Processing : interaction

Après avoir vu comment créer des dessins et des animations, nous allons voir comment créer des interactions avec l'utilisateur à l'aide du clavier et de la souris.

1. Interaction à l'aide de la souris

Processing propose quatre fonctions qui doivent être complétées :

- le code se trouvant dans la fonction **`mousePressed()`** est exécuté une fois quand l'utilisateur appuie sur un des boutons de la souris ;
- le code se trouvant dans la fonction **`mouseReleased()`** est exécuté une fois quand un bouton de la souris qui avait été préalablement pressé est relâché ;
- le code se trouvant dans la fonction **`mouseMoved()`** est exécuté à chaque boucle de la fonction **`draw()`** tant que la souris se déplace ;
- le code se trouvant dans la fonction **`mouseDragged()`** est exécuté à chaque boucle de la fonction **`draw()`** tant que la souris se déplace et que le bouton de la souris est enfoncé.

Ces quatre fonctions ne prennent aucun paramètre et ne retournent aucune valeur.
Pour pouvoir utiliser les quatre fonctions, il faut que la fonction **`draw()`** soit présente dans le programme (même si elle est vide).

✍ Tester et analyser le code suivant :

```
r=10

def setup():
    size(400, 400)
    noStroke()
    fill(0)

def draw():
    background(255)
    ellipse(200, 200, 2*r, 2*r)

def mousePressed():
    global r
    if r < 200:
        r = r + 5
```

✍ Tester et analyser le code suivant :

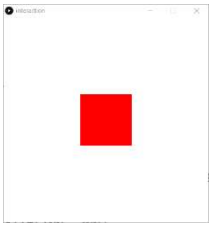
```
r=10

def setup():
    size(400, 400)
    noStroke()
    fill(0)

def draw():
    background(255)
    ellipse(200, 200, 2*r, 2*r)

def mousePressed():
    global r
    if r < 200:
        r = r + 100
```

```
def mouseReleased():
    global r
    r = r - 100
```



✍ Créer un programme permettant d'afficher un carré vert au centre de la fenêtre. En cas d'appui sur un bouton de la souris le carré devient rouge. Ce même carré redevient vert dès que le bouton de la souris est relâché.

Il est possible de connaitre les coordonnées du curseur de la souris grâce aux variables proposées par Processing : **mouseX** et **mouseY**.

✍ Tester et analyser le code suivant :

```
def setup():
    size(200, 200)
    noStroke()
    fill(0)

def draw():
    background(255)
    ellipse(mouseX, mouseY, 30, 30)
```

✍ Créer un programme qui permet d'afficher un nouveau carré à chaque clic de souris (sans effacer les carrés déjà présents). Le centre du carré doit se trouver au niveau du pointeur de la souris au moment du clic (voir **rectMode()**). La couleur du carré doit être aléatoire.
Aide : Dans une fonction, l'instruction **pass** permet de ne rien faire.

✍ Tester et analyser le code suivant :

```
val = 0

def setup():
    size(200, 200)
    background(255)

def draw():
    fill(val)
    rect(75, 75, 50, 50)

def mouseMoved():
    global val
    val = val + 5
    if val > 255:
        val = 0
```

✍ Tester et analyser le code suivant :

```
val = 0

def setup():
```

```
    size(200, 200)
    background(255)

def draw():
    fill(val)
    rect(75, 75, 50, 50)

def mouseDragged():
    global val
    val = val + 5
    if val > 255:
        val = 0
```

mouseX et **mouseY** donnent la position de la souris à l'instant t, **pmouseX** et **pmouseY** permettent d'avoir les coordonnées de la souris à l'instant t-dt, avec dt le temps qui s'est écoulé entre 2 boucle de la fonction **draw()**.

✍ Tester et analyser le code suivant :

```
def setup():
    size(300, 300)
    background(255)

def draw():
    pass

def mouseDragged():
    line(pmouseX, pmouseY, mouseX, mouseY)
```

2. Interaction à l'aide du clavier

Processing propose deux fonctions qui doivent être complétées :

- le code se trouvant dans la fonction **keyPressed()** est exécuté une fois quand l'utilisateur enfonce une touche ;
- le code se trouvant dans la fonction **keyReleased()** est exécuté une fois quand une touche du clavier est relâchée.

Ces deux fonctions ne prennent aucun paramètre et ne retournent aucune valeur.
Pour pouvoir utiliser les deux fonctions, il faut que la fonction **draw()** soit présente dans le programme (même si elle est vide).

✍ Tester et analyser le code suivant :

```
def setup():
    size(200, 200)
    fill(0)

def draw():
    background(255)
    ellipse(100, 100, 50, 50)

def keyPressed():
    fill(255)

def keyReleased():
    fill(0)
```

Il est possible de détecter la touche qui a été frappée grâce à la variable **key**.

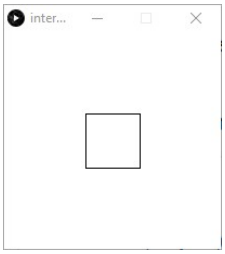
✍ Tester et analyser le code suivant :

```
def setup():
    size(200, 200)
    fill(0)
    textAlign(CENTER)
    background(255)
    text ("Appuyez sur une touche ", 100, 100)

def draw():
    pass

def keyPressed():
    background(255)
    text ("touche : "+key, 100, 100)
```

✍ Créer un programme permettant d'afficher un carré incolore au centre de la fenêtre. En cas d'appui sur la touche r, le carré devient rouge, en cas d'appui sur la touche v, le carré devient vert et en cas d'appui sur la touche b, le carré devient bleu. Ce même carré redevient incolore dès que la touche est relâchée.



Il est aussi possible de détecter l'appui sur des touches qui ne sont pas des lettres ou des chiffres (par exemple, les "flèches", ...).

✍ Tester et analyser le code suivant :

```
def setup():
    size(200, 200)
    fill(0)
    textAlign(CENTER)
    background(255)

def draw():
    pass

def keyPressed():
    background(255)
    if key == CODED:
        if keyCode == UP:
            text ("HAUT", 100, 100)
        if keyCode == DOWN:
            text ("BAS", 100, 100)
        if keyCode == LEFT:
            text ("GAUCHE", 100, 100)
        if keyCode == RIGHT:
            text ("DROIT", 100, 100)
```

✍ Créer un programme permettant d'afficher une "balle" noire. Cette balle pourra être déplacée à l'aide des flèches du clavier. La balle ne devra pas pouvoir sortir de la fenêtre.

Sources : https://pixees.fr/informatiquelycee/python_proc_a6.html