Activités de prise en main de la micro:bit

I

Nous allons voir ici des petits exercices permettant de se familiariser avec la carte micro:bit. Tous ces exercices pourront être faits avec la carte seule mais aussi avec l'*émulateur en ligne*

1. Activité 1

Analyse

Analysez ce code. Que fait-i l ?

```
1 from microbit import *
2
3 while True:
4     display.show("1")
5     sleep(500)
6     display.show(" ")
7     sleep(500)
```

Faîtes-le fonctionner sur la carte ou le simulateur en ligne.

Exercice 1 :

modifiez le programme ci-dessus afin qu'il compte en boucle de 0 jusqu'à 9

Indication : on pourra utiliser la commande `str(i)` qui transforme le nombre **i** en texte. Ex : **str(5)** renvoie **"5"**

2. Activité 2 : jouer avec les pixels

Analyse

Analysez ce code. Que fait-i l ?

```
1 from microbit import *
2
3 for x in range(5):
4     display.set_pixel(x,0,9)
5     sleep(500)
```

Explications : la fonction set_pixel allume un point sur l'écran. Elle prend 3 paramètres :

- les deux premiers sont l'abscisse et l'ordonnée du point (le point de coordonnées 0,0 étant en haut à gauch de l'écran)
- le dernier paramètre est la luminosité du point entre 0 et 9 : 0 signifie que le point est éteint et 9 est la luminosité maximale.

Faîtes-le fonctionner sur la carte ou le simulateur en ligne.

Exercice 2 :

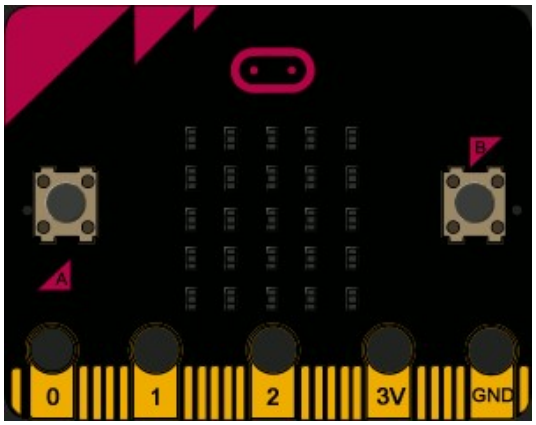
Modifiez ce programme afin qu'il allume la colonne centrale

Exercice 3 :

Modifiez ce programme afin qu'il allume successivement tous les pixels de l'écran.

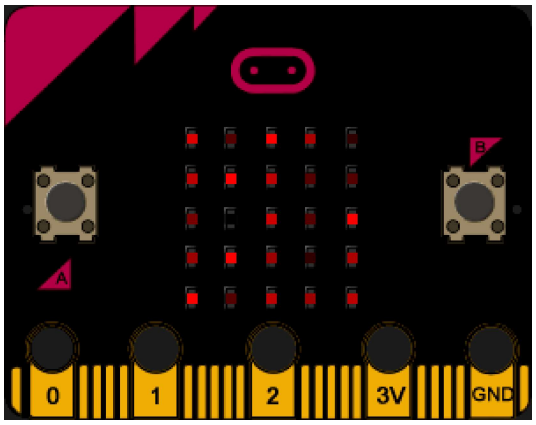
Indication : On pourra utiliser 2 boucles for imbriquées l'une dans l'autre. Pensez à changer le nom de la variable de la seconde boucle !

Exercice 4 :



Modifiez le programme précédent afin qu'il allume tous les pixels colonne par colonne, donc en 5 étapes.

Exercice 5 : le ciel étoilé



Modifiez ce programme afin d'obtenir un affichage avec des pixels dont l'illumination est aléatoire.

Indice : pour obtenir un nombre aléatoire entre 0 et 9 :

- importez la fonction **randint** depuis la librairie **random** : `from random import randint`
- utilisez `randint(0,9)` pour choisir un nombre aléatoire entre 0 et 9

Exercice 6

Analysez la fonction suivante :

```
def carre(l, color):
    for x in range(5-l,l):
        for y in range(5-l,l):
            display.set_pixel(x,y,color)
```

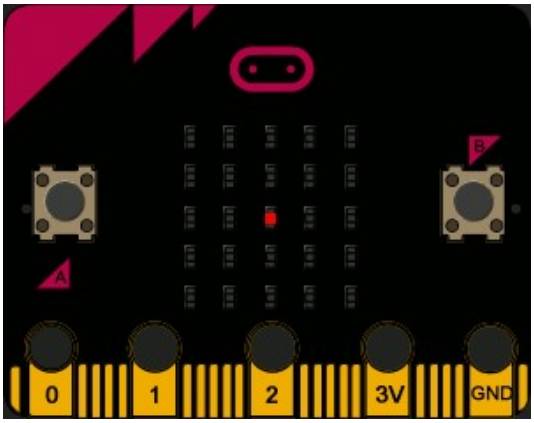
Dessinez sur votre feuille (1 carreau / pixel) l'écran de la carte lorsque l'on invoque

- `carre(3,9)`
- `carre(4,9)`
- `carre(5,9)`

Vous avez donc 3 écrans à représenter.

Une fois le travail réalisé sur papier, vérifiez-le à l'aide d'un programme sur simulateur ou sur la carte.

Exercice 7 :



En utilisant la fonction **carre()** ci-dessus, écrire un programme le plus concis possible réalisant l'animation ci-contre.

3. Activité 3 : jouer avec les boutons

Exercice 8 :

Étudiez le code suivant :

```
from microbit import *

compteur = 0
while True:
    if button_a.was_pressed():
        display.show(str(compteur))
        sleep(1000)
        display.clear()
```

Que fait-il ? vérifiez votre réponse en le testant sur la carte ou le simulateur en ligne.

Exercice 9 :

Modifiez le programme ci-dessus pour

- que le nombre augmente de 1 à chaque appui sur le bouton A.
- que le nombre revienne à 0 lorsqu'on appuie sur le bouton B.
- que le nombre reste affiché en permanence

Exercice 10 :

Étudiez le code suivant :

```
from microbit import *

x = 0
y = 0

while True:
    display.set_pixel(x,y,0)
    if button_a.was_pressed():
        x = x - 1
    if button_b.was_pressed():
        x = x + 1
    x = max(0, min(x, 4))
    display.set_pixel(x,y,9)
    sleep(20)
```

Que fait-il ? vérifiez votre réponse en le testant sur la carte ou le simulateur en ligne.

Expliquez le rôle de la ligne `x = max(0, min(x, 4))`

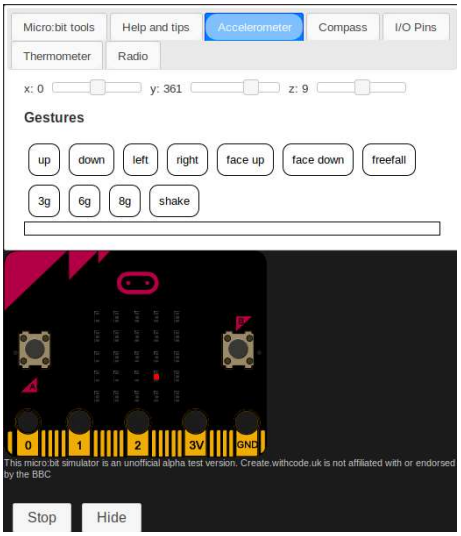
Exercice 11 :

Adaptez le programme ci-dessus afin que le point puisse se déplacer verticalement. On va pour cela utiliser l'accéléromètre de la carte :

- Pour récupérer l'information d'orientation de la carte selon l'axe y, utilisez la ligne `dy = accelerometer.get_y()`
- Si la variable **dy** est plus grande que 600, le point sera sur la 5eme ligne
- Si la variable **dy** est plus grande que 300, le point sera sur la 4eme ligne
- Si la variable **dy** est plus petit que -600, le point sera sur la 1ère ligne
- Si la variable **dy** est plus grande que -300, le point sera sur la 2ème ligne
- sinon, la carte est à peu près horizontale et le point sera su la ligne centrale.

Remarque : Tester le code avec le simulateur

Vous pouvez simuler l'orientation de la carte microbit avec le simulateur en allant dans l'onglet *Accelerometer* puis en déplaçant le curseur *y*. Si vote exercice est correctement réalisé, vous devriez alors voir le point rouge se déplacer verticalement.



Exercice 12 :

Modifiez le programme précédent afin de déplacer le pixel exclusivement avec l'accéléromètre. Selon que l'on penche la carte de droite à gauche, ou d'avant en arrière, le point se déplace sur la matrice LED sans avoir recours aux boutons A et B qui deviennent obsolètes.

4. Activité 4 : Afficher des images

Exercice 13 :

Observez le code suivant. Il réalise une animation.

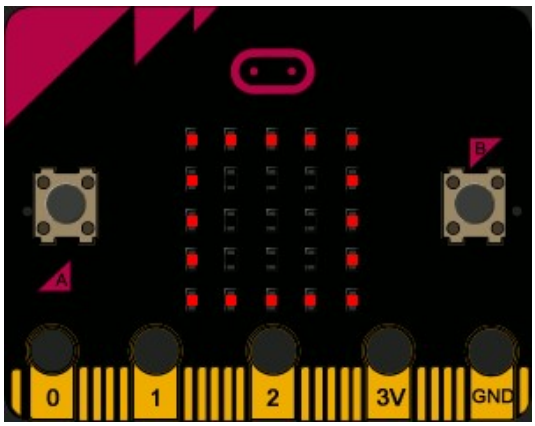
```
from microbit import *

imgs = [
    Image('90000:00000:00000:00000:00000:'),
    Image('90000:09000:00000:00000:00000:'),
    Image('90000:09000:00900:00000:00000:'),
    Image('90000:09000:00900:00090:00000:'),
    Image('90000:09000:00900:00090:00009:')
]
display.show(imgs, delay=500,loop=True)

```

Expliquez comment sont codées les images ?

Exercice 14 : Réalisez vous même votre propre animation



Recréez l'animation ci-contre sur votre carte. Vous remarquerez que les pixels ne sont pas toujours allumés à pleine intensité !

Exercice 15 :

Observez le fonctionnement du code suivant :

```
from microbit import *
from random import choice

mesImages = [Image('00000:00000:00900:00000:00000:'),
        Image('00009:00000:00000:00000:90000:'),
        Image('00009:00000:00900:00000:90000:'),
        Image('90009:00000:00000:00000:90009:'),
        Image('90009:00000:00900:00000:90009:'),
        Image('90009:00000:90009:00000:90009:')]
rolled = choice(mesImages)
display.show(rolled)

```

Remarquez l'emploi de la fonction **choice()** que l'on a importé de la librairie **random**. Elle permet d'extraire une image au hasard depuis la liste *mesImages*.

Modifiez ce programme afin qu'un appui sur le bouton A face rouler le dé.

Exercice 16 : Pierre feuille ciseaux

En utilisant la technique de l'exercice précédant, réalisez un jeu de Pierre-Feuille-Ciseaux. Un appui sur le bouton A affichera une de ces 3 figures au hasard. Vous créerez les images en utilisant la fonction **Image()** comme dans l'exercice précédent.

Mini-projets sur Microbit

II

1. Réaliser un compteur - niveau facile

L'idée dans ce mini projet est de réaliser un simple compteur. Chaque appui sur le bouton A incrémentera le compteur. Un appui sur le bouton B décrémentera le compteur. La problématique réside dans l'affichage du compteur de manière à ce qu'il tienne sur un seul écran de 25 LEDs. Jusqu'à combien peut-on compter ?

Ce mini projet est tout à fait accessible au niveau SNT.

Méthode : Première version

Dans cette version on affiche le compteur sous forme de chiffres. On peut donc compter jusqu'à 9, au delà, l'affichage n'est plus très exploitable...

```
from microbit import *

c=0
while True:
    if button_a.was_pressed():
        c += 1

    if button_b.was_pressed():
        c=0

    display.show(str(c))
```

Méthode : Seconde version

Nous avons 25 LEDs donc il ne doit pas être très difficile de compter jusqu'à 25 ! Voici donc une seconde version un peu plus intéressante car elle introduit deux boucles pour imbriquées. On peut aussi imaginer une variante avec un modulo et une division entière. Il y a donc une variété de solutions et des discussions intéressantes même pour un sujet aussi trivial !

```
from microbit import *

c=0

def affiche(c):
    k=0
    for y in range(5):
        for x in range(5):
            k+=1
            display.set_pixel(x, y, 9 if k<=c else 0)

while True:
```

```
    if button_a.was_pressed():
        c += 1

    if button_b.was_pressed():
        c=0

    affiche(c)
```

Méthode : Troisième version

Chaque LED possède 9 niveaux de couleur. Il n'est pas aisé de distinguer deux niveaux consécutifs mais on peut facilement distinguer entre rien, à moitié allumé et complètement allumé, ce qui fait deux niveaux d'illumination pour 25 LEDs donc un compteur jusqu'à 50 par écran ! Le projet commence à se compliquer...

```
from microbit import *

c=0

def affiche(c):
    k=-1
    for y in range(5):
        for x in range(5):
            k+=2
            if k<c:
                color=9
            elif k==c:
                color=4
            else:
                color=0
            display.set_pixel(x, y, color)

while True:
    if button_a.was_pressed():
        c += 1

    if button_b.was_pressed():
        c=0

    affiche(c)
```

Méthode : Dernière version

Une LED est allumée ou éteinde, donc 1 ou 0... on arrive à la représentation binaire des nombres avec 2^25 soit plus de 33 millions de possibilités ! !

C'est l'occasion d'introduite aux élèves le principe de comptage en binaire. L'algorithme de conversion décimal-binaire ne sera pas exploité ici : on utilisera la fonction **bin()** de MicroPython à cet effet, mais cela peut être une évolution possible pour les plus forts...

```
from microbit import *

c=0

def affiche(c):
    x=0
    y=0
    binaire=bin(c)[2:]
```

```
    nbChiffres = len(binaire)
    display.clear()
    for i in range(nbChiffres):
        digit = binaire[nbChiffres-i-1]
        display.set_pixel(x,y,int(digit)*9)
        x+=1
        if x==5:
            x=0
            y+=1

while True:
    if button_a.was_pressed():
        c += 1

    if button_b.was_pressed():
        c=0

    affiche(c)
```

2. Jeu de Pierre Feuille Ciseaux

Dans ce mini projet réalisable au niveau SNT, on met en oeuvre un jeu de Pierre Feuille Ciseaux sur deux cartes : chaque joueur doit secouer sa carte 3 fois avant que la carte fasse un choix et l'affiche.

Méthode : Première version

Cette première version est assez simple et met en avant les capacités graphiques de la matrice de LEDs.

```
# Olivier Lecluse
# 8 mars 2019

from microbit import *
from random import randint

pierre = Image("00900:"
               "09990:"
               "99599:"
               "09990:"
               "00900")
feuille = Image("99900:"
           "90090:"
           "90009:"
           "90009:"
           "99999")
ciseaux = Image("96009:"
                "69090:"
                "00900:"
                "69090:"
                "96009")

nbSecousses = 0
while True:
    if accelerometer.was_gesture("shake"):
        nbSecousses +=1
        display.show(str(nbSecousses))
    if nbSecousses == 3:
        choix = randint(1,3)
```

```
        if choix == 1:
            display.show(pierre)
        elif choix == 2:
            display.show(feuille)
        elif choix == 3:
            display.show(ciseaux)
        nbSecousses = 0
```

Méthode : Seconde version

Dans cette seconde version, nous allons tricher ! la carte de la victime enverra silencieusement par radio son choix à la carte tricheur qui sera donc assuré de gagner si celui-ci appuie sur les deux boutons A et B à partir de la seconde secousse. La pauvre victime n'y verra que du feu.

La morale de l'histoire est qu'il est toujours intéressant d'avoir accès au code source et qu'il ne faut pas croire aveuglément les boites noires !

Voici la version victime

```
# Olivier Lecluse
# 8 mars 2019
# Shifumi version victime

from microbit import *
from random import randint
import radio
radio.on()
radio.config(group=2)

pierre = Image("00900:"
               "09990:"
               "99599:"
               "09990:"
               "00900")
feuille = Image("99900:"
           "90090:"
           "90009:"
           "90009:"
           "99999")
ciseaux = Image("96009:"
                "69090:"
                "00900:"
                "69090:"
                "96009")

nbSecousses = 0
triche = 1
while True:
    if accelerometer.was_gesture("shake"):
        nbSecousses +=1
        if nbSecousses == 1:
            choix = randint(1,3)
            radio.send(str(choix))
        display.show(str(nbSecousses))
    if nbSecousses == 3:
        if choix == 1:
            display.show(pierre)
        elif choix == 2:
```

```
            display.show(feuille)
        elif choix == 3:
            display.show(ciseaux)
        nbSecousses = 0
```

Et voici la version à flasher sur la carte du tricheur !

```
# Olivier Lecluse
# 8 mars 2019
# shifumi version tricheur

from microbit import *
from random import randint
import radio
radio.on()
radio.config(group=2)

pierre = Image("00900:"
               "09990:"
               "99599:"
               "09990:"
               "00900")
feuille = Image("99900:"
           "90090:"
           "90009:"
           "90009:"
           "99999")
ciseaux = Image("96009:"
                "69090:"
                "00900:"
                "69090:"
                "96009")

nbSecousses = 0
triche = 1
while True:
    incoming = radio.receive()
    if incoming:
        triche = int(incoming)
    if button_a.was_pressed() and button_b.was_pressed():
        choix = triche%3 + 1
    if accelerometer.was_gesture("shake"):
        nbSecousses +=1
        if nbSecousses == 1:
            choix = randint(1,3)
        display.show(str(nbSecousses))
    if nbSecousses == 3:
        if choix == 1:
            display.show(pierre)
        elif choix == 2:
            display.show(feuille)
        elif choix == 3:
            display.show(ciseaux)
        nbSecousses = 0
```