Abdou Mahamadou Zakari Yaou

Compte Rendu Tps 2 INFO501

Introduction :

L'objectif de ce TP est de représenter sur une tablette numérique ou un téléphone fonctionnant avec android des informations issues de capteurs sous la forme d'indicateurs visuels permettant de renseigner très simplement et rapidement sur une situation. La représentation visuelle d'informations est la traduction de l'expression anglophone « visual control » inspirée des méthodes japonaises.

1) Utilisation de pygame

Question 1.a)

Analyser ce programme en traçant sur une feuille de papier le résultat attendu. Exécuter le programme, essayer différents jeux de paramètres (en particulier pour les angles associés à l'arc de cercle).

Explication :

Pygame.init() :initialise pygame

pygame.display.set_mode((800, 600)) : affectation des dimension d'affichage

pygame.time.set_timer(pygame.USEREVENT,200) :temporise les évènements chaque 200ms

self.screen =pygame.display.get_surface() : affecte la surface de l'écran a l'Object screen.

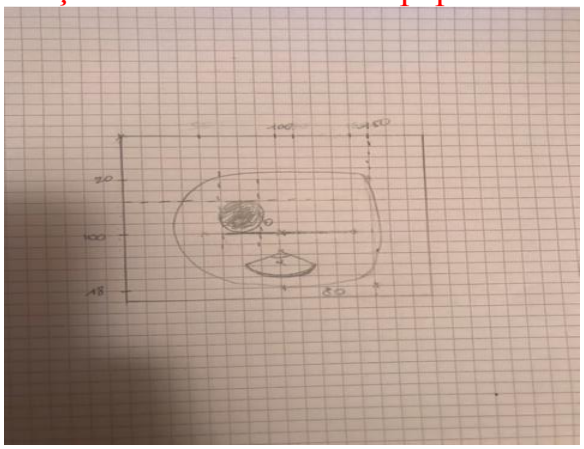
pygame.draw.circle(self.screen, [255,255, 255], [100, 100], 80) : Dessine un cercle de couleur blanche de coordonnees [100,100] correspondant aux nomblre de pixel X,Y et de rayon 80

pygame.draw.ellipse(self.screen, [0,0,0], [50, 60, 15, 20]) : Dessine une ellipse noir ([0,0,0]), de coordonnee [50, 60, 15, 20](coordonnée du point haut gauche du rectangle qui contient l'ellipse(50,60),la largeur du rectangle(15), le hauteur(20))

pygame.draw.arc(self.screen, [0,0,0], [60, 120, 80, 30], 5*math.pi/4, 7*math.pi/4) : meme formule que le précèdent mais avec ajout de deux angles définissant sa forme

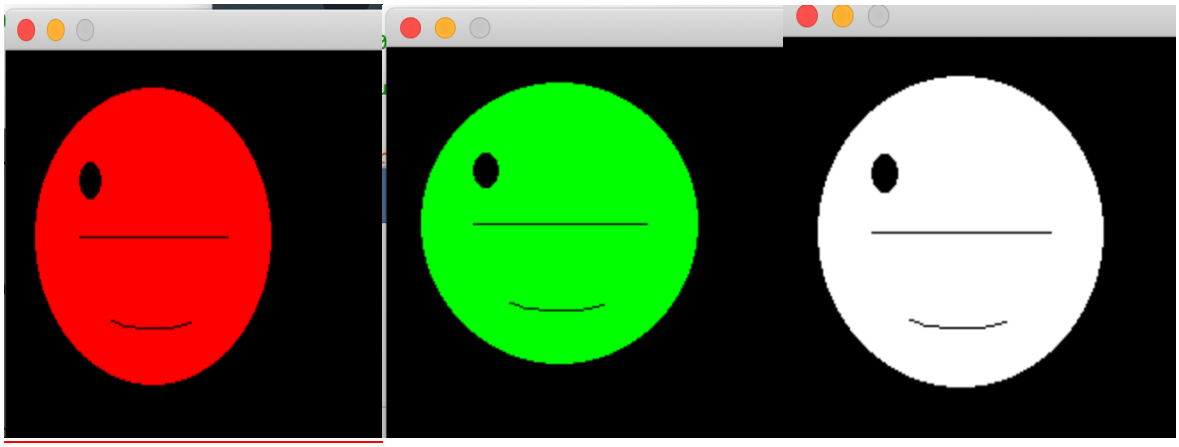
pygame.draw.line(self.screen, [0,0,0], [50,100], [150,100]) : dessine la ligne noire,coordonnees du debut de la ligne([50,100]), coordonnees du debut de la ligne[150,100].

Traçons sur une feuille de papier le résultat attendu :



[Résultat de l'affichage après plusieurs jeux de donnée.]
-Changement de couleur :

```
pygame.draw.circle(self.screen, [0,255, 0], [100, 100], 80)
pygame.draw.circle(self.screen, [255,0, 0], [100, 100], 80)
pygame.draw.circle(self.screen, [255,255, 255], [100, 100], 80)
```



Question1.b)

Afficher, à partir de l'objet generalConfiguration de la fonction main(), la largeur et la hauteur de la zone dessin.
Après un ajout d'un getter dans Q1generalConfiguration :

```
1. def get_screen(self):
2. return self.screen
```

Après on ajoute le code suivant pour l'affichage .

```
1. print(generalConfiguration.get_screen())
```

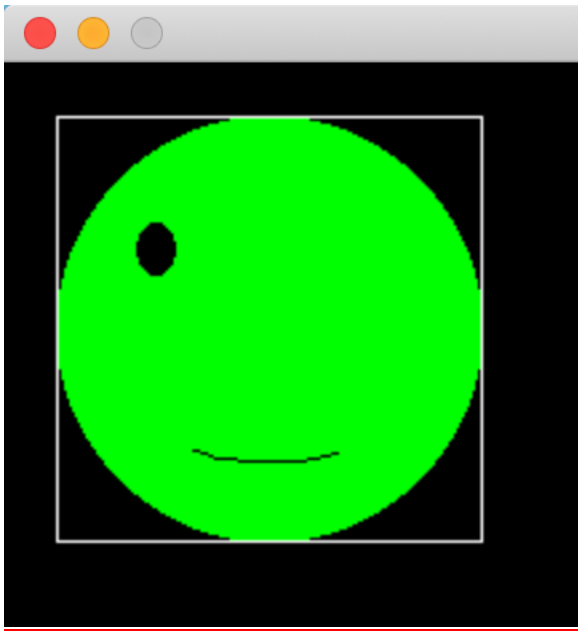
résultat : <Surface(800x600x32 SW)>

Question1.c)
Consulter la documentation de pygame pour utiliser la méthode pygame.draw.rect(). A l'aide de cette méthode, ajouter un cadre blanc d'épaisseur 1 pixel autour du cercle rouge.

[Après la modification de la méthode draw]

```
1. def draw(self):
2. # Draws a circle in red with a center in 100, 100 and
   a radius equal to 80
3. pygame.draw.circle(self.screen, [0,255, 0], [100, 100]
   , 80)
4. # Draws a black ellipse contained in a rectangle whose
    left upper corner is 50,60,
5. # width=15 and height=20
6. pygame.draw.ellipse(self.screen, [0,0,0], [50, 60, 15,
    20])
7. # Draws a black arc contained in a rectangle whose lef
   t upper corner is 60,120,
8. # width=80, height=30, starting angle=5*pi/4, ending a
   ngle=7*pi/4
9. pygame.draw.arc(self.screen, [0,0,0], [60, 120, 80, 30
   ], 5*math.pi/4, 7*math.pi/4)
10.     # Draws a black line starting in position 50,100
   and ending in position 150,100
11.     SIZE = [500,500]
12.     cadrecolor = [255, 255, 255]
13.     rect =[20,20,160,160]
14.     pygame.draw.rect(self.screen, cadrecolor, rect,1)
```

Resultat:



Question1.d)

Ajouter à la classe GeneralConfiguration les getters pour les attributs screen, buttonWidth, buttonHeight, emoticonSize, emoticonBorder. Tester les getters pour vérifier qu'ils retournent bien les valeurs attendues.

Code :

```
1. def get_buttonWidth(self):
2. return self.buttonWidth
3. def get_buttonHeight(self):
4. return self.buttonHeight
5. def get_emoticonSize(self):
6. return self.emoticonSize
7. def get_emoticonBorder(self):
8. return self.emoticonBorder
```

9. Puis dans le main on ajoute :

```
1. generalConfiguration = GeneralConfiguration()
2. print(generalConfiguration.get_emoticonSize())
3. print(generalConfiguration.get_buttonWidth())
4. print(generalConfiguration.get_buttonHeight())
5. print(generalConfiguration.get_emoticonBorder())
6.
```

Résultat :

```
In [1]: runfile('/Users/abdoumahamadouzakariyaou/
Documents/IDU3/INF501-TD-TP/TP2-INF501/EmoticonsV5_EAD
2/Q1/Q1Main.py', wdir='/Users/abdoumahamadouzakariyaou/
Documents/IDU3/INF501-TD-TP/TP2-INF501/EmoticonsV5_EAD
2/Q1')
pygame 2.0.0 (SDL 2.0.12, python 3.7.4)
Hello from the pygame community. https://
www.pygame.org/contribute.html
400
150
80
20
```

Question 2.a) Analyser les fichiers Q2Main.py, Q2GeneralConfiguration.py, Q2Emoticon.py.

2.2) La tête

Q2.b) Ecrire la méthode headToArea(self, position) de la classe Emoticon qui retourne la position, dans le repère de l'écran ($Ox_{écran}y_{écran}$),

Code :

```
1.  def headToArea(self, position):
2.
3.          x=position[0]+self.generalConfiguration.get_screen().get_width()//2
4.          y=self.generalConfiguration.get_screen().get_height()//2-position[1]
5.          return [x,y]
```

test :

```
1.  def main():
2.
3.      # Creates the general configuration and the sensors
4.      generalConfiguration = GeneralConfiguration()
5.
6.      # Creates an emoticon
7.      emoticon = Emoticon()
8.      #------------test Affichage---------------------------------
9.      print(emoticon.headToArea([0,0]))
```

resulat :

```
In [14]: runfile('/Users/abdoumahamadouzakariyaou/
Documents/IDU3/INF501-TD-TP/TP2-INF501/EmoticonsV5_EAD 2/
Q1/Q2Main.py', wdir='/Users/abdoumahamadouzakariyaou/
Documents/IDU3/INF501-TD-TP/TP2-INF501/EmoticonsV5_EAD 2/
Q1')
Reloaded modules: Q1GeneralConfiguration
[400, 300]
```

Q2.c)Gestion de la couleur de la tête de l'émoticône en fonction d'un paramètre x

Code :

```
1.  def color(self, x):
2.          couleur=[]
3.          if x==0:  #jaune
4.              couleur=[255,255,0]
5.          elif -1<=x<0:  # couleur varie entre le rouge et le jaune
6.              couleur=[255,255+255*x,0]
7.          elif 0<x<=1: #couleur varie entre le jaune et le vert
8.              couleur=[-255*x+255,255,0]
```

```
        return couleur

```

test :

```
def main():

    # Creates the general configuration and the sensors
    generalConfiguration = GeneralConfiguration()

    # Creates an emoticon
    emoticon = Emoticon()
    #------------test Affichage---------------------------------
    print(emoticon.color(0))
```

résultat :

```
In [15]: runfile('/Users/abdoumahamadouzakariyaou/
Documents/IDU3/INF501-TD-TP/TP2-INF501/EmoticonsV5_EAD 2/
Q1/Q2Main.py', wdir='/Users/abdoumahamadouzakariyaou/
Documents/IDU3/INF501-TD-TP/TP2-INF501/EmoticonsV5_EAD 2/
Q1')
Reloaded modules: Q2GeneralConfiguration,
Q1GeneralConfiguration, Q2Emoticon
[255, 255, 0]
```

Q2.d) Ecrire la méthode head(self, x) qui dessine la tête colorée

Code :

```
def head(self, x):
        couleur=self.color(x)
        pygame.draw.circle(self.generalConfiguration.screen, couleur, self.headToArea([0,0]), self.generalConfiguration.emoticonSize/2))
 def draw(self):
        self.head(0)
```

teste :

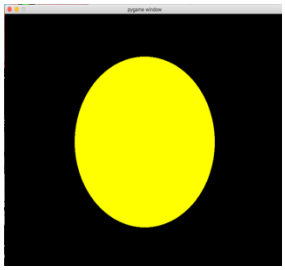
```
def main():

    # Creates the general configuration and the sensors
    generalConfiguration = GeneralConfiguration()

    # Creates an emoticon
```

```
7.      emoticon = Emoticon()
8.      #------------test Affichage---------------------------------
9.      #print(emoticon.headToArea([0,0]))
10.     #print(emoticon.color(0))
11.     emoticon.draw()
12.     emoticon.generalConfiguration.display
```

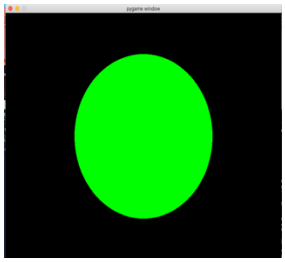
Résultat :



Teste pour paramètre égale à 1

```
1.  def draw(self):
2.          self.head(1)
```

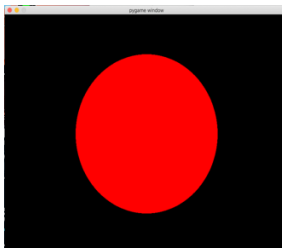
La fonction main reste invariable.
Résultat :



Teste pour paramètre égale à 1

```
1.  def draw(self):
2.          self.head(-1)
```

Résultat :



Q2.e) Ecrire la méthode eye(self, position) qui dessine un oeil noir dont le centre est donné par le
paramètre position

code :

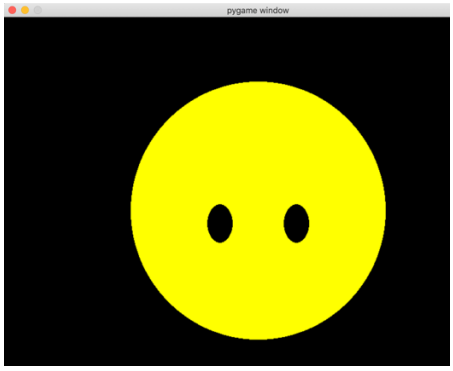
```
1.  def eye(self, position):
```

```
        pygame.draw.ellipse(self.generalConfiguration.screen,[0,0,0],[position[0]-self.eyeWidth/2,position[1]+self.eyeHeight/2,self.eyeWidth,self.eyeHeight])

def draw(self):
        self.head(0)
        self.setEmoticonParameters(400)
        eyeLeftPosition=self.headToArea(self.eyeLeftPosition)
        self.eye(eyeLeftPosition)
        eyeRightPosition=self.headToArea(self.eyeRightPosition)
        self.eye(eyeRightPosition)

```

Le ‘main’ reste invariable
Le résultat est le suivant :



2.4) La bouche

Code :

```
def mouth(self,position,x):
        if x>0.15:
            pygame.draw.arc(self.generalConfiguration.screen,[0,0,0],[position[0]-self.mouthMaxWidth/2,position[1]-self.mouthMaxHeight, self.mouthMaxWidth, self.mouthMaxHeight], math.pi+self.mouthAngle ,2*math.pi-self.mouthAngle)

        elif x<-0.15:
            pygame.draw.arc(self.generalConfiguration.screen,[0,0,0],[position[0]-self.mouthMaxWidth/2,position[1]-self.mouthMaxHeight/2, self.mouthMaxWidth, self.mouthMaxHeight],self.mouthAngle ,math.pi-self.mouthAngle)
        elif -0.15<x<0.15:
            pygame.draw.arc(self.generalConfiguration.screen,[0,0,0],[position[0]-self.mouthMaxWidth/2,position[1]],[position[0]+self.mouthMaxWidth/2,position[1]])
```

Le ‘main’ reste toujours invariable
Résultat :

3) Les boutons

3.1) Position des boutons

Question 3.a)

Ecrire la méthode getPosition(self) de la classe Button qui retourne la position du coin supérieur gauche du bouton sous la forme d'une liste [x, y].

code :

```
1.  def getPosition(self):
2.          return [self.buttonWidth,self.buttonHeight]
```

3.2) Affichage d'un texte

Q3.b) Ecrire la méthode drawLines(self, lines) de la classe Button qui affiche les lignes passés en
Paramètre sous forme de liste, dans le bouton.

Code :

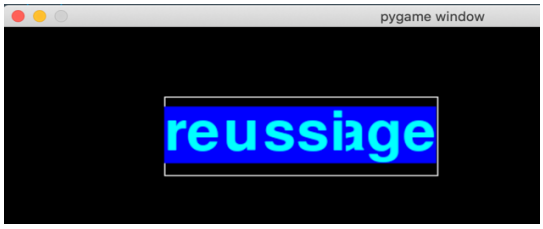
```
1.  def drawLines(self, lines):
2.          screen = pygame.display.get_surface()
3.          # Creates the font
4.          font = pygame.font.Font(None, 80)
5.          # Creates the text image containing « This is a test » written in white
6.          for i in lines:
7.              textImage = font.render(i, 1, [0,255,255],[0,0,255])
8.              # Pastes the image on the screen. The upper left corner is at the positi
    on
9.              screen.blit(textImage,[self.generalConfiguration.buttonWidth,self.genera
    lConfiguration.buttonHeight])
```

Question 3.c) Ecrire la méthode draw(self) de la classe Button qui dessine le bouton sous la forme d'un cadre blanc de 1 pixel qui contienne les informations à afficher.

Code :

```
1.  def draw(self):
2.          SIZE = [150,80]
3.          blanc = [255, 255, 255]
4.          rect =[150,70,255,80]
5.          pygame.draw.rect(self.generalConfiguration.screen, blanc, rect,1)
6.          self.drawLines(['affichage','reussi'])
```

Résultat :



#####A retoucher

4) Gestion des capteurs

Question 4.a)

Utiliser la méthode read() de la classe Sensor pour afficher avec un print la température près du climatiseur dans la salle B204.

Code :

```
1.  print('la temperature de la salle est :'+sensor.read())
```

Resultat :

```
In [1]: runfile('/Users/abdoumahamadouzakariyaou/
Documents/IDU3/INF501-TD-TP/TP2-INF501/
EmoticonsV5_EAD 2/Q1/Q4Main.py', wdir='/Users/
abdoumahamadouzakariyaou/Documents/IDU3/INF501-TD-
TP/TP2-INF501/EmoticonsV5_EAD 2/Q1')
pygame 2.0.0 (SDL 2.0.12, python 3.7.4)
Hello from the pygame community. https://
www.pygame.org/contribute.html
la temperature de la salle est:21.50
```

Q4.b) Afficher avec un print, le label associé à ce capteur.

Code :

```
1.  print('le label associé est:'+sensor.getLabel())
```

Resultat :

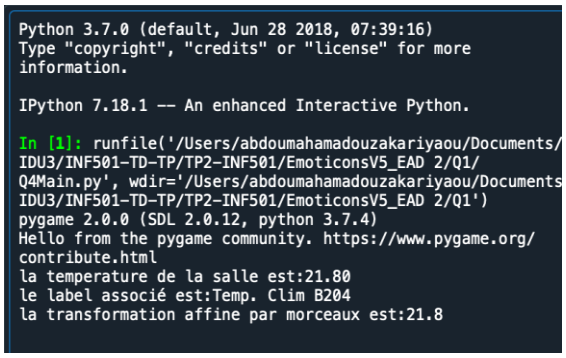
```
In [1]: runfile('/Users/abdoumahamadouzakariyaou/
Documents/IDU3/INF501-TD-TP/TP2-INF501/
EmoticonsV5_EAD 2/Q1/Q4Main.py', wdir='/Users/
abdoumahamadouzakariyaou/Documents/IDU3/INF501-TD-
TP/TP2-INF501/EmoticonsV5_EAD 2/Q1')
pygame 2.0.0 (SDL 2.0.12, python 3.7.4)
Hello from the pygame community. https://
www.pygame.org/contribute.html
la temperature de la salle est:21.80
le label associé est:Temp. Clim B204
```

Q4.c) Ecrire la méthode getTransformedValue(self) qui lit le capteur et réalise la transformation affine par morceaux de la mesure.

Code :

```
1.  # Gets the transformed value
2.      def getTransformedValue(self):
3.          valeurcapteur_float=eval(self.read())
4.          if self.thresholds[0]<valeurcapteur_float<=self.thresholds[1]:
5.              valeurReduite=0.5*valeurcapteur_float-self.thresholds[1]*0.5
6.          elif valeurcapteur_float<=self.thresholds[0]:
7.              valeurReduite=-1.0
8.          elif valeurcapteur_float>=self.thresholds[2]:
9.              valeurReduite=1.0
10.         elif self.thresholds[1]<valeurcapteur_float<self.thresholds[2]:
11.             valeurReduite=valeurcapteur_float/3-self.thresholds[1]/3
12.         return valeurcapteur_float
```

Résultat :



5) Mise en œuvre de l'application

Q5.a) Analyser les fichiers fournis dans le dossier Q5. Dans la fonction main() du fichier Q5Main.py, on a ajouté un capteur à l'objet generalConfiguration à l'aide de la méthode
10
GeneralConfiguration
Sensor
Button Emoticon
addSensor(). Expliquer le fonctionnement de cette méthode. Que vaut l'attribut sensors de la classe
GeneralConfiguration après cet ajout ?

La méthode addSensor permet d'ajouter à la liste « Sensors » des des Sensor qui contient,un id,un Émoticon et un buttons

```
def addSensor(self, sensor):
    sensor.setGeneralConfiguration(self)
    sensor.setSensorId(len(self.sensors))
    sensor.setEmoticon(Emoticon(sensor))
    sensor.setButton(Button(sensor))
    self.sensors.append(sensor)
```

Q5.b) Compléter la classe GeneralConfiguration du fichier Q5GeneralConfiguration.py avec les getters de la question Q1.4. Ajouter les getters pour les attributs sensors et selectedSensor.

C'est fait

Q5.c) Compléter la classe Emoticon du fichier Q5Emoticon.py avec les méthodes de la question Q2. On pourra supprimer la méthode setGeneralConfiguration(self, generalConfiguration) qui n'est plus utile.

C'est fait

Q5.d) Modifier la méthode draw(self) de la classe GeneralConfiguration pour qu'elle appelle la méthode drawEmoticon(self) de la classe Sensor.

C'est fait

Q5.e) Modifier la méthode drawEmoticon(self) de la classe Sensor pour qu'elle appelle la méthode draw(self) de la classe Emoticon. Tester le programme.

C'est fait

Q5.f) Compléter la classe Button du fichier Q5Button.py avec les méthodes de la question Q3. On pourra supprimer la méthode setGeneralConfiguration(self, generalConfiguration) qui n'est plus utile.

C'est fait

Q5.g) Modifier la méthode draw(self) de la classe GeneralConfiguration pour qu'elle appelle la méthode drawButton(self) de la classe Sensor, en plus de la méthode drawEmoticon(self).

C'est fait

Q5.h) Modifier la méthode drawButton(self) de la classe Sensor pour qu'elle appelle la méthode draw(self) de la classe Button. Tester le programme.

C'est fait

5.2) Prise en compte de la sélection

Q5.i) Pour savoir si un capteur est sélectionné, écrire la méthode positionToSensorId(self, position) dans la classe GeneralConfiguration qui retourne le numéro du capteur si le paramètre position est dans une position dans la zone du bouton et None sinon.