Travaux pratiques - Programmation Orientée Objet (POO)

Exercice 1 Implémentez dans votre éditeur Python le code la classe `Player` que vous avez écrit lors de l'exerccie 6 du TD. Vous reprendrez l'exemple d'utilisation de l'exercice afin de tester votre implémentation.

Exercice 2 Reprendre les classes `Piece` et `Appartement` de l'exercice 3 du TD.

1. Écrire les méthodes constructeurs des deux classes `Piece` et `Appartement`.
2. Finaliser la classe `Piece` : Écrire les méthodes accesseurs et mutateurs de la classe `Piece`.
3. Finaliser la classe `Appartement`.
   (a) Écrire la méthode `ajouter(self,piece)` qui ajoute une pièce à la liste de pièces présentes dans l'appartement.
   (b) Écrire la méthode qui permet de retourner le nombre de pièces présentes dans l'appartement : `nbPieces(self)`.
   (c) Écrire la méthode `getSurfaceTotale(self)`, qui renvoie la surface totale de l'appartement.
   (d) Écrire la méthode `getListePieces(self)`, qui renvoie la liste des pièces de l'appartement.

```
class Piece:
  # nom est une string et surface est un float
  def __init__(self,nom,surface):
    self.nom=nom
    self.surface=surface

  # Accesseurs: retournent les attributs d'un objet de cette classe
  def getSurface(self):
    return self.surface

  def getNom(self):
    return self.nom

  # Mutateur
  def setSurface(self,s): # s est un float,
    self.surface = s

class Appartement:
  # nom est une string
  def __init__(self,nom):
    # L'objet est une liste de pieces (objets issus de la classe Piece)
    self.listeDePieces=[]
    self.nom=nom

  def getNom(self):
    # Accesseurs:
    return self.nom

  # pour ajouter une piece de classe Piece
  def ajouter(self,piece):
    self.listeDePieces.append(piece)

  # pour avoir le nombre de pieces de l'appartement
  def nbPieces(self):
    return len(self.listeDePieces)

  # retourne la surface totale de l'appartement (un float)
  def SurfaceTotale(self):
    total=0
        for piece in self.listeDePieces:
          surf=piece.getSurface()
      total=total+surf

  # retourne la liste des pieces avec les surfaces
  def getListePieces(self): # sous forme d'une liste de tuples
    L=[]
        for piece in self.listeDePieces:
                surf=piece.getSurface()
                nom=piece.getNom()
                L.append((nom,surf))
        return L
```

Vous pourrez tester votre implémantation à l'aide du programme principal suivant :

```
a=Appartement('appt25')
p1=Piece("chambre", 11.1)
p2=Piece("sdbToilettes", 7)
p3=Piece("cuisine", 7)
p4=Piece("salon", 21.3)
print(p4.getNom(),p4.getSurface())
p1.setSurface(12.6)
a.ajouter(p1)
a.ajouter(p2)
a.ajouter(p3)
a.ajouter(p4)
print(a.getNom(),a.getListePieces())
print('nb pieces =', a.nbPieces(),', Surface totale =',a.SurfaceTotale())
```

Votre programme pincipal devra retourner les résultats suivants :

```
salon 21.3
appt25 [('chambre', 12.6), ('sdbToilettes', 7), ('cuisine', 7),
('salon', 21.3)]
nb pieces = 4 , Surface totale = 47.9
```

Exercice 3 Les balles rebondissantes

Prise en main de Pygame

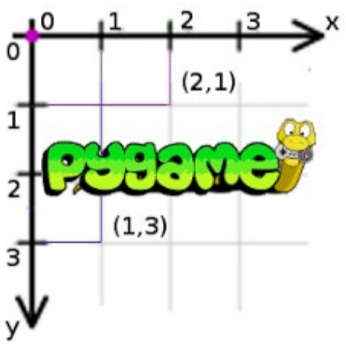
Pygame est une bibliothèque facilitant le développement de jeux vidéo en temps réel avec le langage Python :

- Article Wikipédia : https ://fr.wikipedia.org/wiki/Pygame
- Site Web : https ://www.pygame.org/news
- Dépôt GitHub avec le code source : https ://github.com/pygame/pygame/

Question 1 : Voici le code minimal permettant de créer une fenêtre Pygame, la maintenir ouverte jusqu'au clic sur le bouton de fermeture de la fenêtre. Testez ce code.

```
import pygame, sys
import time

# Initialisation de la fenetre pygame

LARGEUR = 640
HAUTEUR = 480

pygame.display.init()

# pour definir les dimensions de la fenetre
fenetre = pygame.display.set_mode((LARGEUR, HAUTEUR))

fenetre.fill([0,0,0]) # pour colorier le fond avec une couleur

# Boucle infinie pour afficher la fenetre et son contenu
continuer = True
while continuer:
        # routine pour pouvoir quiter la boucle while
        for event in pygame.event.get():
        if event.type == pygame.QUIT: # lorsqu'on clique sur la croix de la fenetre
        continuer = False

# Fermeture de la fenetre (qui a donc lieu si on quitte la boucle while)
pygame.display.quit()
sys.exit()
```

Attention, dans Pygame, l'axe des ordonnées est orienté vers le bas. De ce fait, le pixel en haut à gauche a pour coordonnées (0, 0).



Ajout d'une balle

On va maintenant ajouter une balle dans notre fenêtre !
Question 2 : Analysez et testez ce code. Modifiez-le de façon à créer une balle de rayon 20 (pixels) qui sera positionnée au centre de la fenêtre Pygame.

```
import pygame, sys
import time

# Initialisation de la fenetre pygame

LARGEUR = 640
HAUTEUR = 480

pygame.display.init()

fenetre = pygame.display.set_mode((LARGEUR, HAUTEUR))

fenetre.fill([0,0,0])

# Creation d'une balle
rayon = 20
x = LARGEUR // 2
y = HAUTEUR // 2
couleur = (45, 170, 250)

# Boucle infinie pour afficher la fenetre et son contenu
continuer = True
while continuer :
        # construction de la balle dans la fenetre
        pygame.draw.circle(fenetre,couleur,(x,y),rayon)

        # mise a jour de la fenetre
        pygame.display.update()

        # routine pour pouvoir quiter la boucle while
        for event in pygame.event.get():
                if event.type == pygame.QUIT:
                continuer = False

# Fermeture de la fenetre (qui a donc lieu si on quitte la boucle while)
pygame.display.quit()
sys.exit()
```

Faire bouger la balle

C'est assez simple de faire bouger la balle dans la fenêtre à chaque tour de boucle while. Pour cela :

- on va dessiner une nouvelle balle à chaque tour de boucle
- cette nouvelle balle sera "décalée" par rapport à la position précédente : on utilise les variables dx et dy pour définir les valeurs de déplacement
- il ne faut pas oublier de recolorer le fond au début de chaque tour de boucle, pour que les balles aux instants précédents n'apparaissent plus dans la fenêtre
- on définit un temps de pause à la fin de chaque tour de boucle pour que le déplacement ne soit pas trop rapide.

Question 3 : Analysez le code suivant et déterminez les coordonnées du centre de la balle à l'issue du 3ème tour de boucle (papier-crayon !). Expliquez !
Testez le code pour bien comprendre le rôle des variables `dx` et `dy` que l'on appellera les variables de vitesse dans la suite.

```
import pygame, sys
import time

# Initialisation de la fenetre pygame

LARGEUR = 640
HAUTEUR = 480
pygame.display.init()
fenetre = pygame.display.set_mode((LARGEUR, HAUTEUR))
fenetre.fill([0, 0, 0])

# Creation d'une balle
rayon = 20
x = LARGEUR // 2
y = HAUTEUR // 2
couleur = (45, 170, 250)
dx = 4 # vitesse en abscisse
dy = -3 # vitesse en ordonnee

# Boucle infinie pour afficher la fenetre et son contenu
continuer = True
while continuer :
        # couleur du fond reinitialisee
        fenetre.fill([0, 0, 0])

        # construction de la balle dans la fenetre
        pygame.draw.circle(fenetre,couleur,(x,y),rayon)

        # deplacement de la balle
        x = x + dx
        y = y + dy

        # mise a jour de la fenetre
        pygame.display.update()

        # routine pour pouvoir quiter la boucle while
        for event in pygame.event.get():
                if event.type == pygame.QUIT:
                        continuer = False
        # temps de pause a ajuster
        time.sleep(0.1)

# Fermeture de la fenetre (qui a donc lieu si on quitte la boucle while)
pygame.display.quit()
sys.exit()
```

Coordonnées de la balle :
Initialisation : (LARGEUR // 2 ;HAUTEUR //2) = (320 ;240)
Itération 1 : (320 + 4 ;240 - 3) = (324 ;237)
Itération 2 : (324 + 4 ;237 - 3) = (328 ;234)
Itération 3 : (328 + 4 ;234 - 3) = (332 ;231)

Ajout d'un rebond sur les parois

Question 4 : Modifiez le code précédent pour que la balle rebondisse sur chaque paroi. Pour cela, il faut modifier intelligemment les variables de vitesse `dx` et `dy`.

Ajout d'une deuxième balle

Question 5 : Ajoutez une deuxième balle (qui devra également rebondir sur chaque paroi).

Gestion de la collision entre les deux balles

Question 6 : On veut désormais gérer les collisions entre les deux balles.

1. À l'aide d'un schéma (papier-crayon !), mettez en évidence le test devant être réalisé pour détecter une collision.
2. Implémentez ce test et affichez la chaîne "collision" en console lorsque les deux balles se touchent.
3. Pour l'illusion du rebond, échangez les valeurs respectives de dx et dy pour les deux balles.

Correction question 4, 5 et 6

```
import pygame, sys
import time
import math
import pygame

LARGEUR = 640
HAUTEUR = 480
RAYON = 40

pygame.init()
pygame.display.init()
fenetre = pygame.display.set_mode((LARGEUR, HAUTEUR))
fenetre.fill([0,0,0])

# balle 1
x = 300
y = 200
dx = 2
dy = 0
couleur = (45,170,250)

#balle 2
x2 = 100
y2 = 100
dx2 = -3
dy2 = 1
couleur2 = (157,11,1)

police = pygame.font.SysFont('monospace',15)
image_texte = police.render ( 'Collision !!!!', 1 , (255,0,0) )
while True :
    fenetre.fill([0,0,0])
    pygame.draw.circle(fenetre,couleur,(x,y),RAYON)
    pygame.draw.circle(fenetre,couleur2,(x2,y2),RAYON)

#mouvement de la balle
    x += dx
    y += dy

#rebond
    if (y <= RAYON) or (y >= HAUTEUR - RAYON):
       dy = -dy
    if (x <= RAYON) or (x >= LARGEUR - RAYON):
        dx = -dx

    x2 += dx2
    y2 += dy2

    if (y2 <= RAYON) or (y2 >= HAUTEUR - RAYON):
       dy2 = -dy2
    if (x2 <= RAYON) or (x2 >= LARGEUR - RAYON):
        dx2 = -dx2

#gestion des collisions
    if (math.sqrt((x-x2)*(x-x2) + (y-y2)*(y-y2))) < 2*RAYON:
        dx = -dx
        dy = -dy
        dx2 = -dx2
        dy2 = -dy2

    pygame.display.update()
    #Routine pour pouvoir fermer proprement la fenetre Pygame
    for event in pygame.event.get():
```

Ajout d'une troisième balle et gestion du rebond avec les deux autres

À vous jouer ! Et vous irez jusqu'à 100 balles ! ... non c'est une blague car cela devient vraiment long de procéder ainsi.

En revanche, la POO va nous venir en aide pour générer 100 balles sans aucun problème !

La POO à la rescousse

Création d'une classe `Balle`

Question 7 : Créez une classe `Balle` qui initialisera chaque balle avec des valeurs aléatoires pour l'abscisse, l'ordonnée, les vitesses, la couleur. On souhaite que toutes les balles aient le même rayon. De plus, cette classe devra posséder deux méthodes `dessiner` et `bouger` qui permettent respectivement de dessiner la balle et de la faire bouger dans la fenêtre.

```
import pygame, sys
import time
import math
import pygame
import random
#from pygame.locals import *

LARGEUR = 640
HAUTEUR = 480
RAYON = 5

######################  CLASSE BALLE ##########################
class Balle: #definition de la classe
    "Une balle avec une vitesse et une position initialises aleatoirement"
    def __init__(self): #constructeur  de la classe
        self.x = random.randrange(LARGEUR) #attribut : abscisse e de la balle
        self.y = random.randrange(HAUTEUR) #attribut : ordonnee de la balle
        self.dx = random.randrange(-5,5)  #attribut : vitesse de deplacement en
           abscisse de la balle
        self.dy = random.randrange(-5,5)  #attribut : vitesse de deplacement en
           ordonnee de la balle
        self.couleur=(random.randrange(256),random.randrange(256),random.randrange
           (256)) #attribut : couleur de la balle
        self.collision = False

    def dessiner(self):
        pygame.draw.circle(fenetre,self.couleur,(self.x,self.y),RAYON)

    def bouger(self):
        if (self.collision == True):
            self.dx = -(self.dx)
            self.dy = -(self.dy)
            self.collision = False
        #rebond
        if (self.y <= RAYON) or (self.y >= HAUTEUR - RAYON):
            self.dy = -(self.dy)
        if (self.x <= RAYON) or (self.x >= LARGEUR - RAYON):
            self.dx = -(self.dx)
        self.x += self.dx
        self.y += self.dy


    def setCollision(self,valeur):
        self.collision = valeur

    def getCollision(self):
        return self.collision
```

Question 8 : Modifiez le code de la question 4 pour faire rebondir une balle dans la fenêtre, la balle étant une instance de la classe `Balle` créée à la question précédente.

Création d'une classe `SacDeBalles`

Question 9 : créez une classe `SacDeBalles` dont la méthode `__init__(self, nb_balles)` définit un attribut de la classe qui est une liste de `nb_balles` balles.

```
class SacDeBalle: #definition de la classe
    def __init__(self,nbBalles):
        self.liste=[]
        self.nbBalles = nbBalles
        for i in range(nbBalles):
            self.liste.append(Balle())

    def afficher(self):
        for b in self.liste:
           b.dessiner()

    def animer(self):
        for b in self.liste:
           b.bouger()


    def collision(self):
        for b in self.liste:
            for b1 in self.liste:
                if (b1 !=b):
                    d = math.sqrt((b1.x-b.x)*(b1.x-b.x) + (b1.y-b.y)*(b1.y-b.y))
                    if d < 2*RAYON:
                       # print(d)
                        b1.setCollision(True)
                        b.setCollision(True)
```

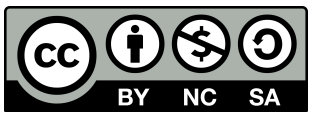
Exemple : L'instruction **mon_sac_de_balles = SacDeBalles(20)** doit créer un sac contenant 20 balles.

Question 10 : Modifiez le code de la question 8 pour afficher et animer toutes les balles de laliste `mon_sac_de_balles`. Vous devez créer une méthode `afficher` qui affiche toutes les balles du sac de balles et une méthode `animer` qui fait bouger toutes les balles du sac.

Question 11 : Créer une méthode collision dans la classe `SacDeBalles` qui détermine si deux balles sont entrée en collision.

Question 12 :Si ce n'est pas déjà fait, modifiez/complétez la méthode `bouger` afin qu'elle gère les collisions d'une balle (self) avec toutes les autres.

```
###################### PROGRAMME PRINCIPALE ########################
pygame.init()
pygame.display.init()
fenetre = pygame.display.set_mode((LARGEUR, HAUTEUR))
fenetre.fill([255,255,255])

sac = SacDeBalle(100)
police = pygame.font.SysFont('monospace',15)
image_texte = police.render ( 'Collision !!!!', 1 , (255,0,0) )
while True :
    fenetre.fill([255,255,255])
    sac.afficher()
    sac.collision()
    sac.animer()

    pygame.display.update()

    # routine pour pouvoir fermer &proprement la fenetre Pygame
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.display.quit()
            sys.exit()

    time.sleep(0.02)
```



Source : www.math93.com