0 ,0x2b335f ,0x7e2072 ,0x19959c ,0x8b4852 ,0x
395c98 ,0xa9c1ff ,0xeeeeee ,0xd4186c ,0xd38441 ,0xe9c35b ,0x70c6a9 ,0x
7696de ,0xa3a3a3 ,0xff9798 ,0xedc7b0)
```

Une couleur est représentée par un nombre en hexadécimal `0xabcdef`. Les trois nombres `ab`, `cd` et `ef` correspondent au codage héxadécimal des couleurs RVB (RGB en anglais). Chaque nombre est un entier compris entre 0 et 255 (00 et FF en hexadécimal) qui donnent la quantité de Rouge, Vert et Bleu. Il y a donc $256 \times 256 \times 256 = 16777216$ couleurs possibles.

On peut faire un choix de couleur sur le site htmlcolorcodes

Tester le code

Etape 2 : Faire apparaitre des cercles de tailles aléatoires.

- La fonction de creation de l'objet cercle. Regarder sa documentation.
- Les variables `posx`, `posy`, et `rayon` sont utilisées dans les deux fonctions. Ce sont donc des variables globales pour cela on ajoute l'instruction:

```
global posx,posy,rayon
```

Elles représentent les abscisses et ordonnées du centre du cercle et son rayon

On importe `randint()` et on ajoute des lignes à la fonction `draw()` :

```
# -*- coding: utf-8 -*-
from random import randint
import pyxel

# taille de la fenetre 128x128 pixels
pyxel.init(128, 128, title="Tape Taupe")

# ==========================================================
# == UPDATE
# ==========================================================
```

```
def update():
    """mise à jour des variables (30 fois par seconde)"""
    global posx, posy, rayon,touche

    posx = randint(0, pyxel.width)  # width est la largeur de la
fenétre
    posy = randint(0, pyxel.height)  # height est la hauteur de la
fenetre
    rayon = randint(1, 10)


# ==================================================
# == DRAW
# ==================================================
def draw():
    """création des objets (30 fois par seconde)"""
    pyxel.cls(5)  #Le fond est rempli avec la couleur numéro 5

    pyxel.circ(posx, posy,  rayon, 0 )  #Le cercle est rempli avec la
premiere couleur
pyxel.run(update, draw)
```

Tester le code

Ouaahhhh ! C'est jolie mais cela va trop vite.

Etape 4 : Ralentir l'apparition des cercles

Il faut donc éviter de changer trop souvent certaines variables comme `posx`, `posy` et `diametre` pour afficher les cercles.

On utilise une variable `pyxel.frame_count` qui compte le nombre de rafraichissement de la fenêtre. On fera apparaitre une nouvelle nouvelle cible modulo une certaine valeur. On utilise donc `%` le reste de la division euclidienne.

```
# -*- coding: utf-8 -*-
from random import randint
import pyxel

# taille de la fenetre 128x128 pixels
pyxel.init(128, 128, title="Tape Taupe")

# ==================================================
# == UPDATE
# ==================================================
```

```
def update():
    """mise à jour des variables (30 fois par seconde)"""
    pyxel.mouse(True)
    global posx, posy, rayon,touche
    if pyxel.frame_count % 10 == 0:  #Une nouvelle cible est crée tous
les 100 cycles
        posx = randint(0, pyxel.width)  # width est la largeur de la
fenétre
        posy = randint(0, pyxel.height)  # height est la hauteur de la
fenetre
        rayon = randint(1, 10)



# ==========================================================
# == DRAW
# ==========================================================
def draw():
    """création des objets (30 fois par seconde)"""
    pyxel.cls(5)  #Le fond est rempli avec la couleur numéro 5

    pyxel.circ(posx, posy,  rayon, 0 ) #Le cercle est rempli avec la
premiere couleur (numero 0)


pyxel.run(update, draw)
```

Tester le code

Etape 4: comparer la position de la souris avec la position du cercle

Si la souris est sur le cercle, on affiche "Bravo" sinon "Perdu".

On fais apparaitre le pointeur de la souris avec `pyxel.mouse(True)`. La position de la souris est donnée par les variables de **pyxel** `mouse_x` et `mouse_y`. Utilisez les pour afficher le message "Bravo" ou "Perdu" suivant la position de la souris.

Je rajoute `touche` comme variable globale.

Faut il mettre le code suivant à compléter dans `update()` ou `draw()` ?

```
if ...:
            touche=True
else:
            touche=False
```

Et celui là ?

```
if ...:

    pyxel.text(10, 30,"Bravo",2)
else:
    pyxel.text( 10, 30,"Perdu",2)
```

Tester le code

On peut positionner le texte en fonction de la taille de la fenétre. Pour cela on peut utiliser les variables globales `width` et `height` qui ont été créées automatiquement à l'initiation de pyxel.

Positionner le texte au centre de la fenétre en utilisant `width` et `height` et mettez votre code ici.

```
#Votre code
```

Tester le code

Etape 5 : Mettre un carré à la place de l'ellipse.

- Utiliser la fonction `rect(x, y, w, h, col)` à la place de `cerc`. Regarder la documentation pour savoir à quoi correspondent x, y, w, h et col.
- **Changer la condition pour que le carré soit touché par la souris**

Pour l'instant, il faut faire glisser la souris sur l'objet pour avoir une interaction. Je voudrais que cela soit un clic qui valide l'interaction de la souris avec l'objet.

Etape 6 : Cliquer sur la souris et créer un évenement.

Je rajoute une condition qui agit quand on clique sur la souris avec la fonction `btn(key)`.

Regarder sa documentation.

Chercher quel est la nom de `key` pour le clic gauche dans la liste Des variables.

Dans `update()`, je compare la position de la souris avec l'objet uniquement quand j'ai cliqué.

```
if pyxel.btn(.....):
        if ...:
            touche=True
        else:
            touche=False
```

Je n'oublie pas de rendre `touche` comme variable globale. Si cela ne marche pas, peut être avez vous oublié d'initialiser la variable `touche` avant de lancer `run`

Tester le code

Etape 7: Ne faire apparaitre le résultat qu'au moment du clic.

On rajoute encore une variable globale boléenne `resultat`. Il faut faire 4 choses:

- L'initialiser et la rendre globale;
- La rendre `True` au bon moment pour faire l'affichage.
- La rendre `False` dés que l'on créée un nouvel objet.
- L'utiliser comme condition supplementaire d'affichage.

A vous de faire le code. De le tester et de l'écrire dans la cellule suivante:

```
#Votre code ici
```

Tester le code !!!!

Oui je me répéte mais c'est l'erreur du mauvais codeur, il va ajouter des dizaines de lignes sur un code qui marche et tout d'un coup, il va tester et cela ne marche pas. Et le probleme, où est l'erreur ? Parfois c'est tout simplement la premiére ligne qu'on a ajouté à un code qui marchait bien. On aura écrit des dizaines de lignes pour "rien".

Donc **Tester le code dès que cela est possible**.

Etape 8 : faire des améliorations

Maintenant, c'est à vous de jouer, le jeu est trés sommaire. Il faut le rendre plus attactif.

Pour finaliser le projet, on peut :

- Changer le fond
- Augmenter la difficulté
- Changer l'image de l'objet
- Rajouter un score...
- Ajouter des malus..des bonus...
- Faire perdre le joueur et donner un score final
- Ajouter des niveaux...

- Mettre plusieurs cibles. Là cela risque d'être dur car il faut presque reprendre le code au début.

  etc.

Les régles d'un projet informatique:

- Commenter bien votre projet
- Faire des essais au fur et à mesure. N'ecrivez jamais beaucoup de lignes avant de tester
- Utiliser la documenttion des commandes Processing.py.

Et SURTOUT, SURTOUT, SURTOUT toujours garder une trace de chaque code qui marche bien.

D'autres idées

- Pour charger une image

```
A faire
```

- Tester une couleur sous la souris

```
A faire
```

- Faire une zone de transparence sur votre image avec Gimp https://doc.ubuntu-fr.org/tutoriel/comment_creer_un_gif_transparent_avec_gimp

Stockage des differentes versions

```
#Version 1

# -*- coding: utf-8 -*-
from random import randint
import pyxel

# taille de la fenetre 128x128 pixels
pyxel.init(128, 128, title="Tape Taupe")

# ==========================================================
# == UPDATE
# ==========================================================


def update():
    """mise à jour des variables (30 fois par seconde)"""
    global posx, posy, rayon,touche

    posx = randint(0, pyxel.width)  # width est la largeur de la
```

```
fenétre
    posy = randint(0, pyxel.height)  # height est la hauteur de la
fenetre
    rayon = randint(1, 10)


# ==========================================================
# == DRAW
# ==========================================================
def draw():
    """création des objets (30 fois par seconde)"""
    pyxel.cls(5)  #Le fond est rempli avec la couleur numéro 5

    pyxel.circ(posx, posy,  rayon, 0 )  #Le cercle est rempli avec la
premiere couleur
pyxel.run(update, draw)



#Version 2

#Version 3
```

ETC...