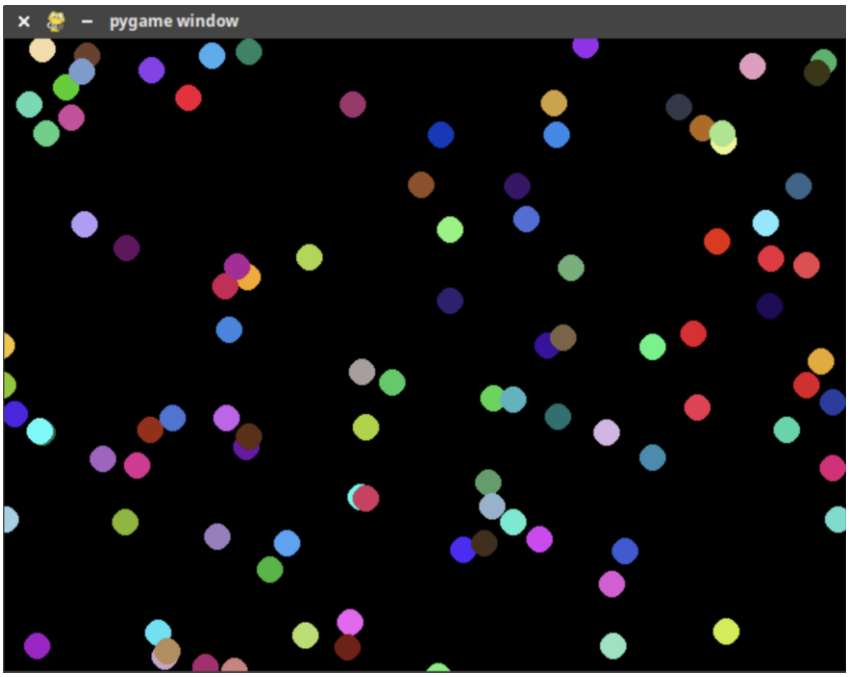
Mini projet Poo

A faire en groupe de 2

I. Prise en main de Pygame



Voici le code de départ que vous trouverez dans le fichier joint :

```
import pygame, sys
import time
from pygame.locals import *


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

x = 300
y = 200
dx = 4
dy = -3
couleur = (45,170,250)

while True :
    fenetre.fill([0,0,0])
    pygame.draw.circle(fenetre,couleur,(x,y),10)

    x += dx
    y += dy

    pygame.display.update()

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

```
time.sleep(0.1)
```

1) Rajout d'un rebond sur les parois :

Modifiez le code précédent afin que la balle rebondisse sur chaque paroi (il suffit de modifier intelligemment les variables de vitesse dx et dy)

2) Rajout d'une deuxième balle :

Attention au nommage des variables...

3) Gestion de la collision entre les deux balles :

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

4) Rajout d'une troisième balle et gestion du rebond avec les deux autres :

... vraiment ? Peut-on continuer comme précédemment ?

II – La POO à la rescousse : création d'une classe Balle

L'objectif est que la méthode constructeur dote chaque nouvelle balle de valeurs aléatoires : abscisse, ordonnée, vitesse, couleur...
Créez cette classe et instanciez une balle.

Puis plusieurs balles ! (qui se collisionnent...)

Pour bien commencer :

Il va falloir créer une classe Ball. Cette classe devra être instancié plusieurs fois au début du programme, et toutes ces instances seront affichés dans la boucle principale.

Votre classe balle devra implémenter

- Un constructeur attribuant une position et une direction de départ à la balle
- Une méthode draw() qui permet de l'afficher à l'écran
- Une méthode move() qui permet de la déplacer selon sa direction actuelle, et gérer les bordures

Et plus tard,

- Une méthode check_collide(ball) vérifiant si la balle vient d'entrer en collision avec 'ball', qui est une autre balle. Gérer alors sa nouvelle direction.

À rendre :

Chaque groupe devra rendre :

- Le fichier python correctement nommé dans le casier ou sur Github (-1 point par jour de retard)
- Un court rapport en PDF décrivant
  - Vos difficultés/Facilités & que vous avez compris
  - Ce que vous n'avez pas réussi à faire.
  - Les implémentations optionnelles que vous auriez pu faire.

Barème envisagé :

Critères	**Points**
Respect des consignes (noms des fonctions, affichages demandés, ...)	/ 2
Respect de la norme PEP8 (google)	/ 1
Utilisation correcte des listes	/ 2
Noms des variables parlants	/ 1
Boucle principale correcte	/ 2
Méthodes, attributs & constructeur corrects	/ 3
Commentaires intelligents et documentation de la fonction	/ 1
Version Graphique tourne correctement	/ 2
Un code compréhensible ; les parties doivent être bien séparées et clairement commentées	/ 1
Originalité : Bonus	/42
Total	/ 15