



# Optimisation des mouvements de foule lors d'évacuations d'urgence

Numéro d'inscription :

1



Lien Avec Le Thème :

Ville

Source <https://www.algerie-dz.com/forums/international/8567627-irak-au-moins-un-mort-et-60-blesses-suite-a-une-bousculade-dans-un-stade-de-football>

Corée du Sud 151 morts et plus de 76 blessés dans une bousculade à Séoul



Source :<https://lareleve.ma/69298/>



## Problématique

-Comment gérer efficacement les mouvements de foule lors d'évacuations d'urgence pour éviter des accidents tragiques ?

# Plan

A- 1<sup>er</sup> Tentative :

B-2<sup>nd</sup> Tentative :

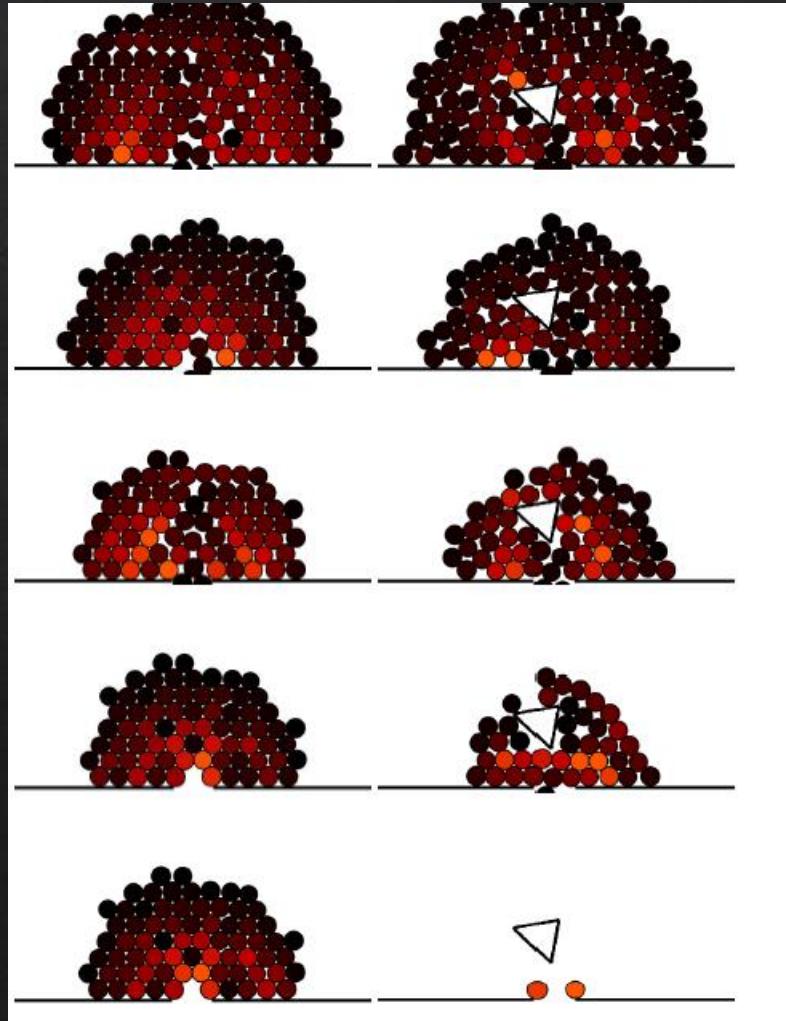
C-Comparaison :

D-Conclusion :

A

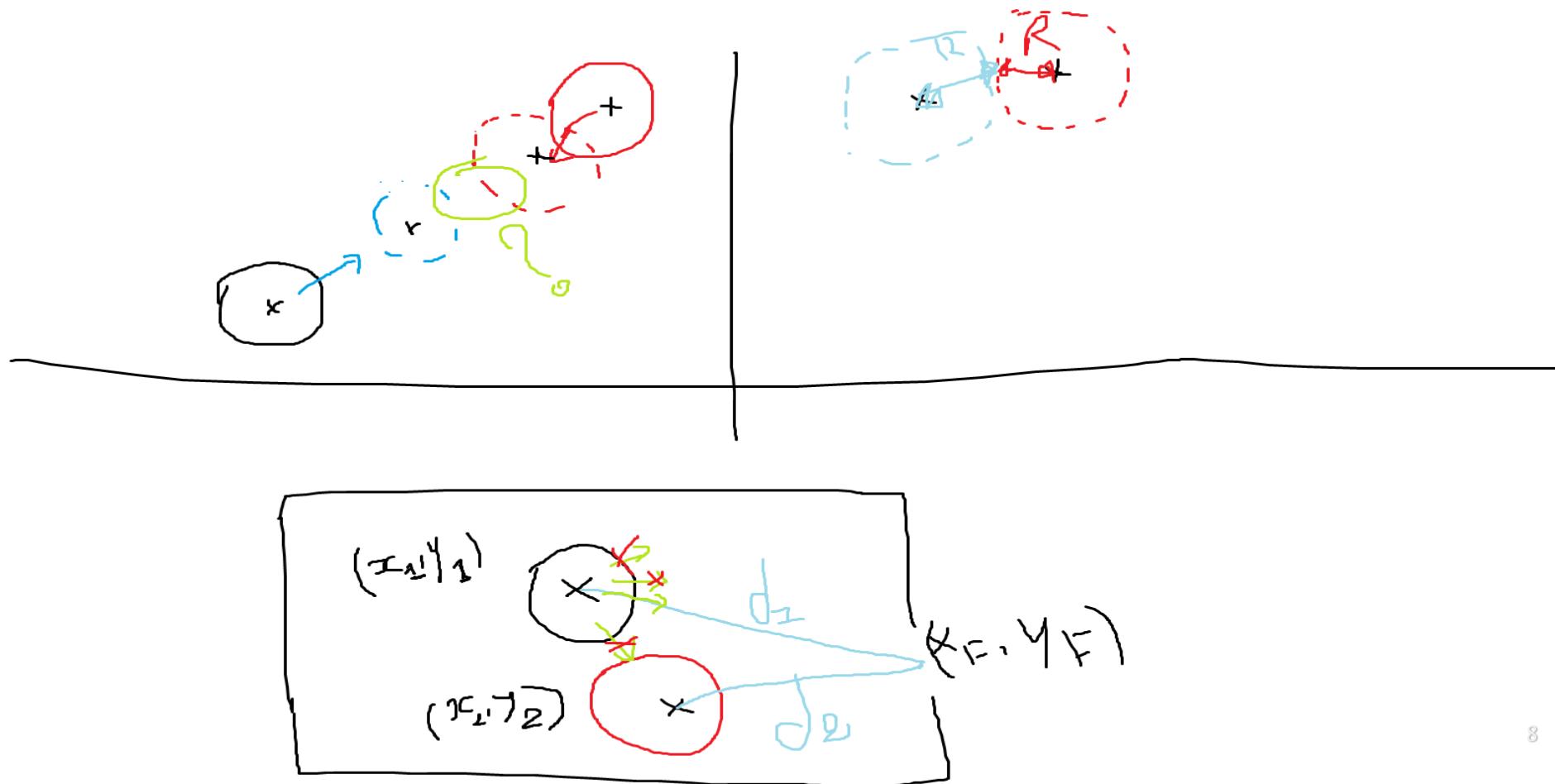
- Tentative 1

# Inspiration du modèle granulaire



Source : « Crowds In Equations » \_ B. Maury & S. Faure page 78

# Comment modéliser une foule en 2D ?



Comment modéliser une foule en 2D ?



Pygame : est une bibliothèque Python permettant de créer des simulations interactives.



# Premiers pas Pygame

The image shows a YouTube thumbnail for a tutorial video. The thumbnail has a light beige background with a green dotted border. On the left, there's a large black number '1' above the text 'PYGAME PROGRAMMING TUTORIAL'. Below that is the subtitle 'LEARN TO MAKE GAMES WITH PYTHON'. In the top right corner of the thumbnail is a black icon of a game controller. The main title 'Pygame Tutorial #1 - Basic Movement and Key Presses' is centered at the top in a large, bold, black font. Below it, the text '1 M de vues • il y a 5 ans' indicates views and upload date. To the left of the channel name is a small orange circular profile picture of a person. The channel name 'Tech With Tim' is next to it, followed by a verified checkmark. A short description below the channel info reads: 'This is a new series on my channel where I am going to be going through the pygame module in python. Pygame is used to make ...'. There are two buttons at the bottom: 'Sous-titres' (Subtitles) and a play button icon with the text '12:32' indicating the video duration. The video player interface below the thumbnail shows the same title and description, along with a progress bar showing 'showing basic movement | give our window a caption | start by checking for events | draw a...', a '5 moments' button with a dropdown arrow, and the number '10' in the bottom right corner.

File Edit Format Run Options Window Help

```
import pygame
pygame.init()
win = pygame.display.set_mode((500, 500))
pygame.display.set_caption("Premier Jeu")
x,y=50,50
run = True
# Boucle principale
while run:
    # "Horloge" pour éviter que les choses ne se déroulent trop vite
    pygame.time.delay(50)
    # Événements (tout ce qui se passe de la part de l'utilisateur)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False
    # Mouvement
    keys = pygame.key.get_pressed()
    # Déplacement horizontal
    if 0 <= x <= 460:
        if keys[pygame.K_LEFT]:
            x -= 10
        if keys[pygame.K_x]:
            x -= 20
        if keys[pygame.K_RIGHT]:
            x += 10
        if keys[pygame.K_z]:
            x += 20
    elif x > 460:
        x = 460
    else:
        x = 0
    win.fill((0, 0, 0)) # Effacer la fenêtre
    pygame.draw.circle(win, (255, 0, 0), (x, y), 1)
    pygame.display.update() # Rafraîchir la fenêtre
pygame.quit()
```

Premier Jeu



File Edit Shell Debug Options Window Help

Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.

&gt;&gt;&gt;

= RESTART: C:\Users\rzeig\Desktop\Cours\8.Tipe\2.Atelier\0.Premier\_Pas\_Python\Es

4.0 (SDL 2.26.4, Python 3.11.2)  
m the pygame community. https://www.pygame.org/contribute.html

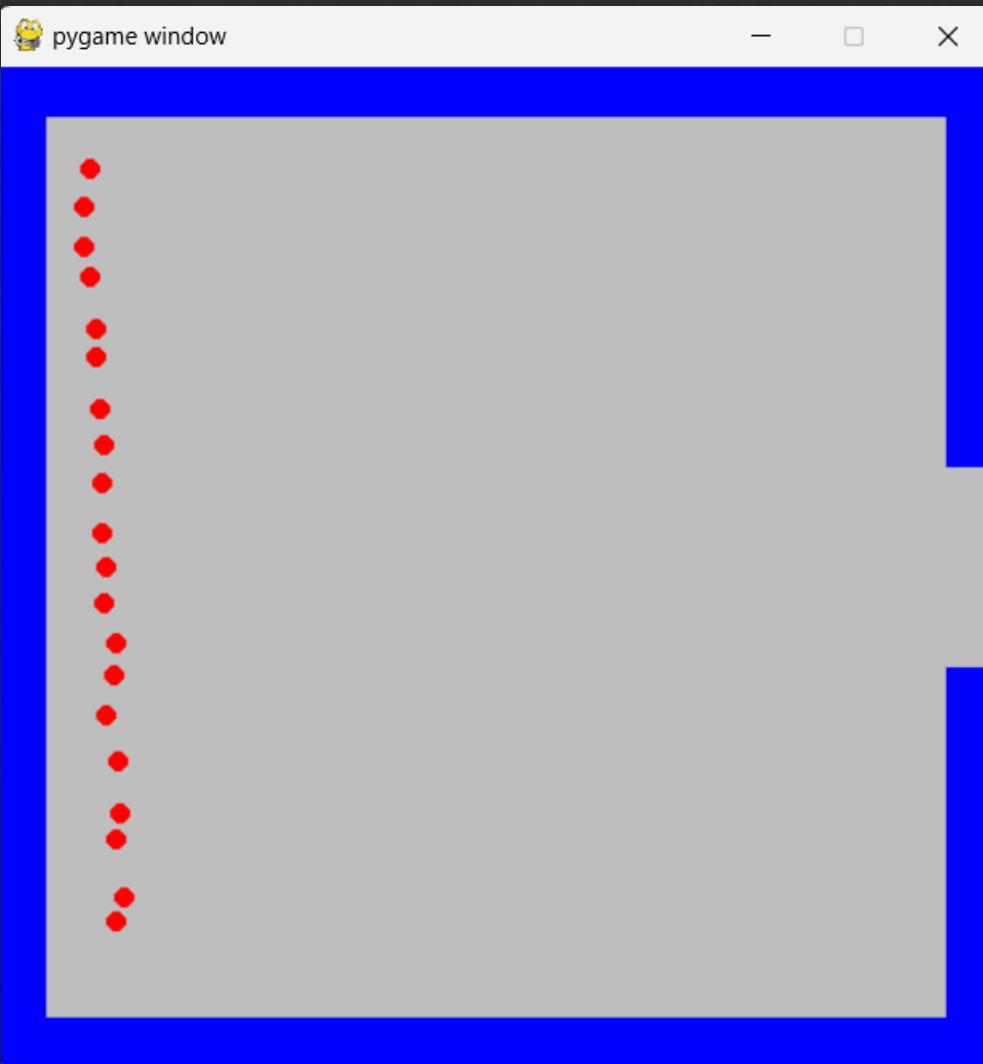
Ln: 7 Col: 0

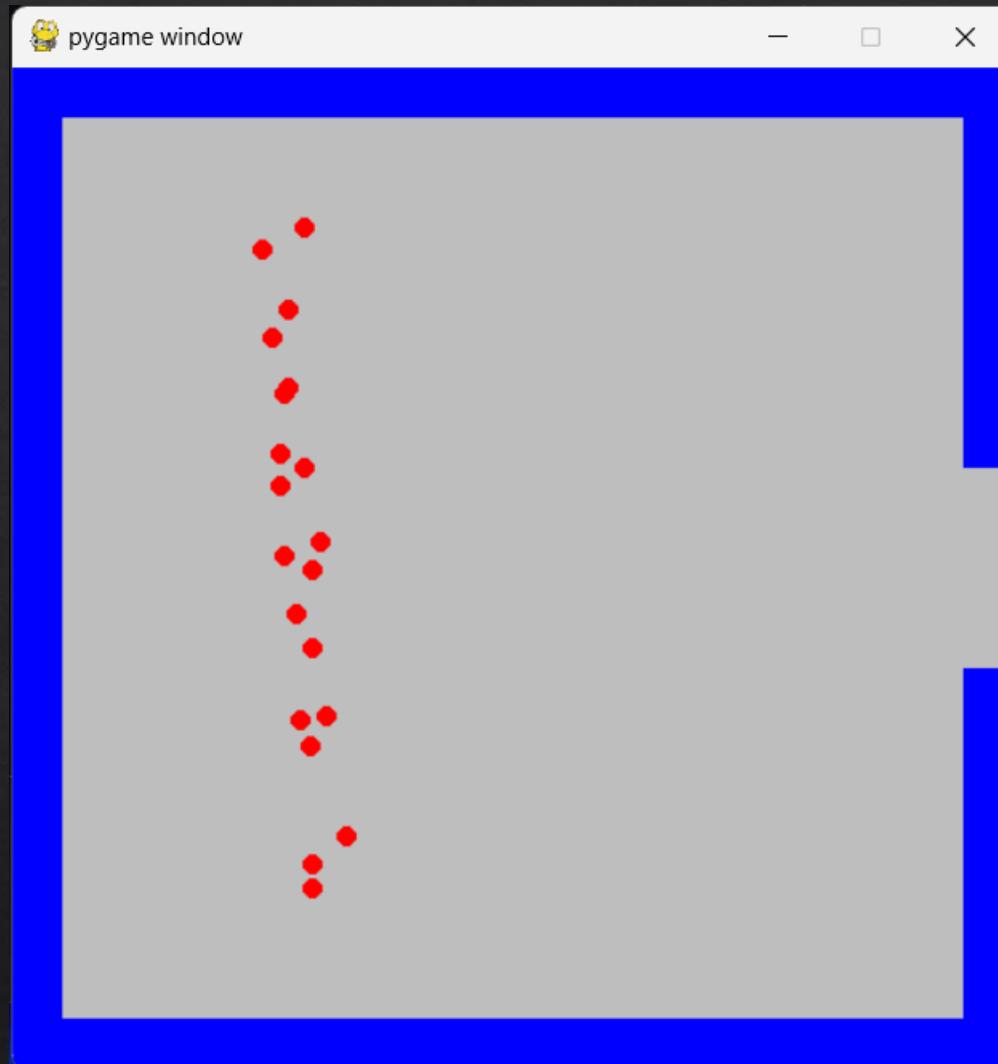
Ln: 31 Col: 0



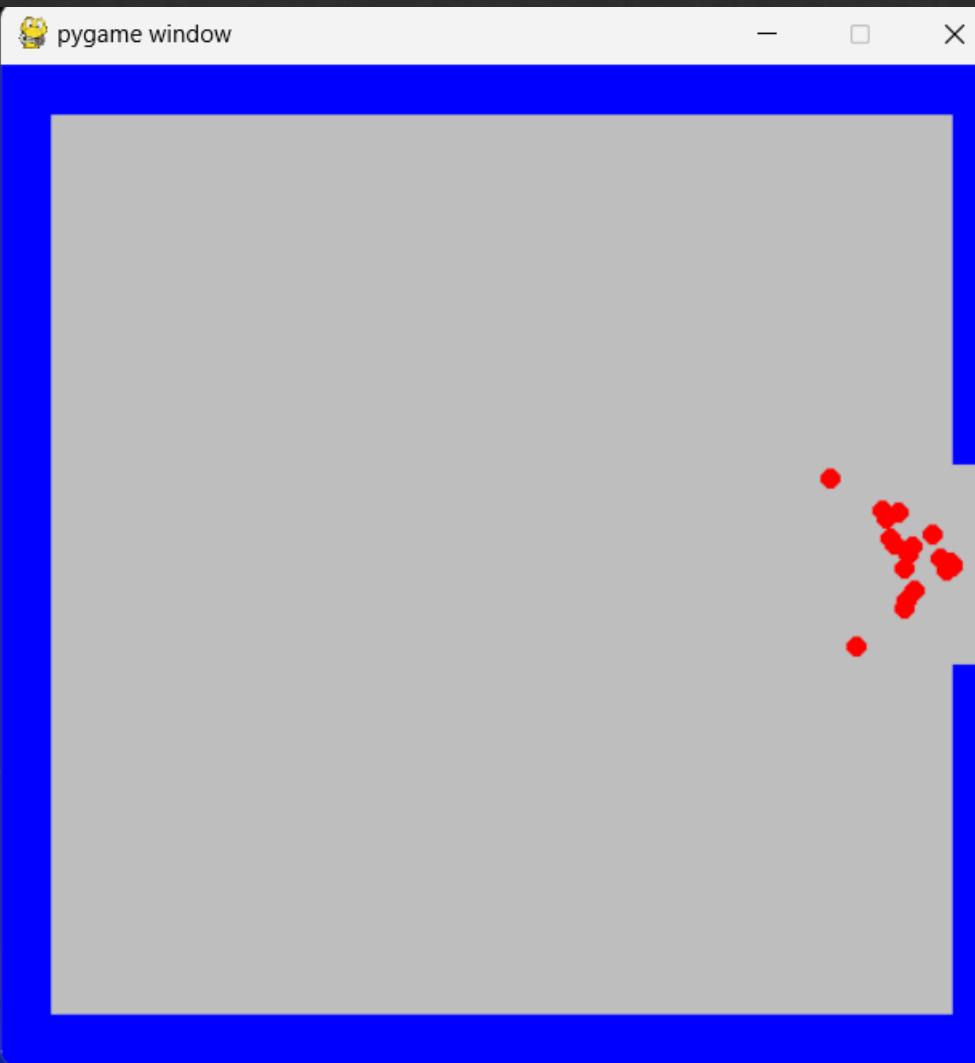
21:55  
FRA  
28/05/2023

```
import pygame ←  
pygame.init() ←  
win = pygame.display.set_mode((500, 500)) ←  
pygame.display.set_caption("Premier Jeu")  
x,y=50,50  
run = True ←  
# Boucle principale  
while run:  
    # "Horloge" pour éviter que les choses ne se déroulent trop rapidement  
    pygame.time.delay(50)  
    # Événements (tout ce qui se passe de la part de l'utilisateur)  
    for event in pygame.event.get():  
        if event.type == pygame.QUIT:  
            run = False  
    # Mouvement  
    keys = pygame.key.get_pressed()  
    # Déplacement horizontal  
    if 0 <= x <= 460:  
        if keys[pygame.K_LEFT]:  
            x -= 10  
            if keys[pygame.K_x]:  
                x -= 20  
        if keys[pygame.K_RIGHT]:  
            x += 10  
            if keys[pygame.K_x]:  
                x += 20  
    elif x > 460:  
        x = 460  
    else:  
        x = 0  
  
    win.fill((0, 0, 0)) # Effacer la fenêtre ←  
    pygame.draw.circle(win, (255, 0, 0), (x, y), 10) # Dessiner le cercle ←  
    pygame.display.update() # Rafraîchir la fenêtre ←  
pygame.quit() ←
```









File Edit Format Run Options Window Help

```
import pygame , random , time
from Boids import *
from Bords import *

screen = pygame.display.set_mode((500,500))

l=[ i for i in range(50,451,20)]
p=[ i  for i in range(50,451,20)]

##m=np.zeros((450,450))
##for i in range(450):
##    for j in range(450):
##        m[i,j]=

Birds=[]
for i in range(20):
    Birds.append(Boid(i+40,p[i],1,1,255,0,0))
    #Birds.append(Boid(250-i*10.5,250,1,1,255,0,0))
##a=Boid(200,200,1,1,1,255,1)
##b=Boid(300,300,5,5,255,0,0)

bord1=Bord(25)

run=True

while run:
    pygame.time.delay(50)
    #win.fill((0,0,0))
    screen.fill("Grey")

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False
    bord1.dessiner_bords()
    for i in range(20):
        Birds[i].move()

    ##    Birds[0].move()
    ##    Birds[10].move()
    #print('Birds[{}](x),Birds[{}](y)'.format(0,0),Birds[0].x,Birds[0].y)
    pygame.display.update()
    #time.sleep(0.01)

pygame.quit()
```

bord1.dessiner\_bords()  
for i in range(20):  
 Birds[i].move()

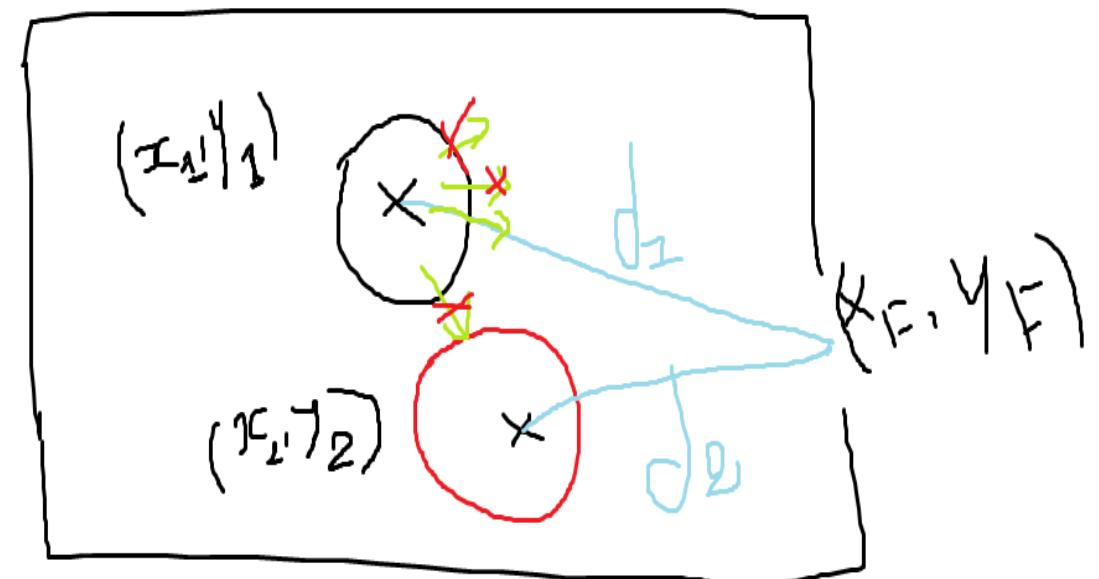


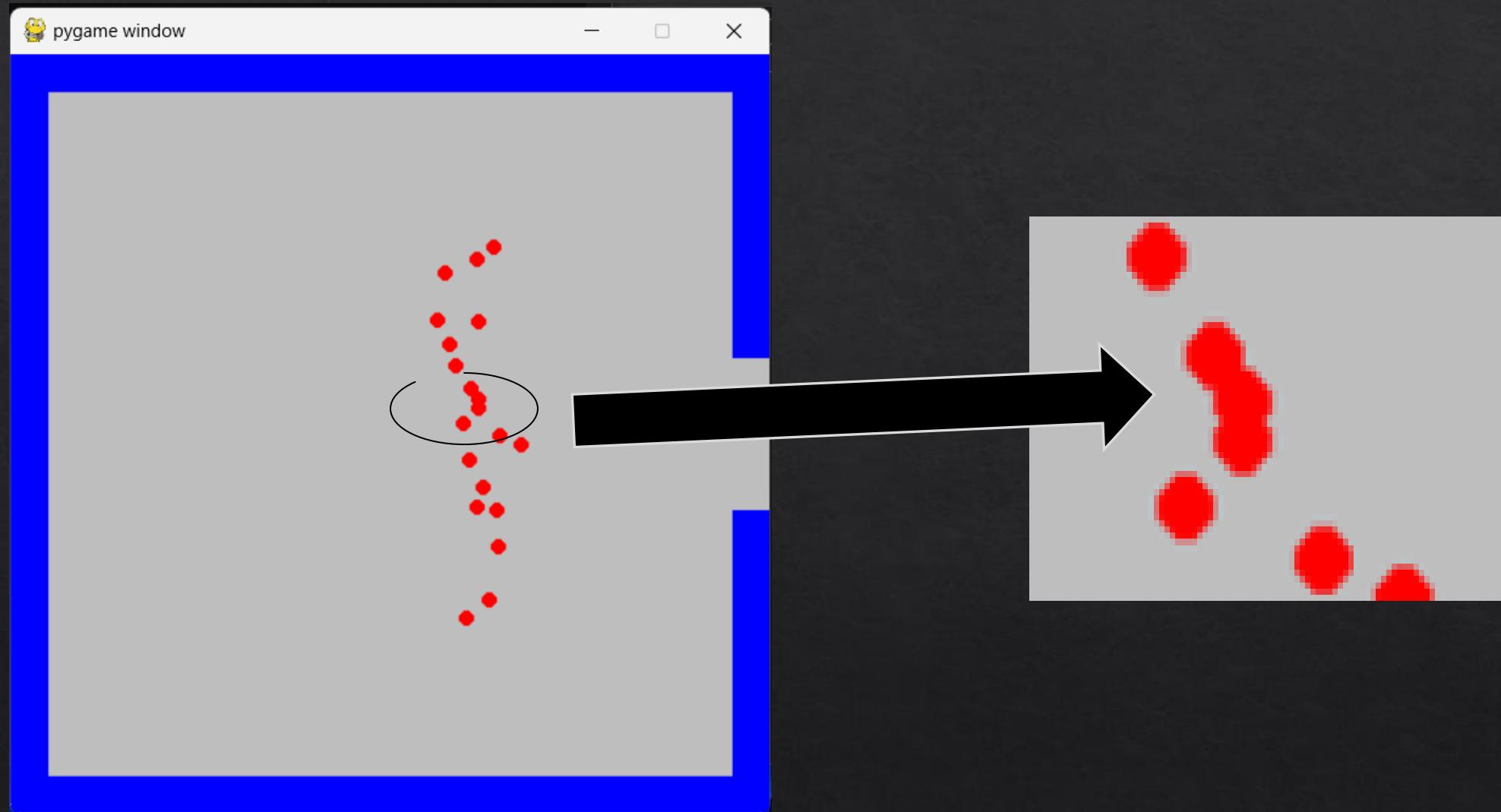
```
#class Bord:
#    def __init__(self, v):
#        self.v=v
#    def dessiner_bords(self):
#        l=[ i for i in range(50,451,20)]
#        p=[ i  for i in range(50,451,20)]
```

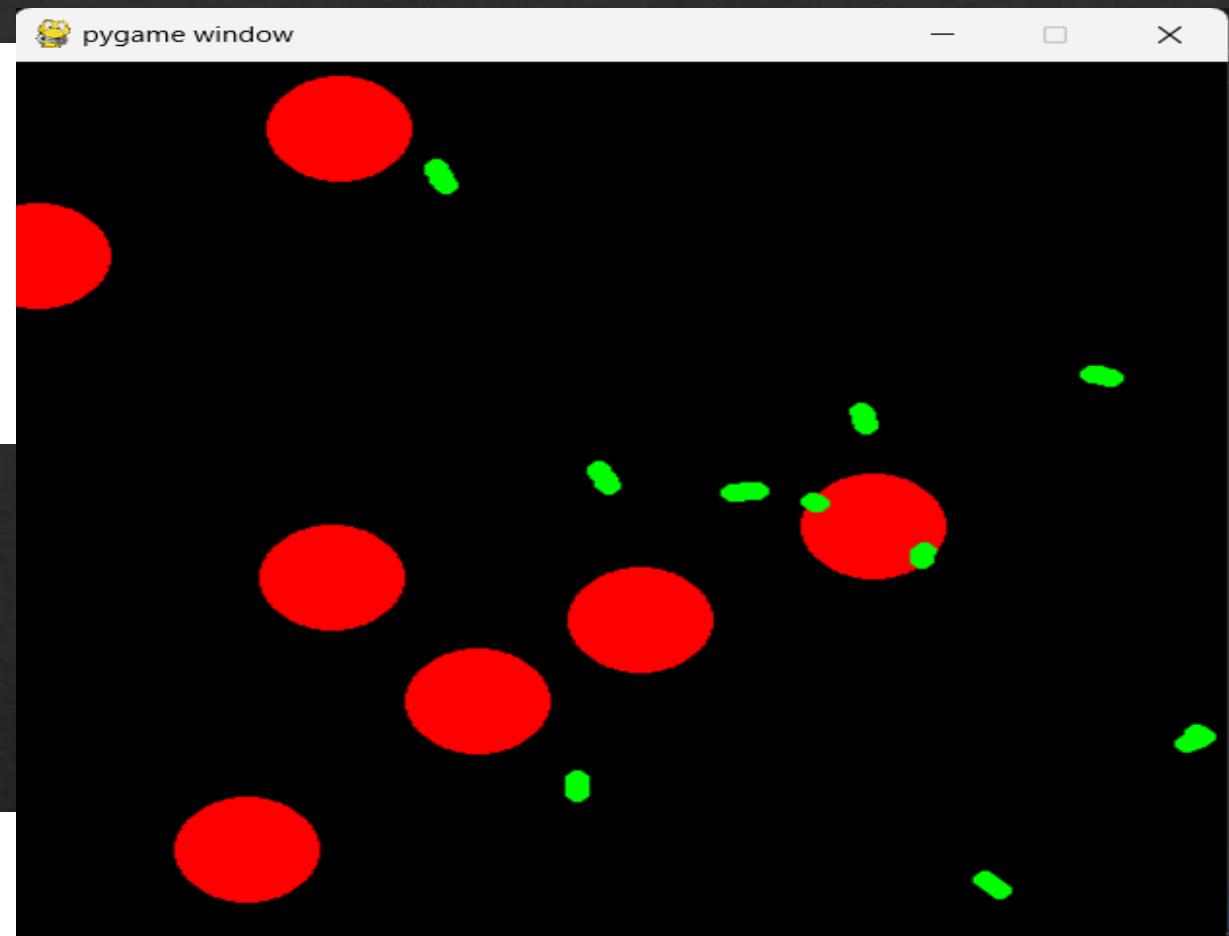
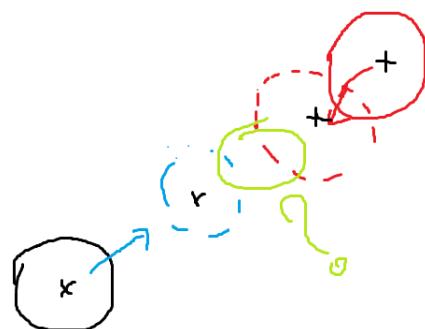
```

def Distance_Existnce(self,c,d):
    a=(Exist_x-c)**2
    b=(Exist_y-d)**2
    return math.sqrt(a+b)
def update_position(self):
    self.dx=random.uniform(-3.5,3.5)
    self.dy=random.uniform(-3.5,3.5)
    return (self.dx, self.dy)
def move(self): x
    d1=self.Distance_Existnce(self.x, self.y)
    Run=(self.x, self.y)==(Exist_x, Exist_y)
    while not(Run):
        dx,dy=self.update_position()
        c=self.x+dx
        d=self.y+dy
        d2=self.Distance_Existnce(c, d)
        if d2<d1:
            #print("d1={}, d2={}".format(d1, d2))
            self.x+=dx
            self.y+=dy
            break

```







```
def DetectColli(self):  
    if distance(self.persx, self.persy, self.obsx, self.obsy) < (self.persradius + self.obsradius) - 2:  
        return (True)
```

B

- Tentative 2

B

**B-1**

## Recherche de modèle

# Des Modèles qui existaient Déjà

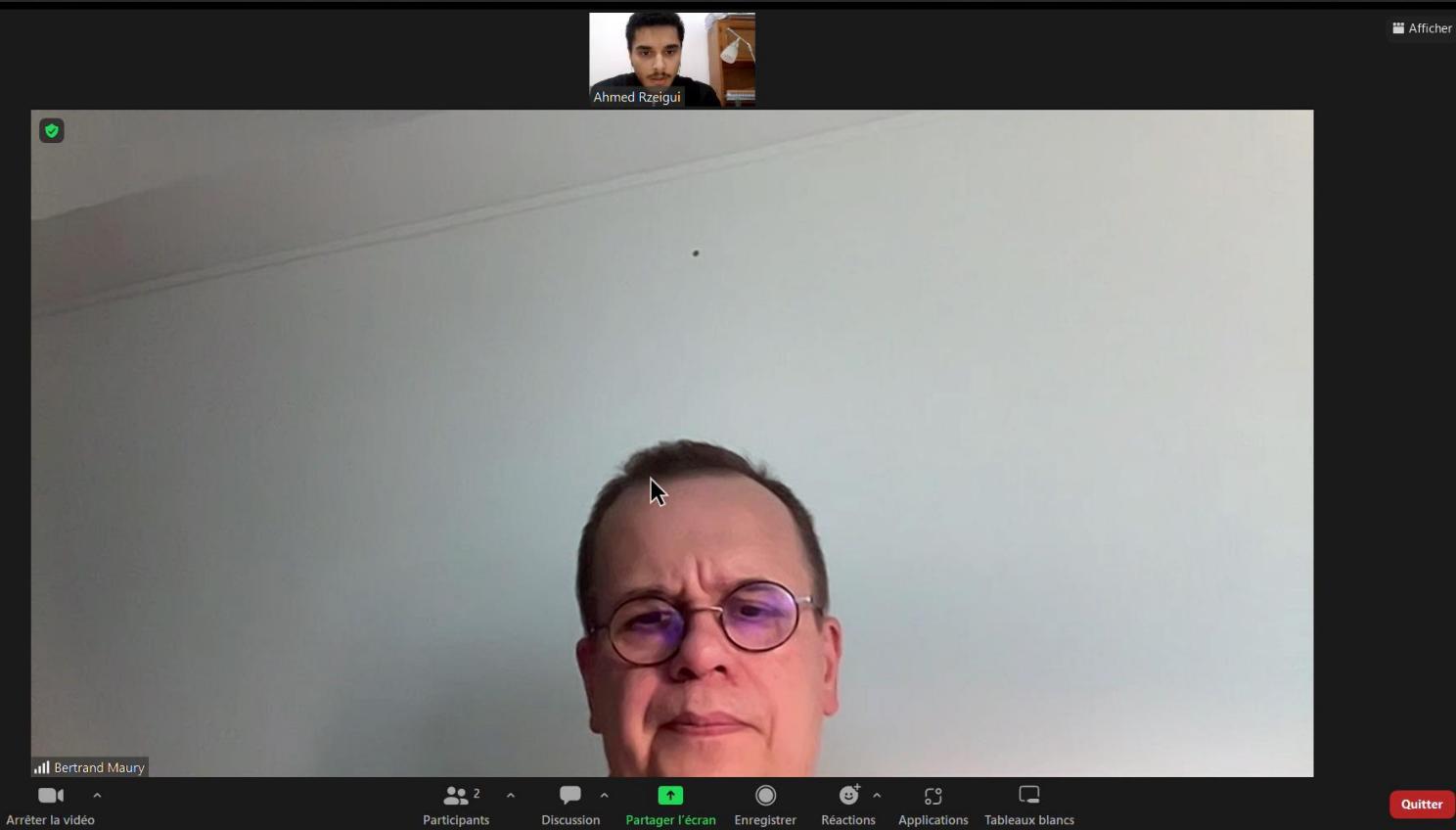
- ❖ Des Modèles Macroscopiques:

1. modèle de régression
2. modèle dynamique de fluide ou gaz

- ❖ Des Modèles Microscopiques :

1. Modèle à base de règles (Boids ,Introduits par Reynolds)
2. Modèle de forces sociales
3. modèle d'automates cellulaires
4. Modèle mathématique de Venel

# UN conatact Avec M.Bertrand Maury

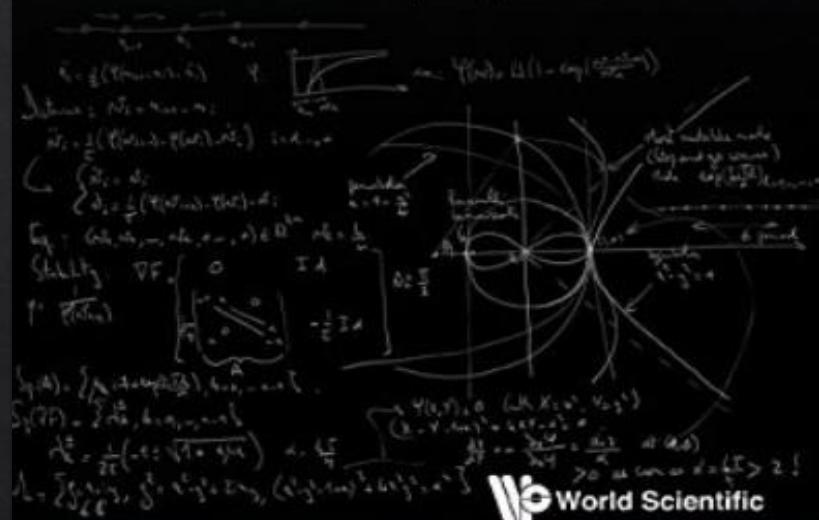


Advanced Textbooks in Mathematics

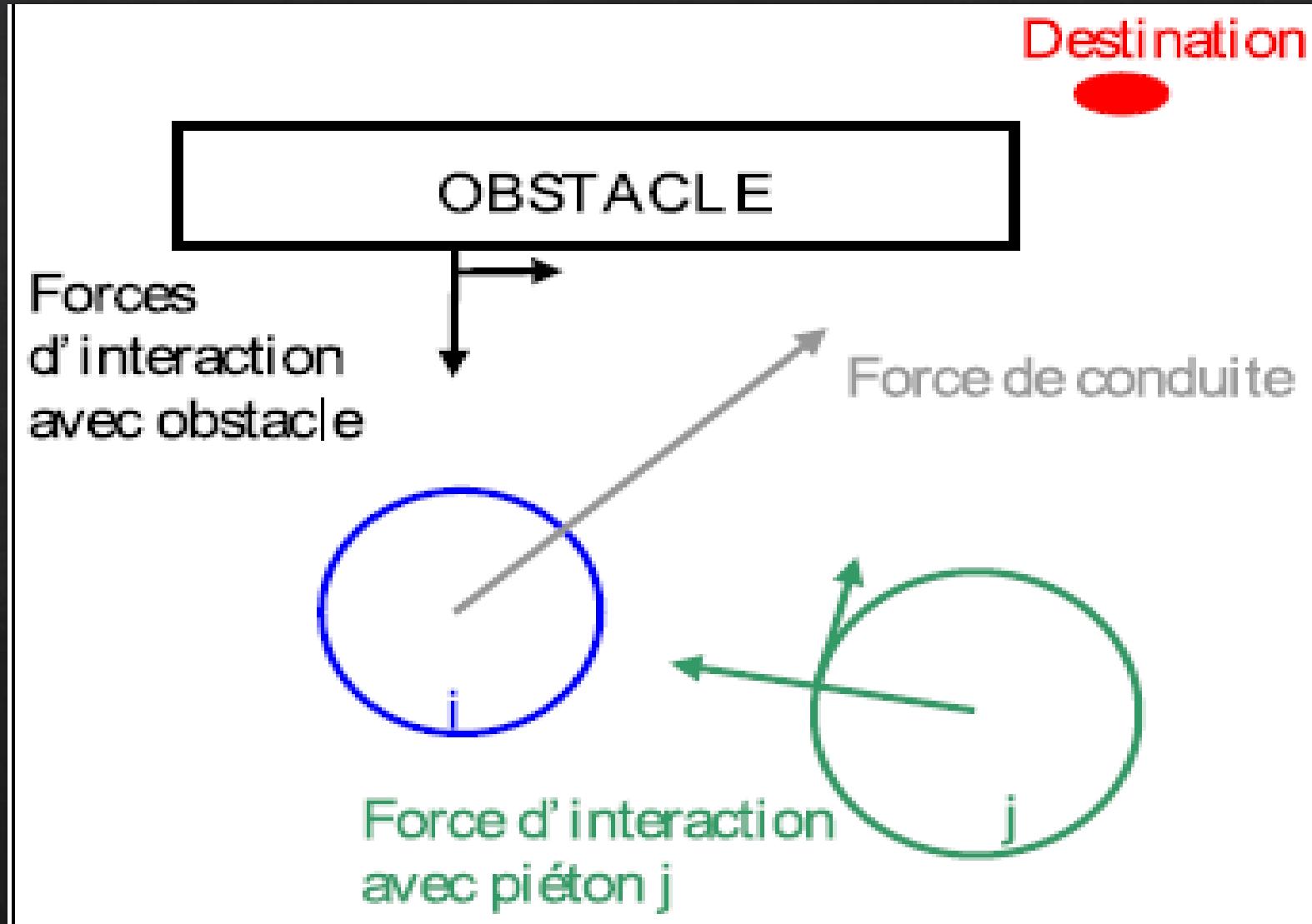
# Crowds in Equations

An Introduction to the Microscopic Modeling of Crowds

Bertrand Maury • Sylvain Faure



# Modèle de Force Sociale



# Equation du Mouvement

$(x_i \in \mathbb{R}^2)$

$$\frac{dx_i}{dt} = U_i + \sum_{i \neq j} w_{ij}$$

The diagram illustrates the components of the movement equation. Three yellow rectangular boxes, each containing a question mark, are positioned below the equation. Arrows from each box point upwards towards the corresponding terms: the first arrow points to  $\frac{dx_i}{dt}$ , the second to  $U_i$ , and the third to  $w_{ij}$ .

$$\frac{dx}{dt} = -\nabla \psi(x) \longrightarrow U_i = -\nabla \left( \sum_i v_i(x_i) \right)$$



$$v_i(x_i) = UD(x_i)$$

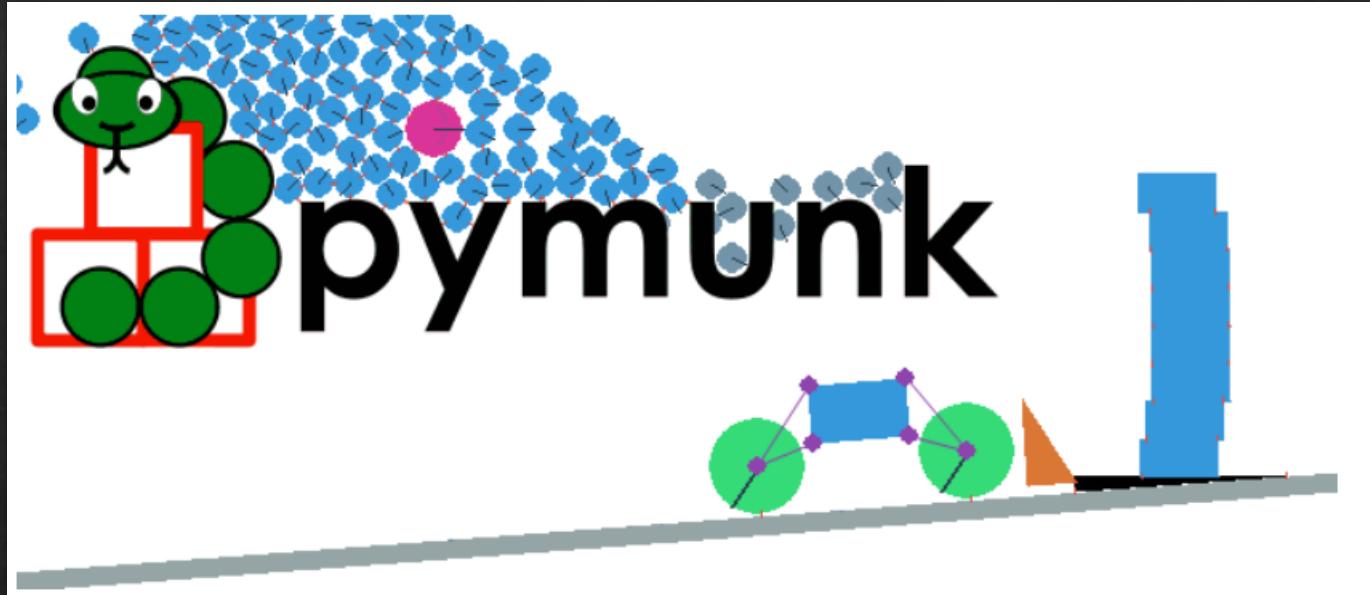
$$W_{ij} = -U \exp\left(-\frac{D_{ij}}{\delta}\right) e_{ij}$$

B

B-2

## Implémentation du modèle

# Comment améliorer le Code ?



Son Avantage

Manipuler les Forces , Les Vitesses ,  
les Collisions PLUS FACILEMENT

A thumbnail for a YouTube video titled "Simulating physics in Python". The thumbnail has a red background with a white Apple logo on the left. The title "Simulating physics in Python" is at the top, followed by "104 k vues • il y a 2 ans". Below that is a "Clear Code" badge with a checkmark. The video description at the bottom reads: "This tutorial is about simulating physics in python with the pymunk module; and to visualise the results pygame will be used." A progress bar at the bottom indicates the video is 23:21 long.

Simulating physics in Python  
104 k vues • il y a 2 ans  
Clear Code ✓  
This tutorial is about simulating physics in python with the pymunk module; and to visualise the results pygame will be used.

# En effet :

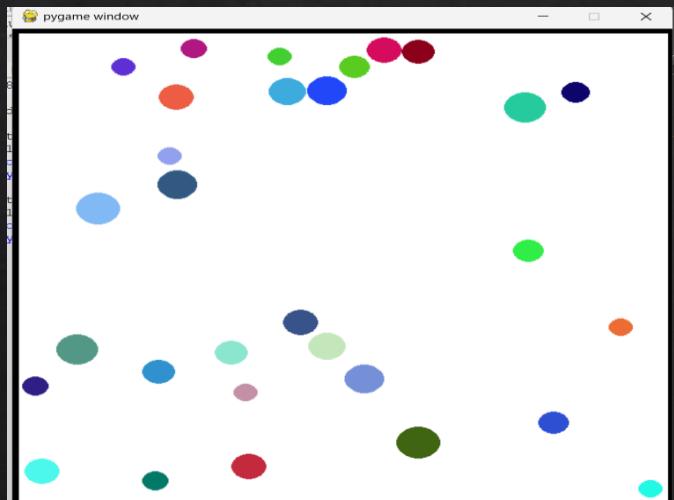
```
# Classe Personne
class Personne:
    def __init__(self, position, radius):
        self.radius = radius
        self.position = position
        self.velocity = (0, 0)

    # Création du corps physique
    mass = 1.0
    inertia = pymunk.moment_for_circle(mass, 0, self.radius)
    self.body = pymunk.Body(mass, inertia)
    self.body.position = self.position
    self.shape = pymunk.Circle(self.body, self.radius*1.43)
    space.add(self.body, self.shape)
```

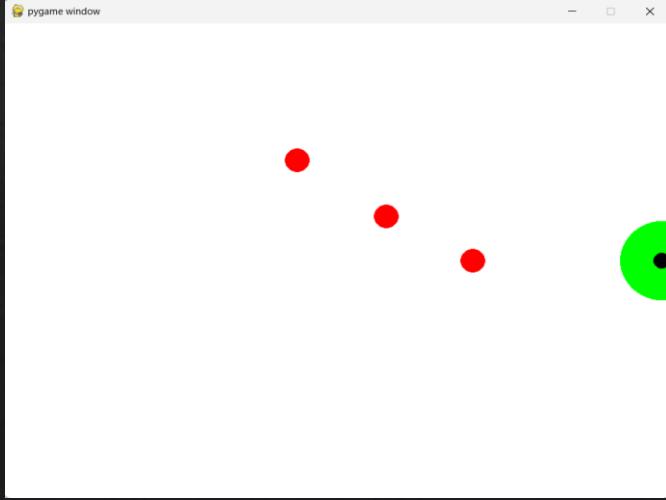
# Simplification des Taches



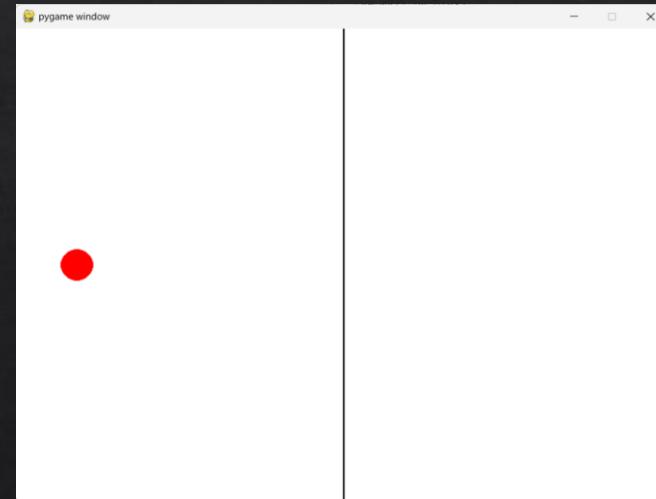
0.Boules en mouvement aléatoires sans collision



1.Boules en mouvement vers un Target Point



2.Comment\_Calculer\_Temps



$$\text{Retour sur L'équation : } \frac{dx_i}{dt} = U_i + \sum_{i \neq j} w_{ij}$$

```
def calculate_desired_velocity(self, target_point):
    # Calculer le gradient de la fonction de potentiel
    diff_x = target_point[0] - self.position[0]
    diff_y = target_point[1] - self.position[1]
    distance = math.sqrt(diff_x**2 + diff_y**2)
    gradient_x = diff_x / distance
    gradient_y = diff_y / distance

    # Calculer la vitesse désirée
    desired_velocity_x = U * gradient_x
    desired_velocity_y = U * gradient_y

    return (desired_velocity_x, desired_velocity_y)
```

```
def calculate_correction_velocity(self, balls):
    correction_velocity = (0, 0)

    for ball in balls:
        if ball != self:
            diff_x = ball.position[0] - self.position[0]
            diff_y = ball.position[1] - self.position[1]
            distance = math.sqrt(diff_x**2 + diff_y**2)
            direction_x = diff_x / distance
            direction_y = diff_y / distance
            direction = [diff_x, diff_y]

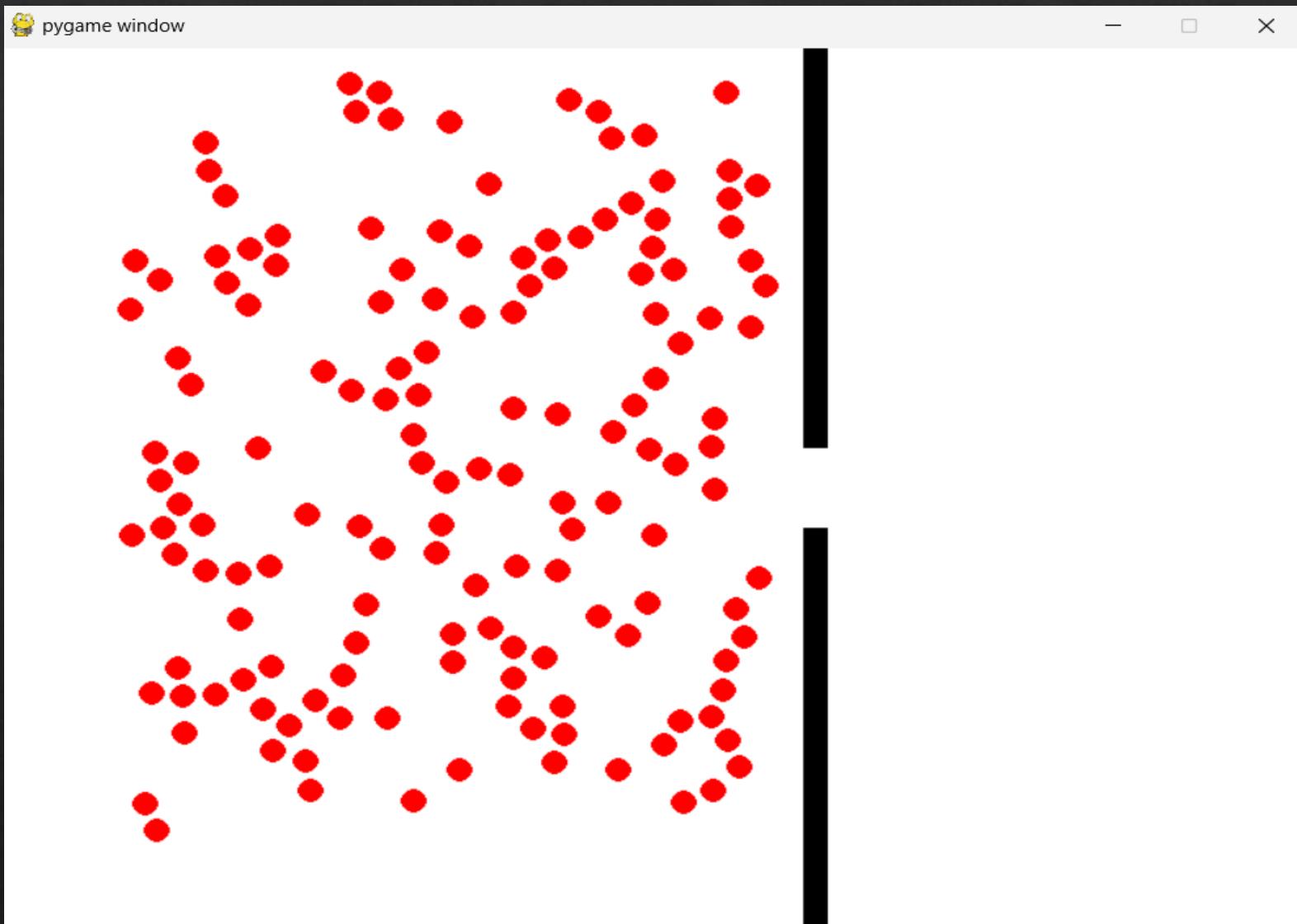
            # Calculer la correction de vitesse
            correction = [-U * math.exp(-distance/delta) * d for d in direction]
            correction_velocity = (correction_velocity[0] + correction[0], correction_velocity[1] + correction[1])

    return correction_velocity
```

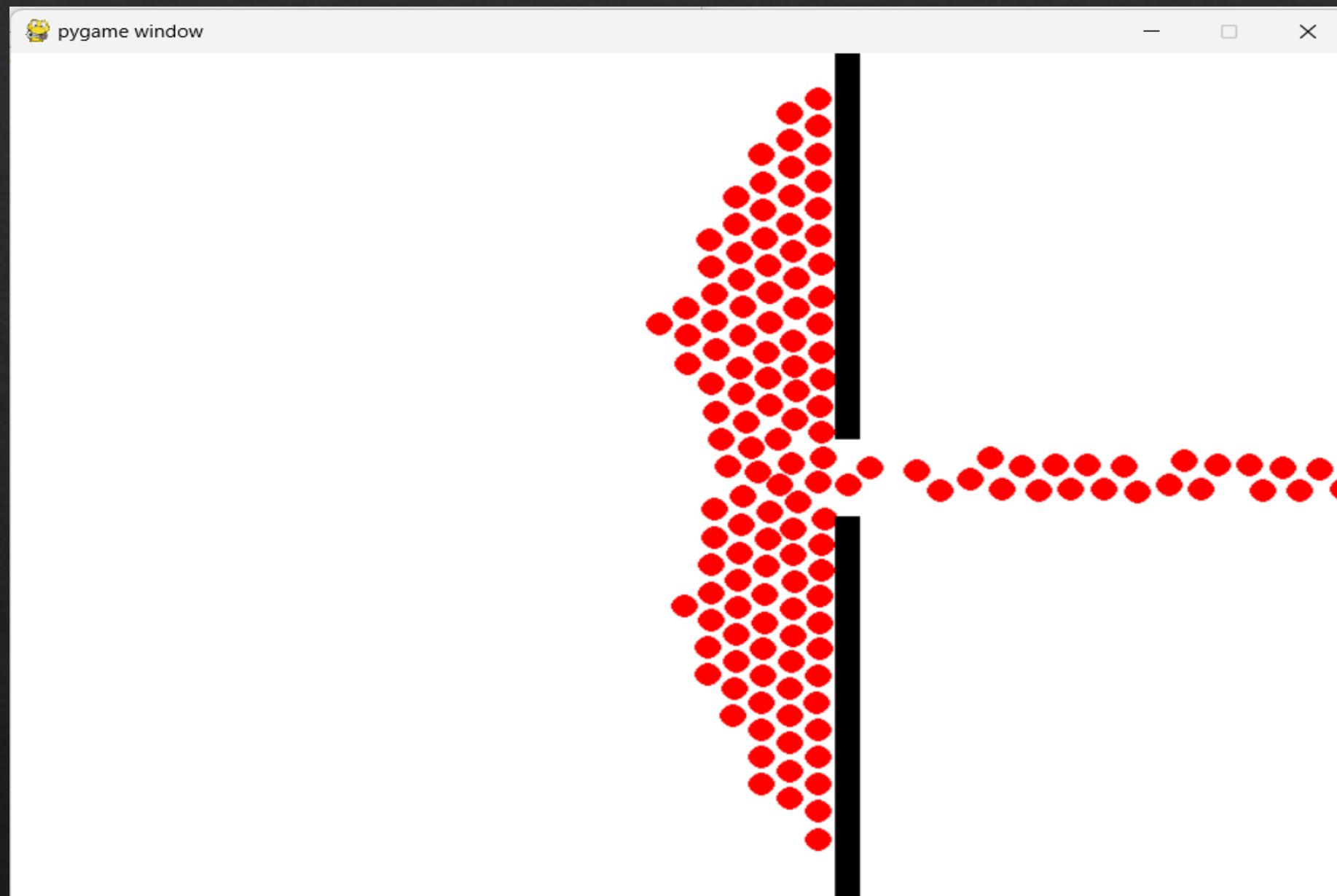
Résultat Obtenu :

$U=50.0$  ,  $\delta=0.08$ , Nombre de personne = 150

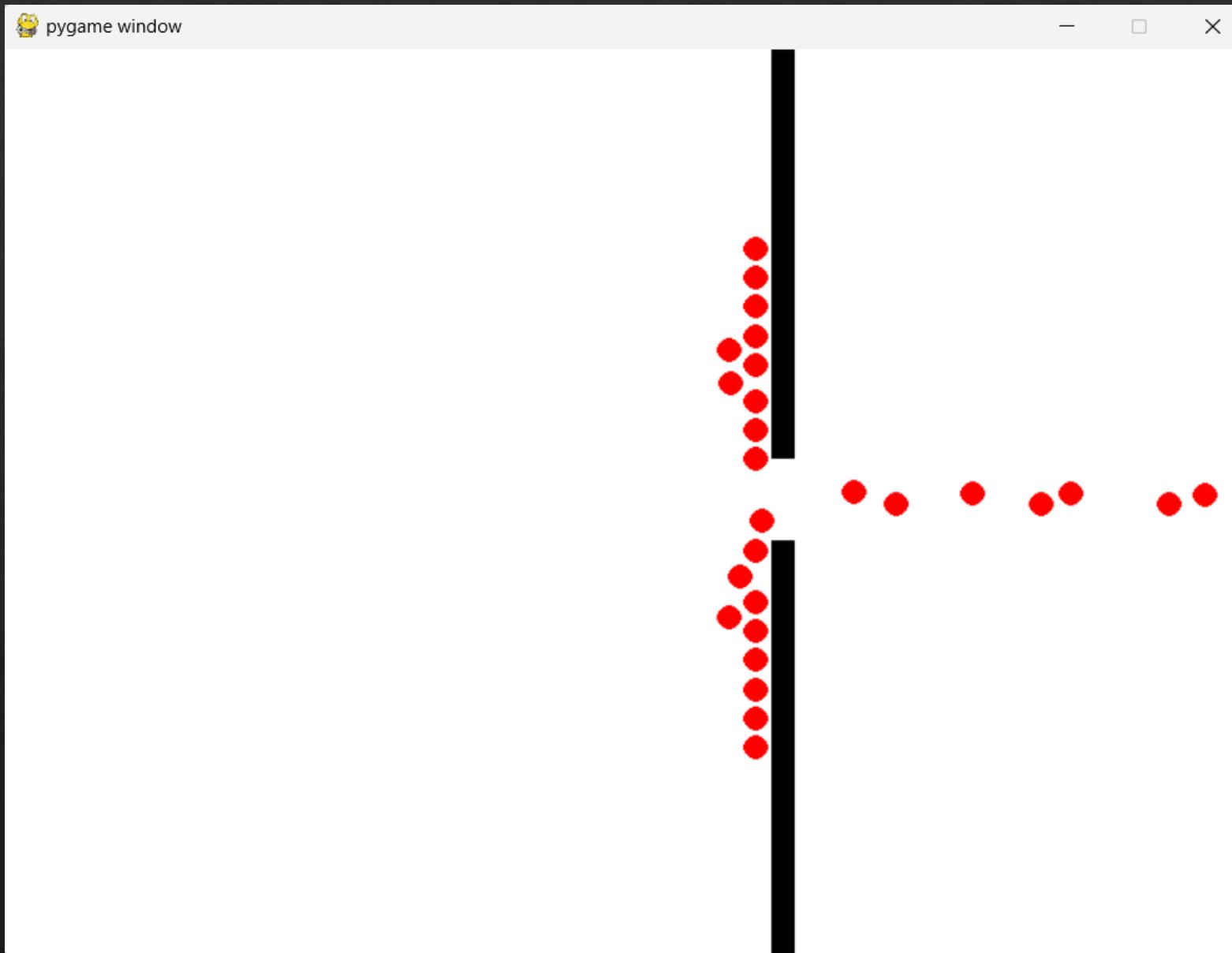
$T = 0,05 \text{ s}$



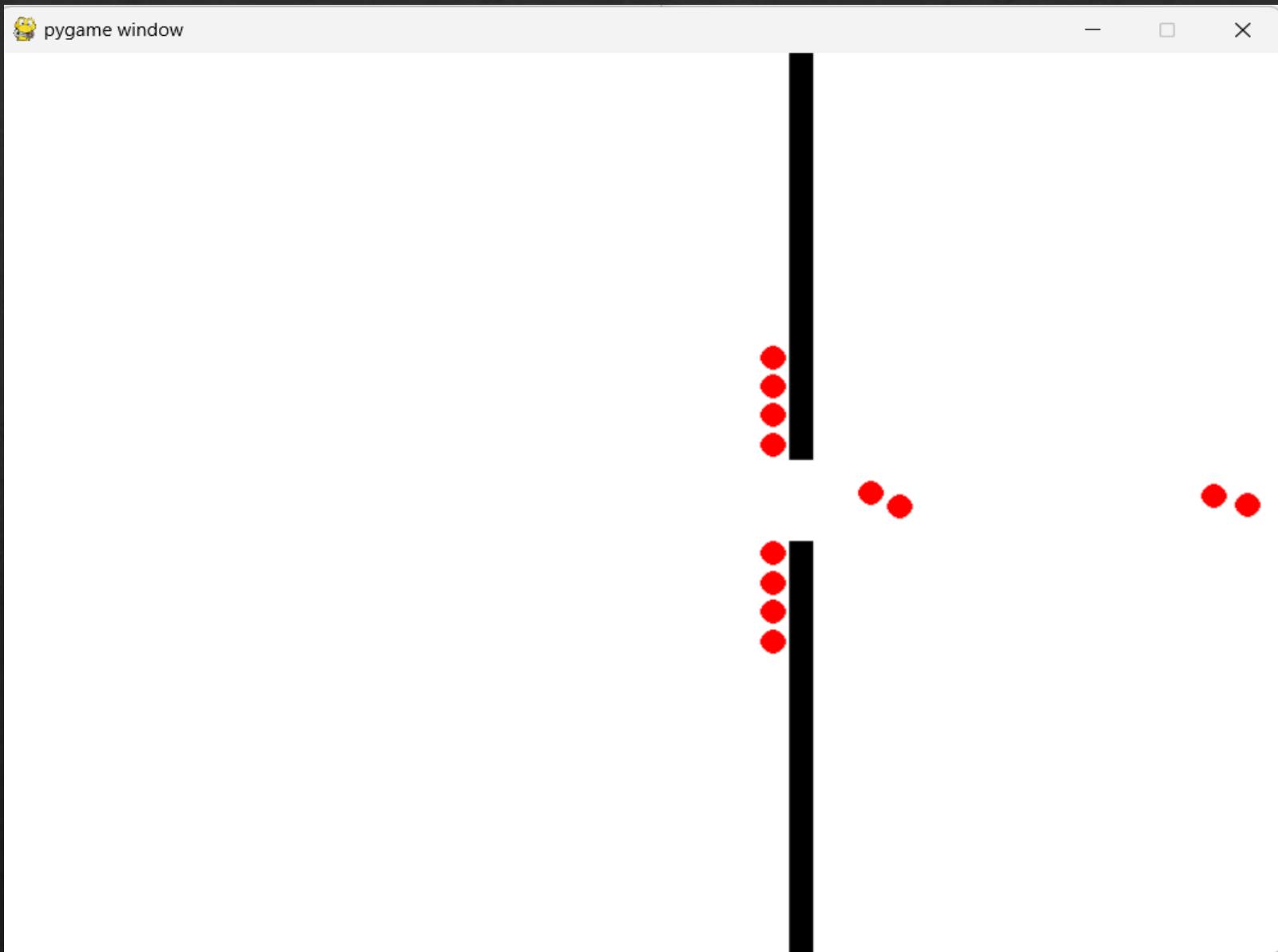
$T = 30 \text{ s}$



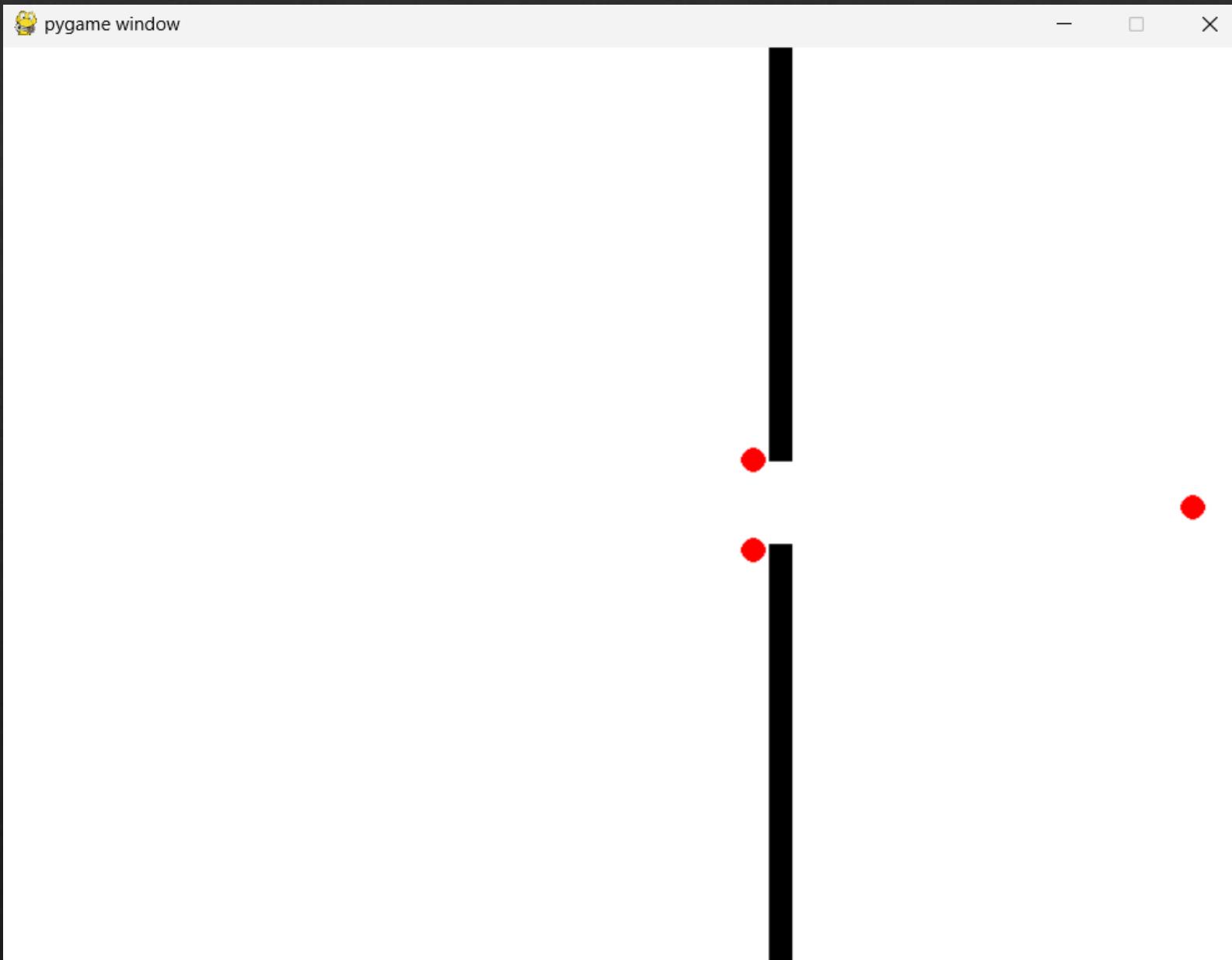
$T = 2 \text{ min}$



$T = 2 \text{ min } 41 \text{ s}$

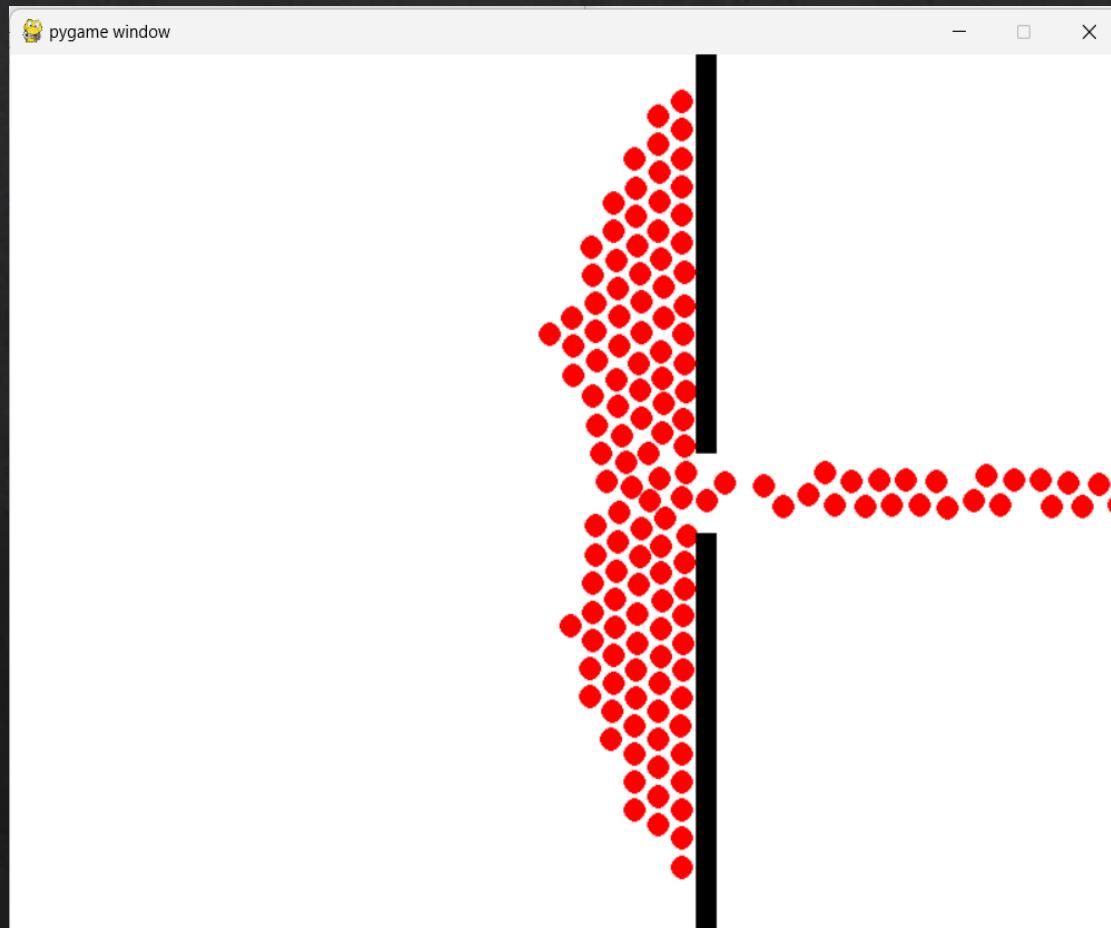


$T = 4 \text{ min}$



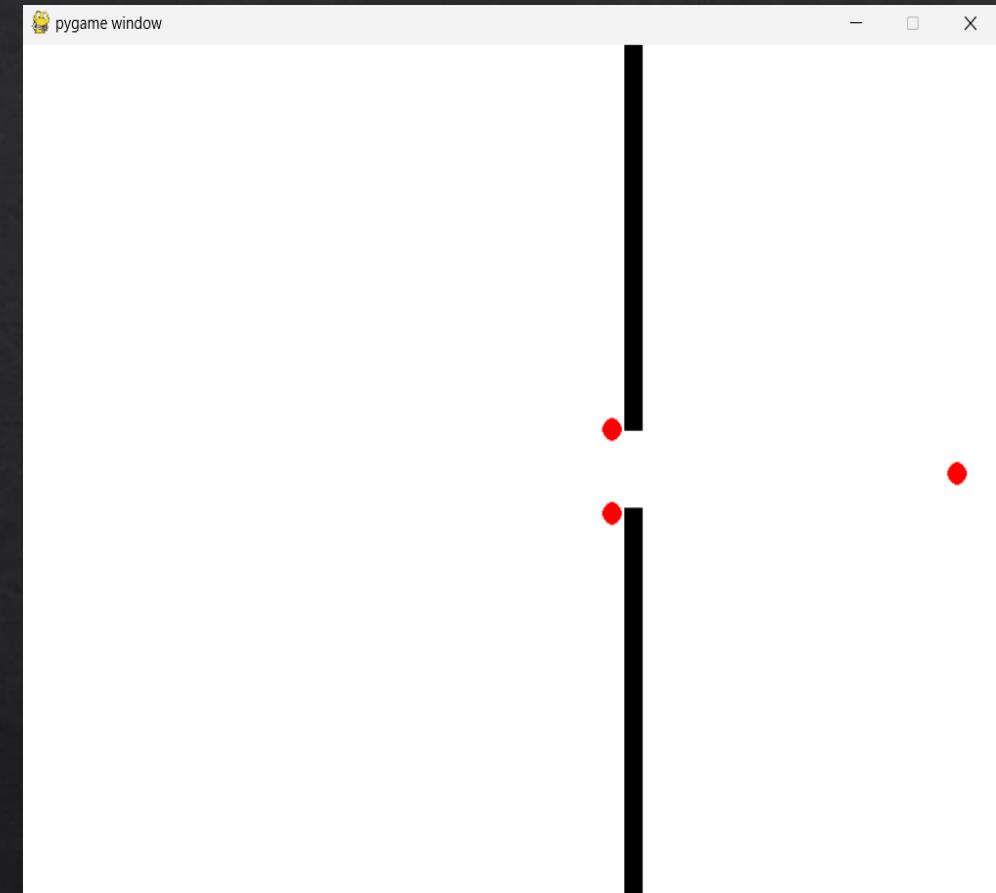
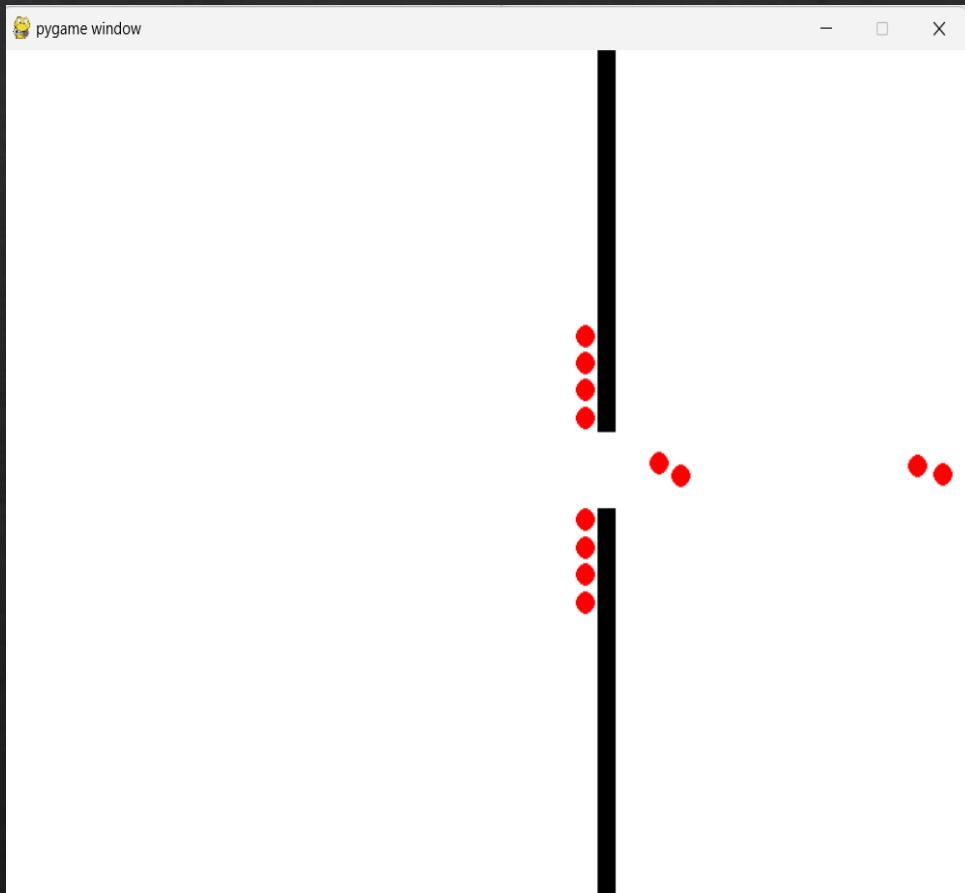
## Vérification de Phénomènes d'auto-organisation observés :

- l'embouteillage
- le blocage et le phénomène d'arche

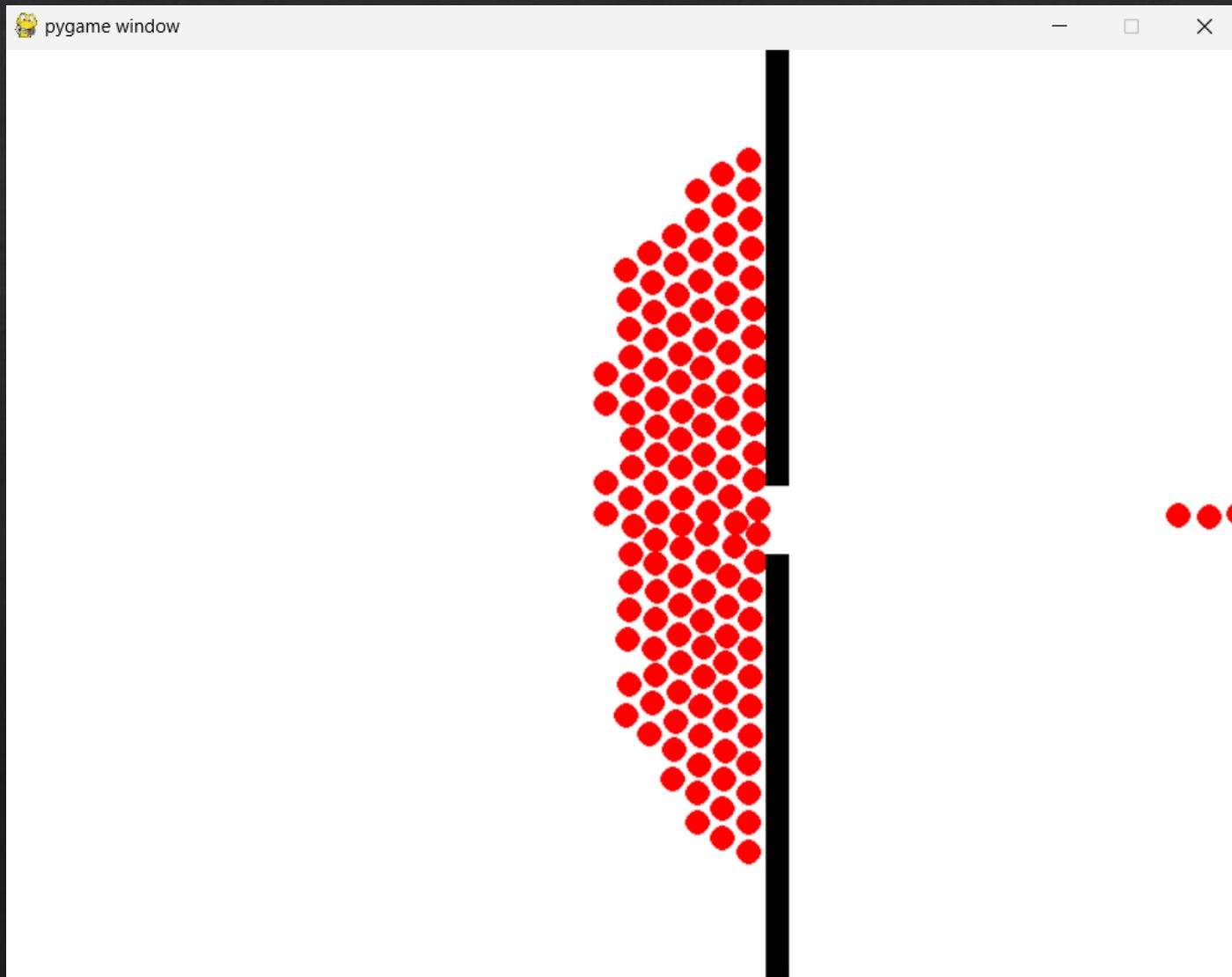


# Limites de la simulation :

- ❖ La vitesse diminue considérablement pour les dernières boules en effet :



- ❖ La largeur du Porte (Ici Fente ) peut Bloquer Entièrement le mouvement même si elle supérieur au diamètre d'une boule en effet :



C

- Comparaison

# Simulation d'une évacuation d'une Salle PROPOS2 PAR

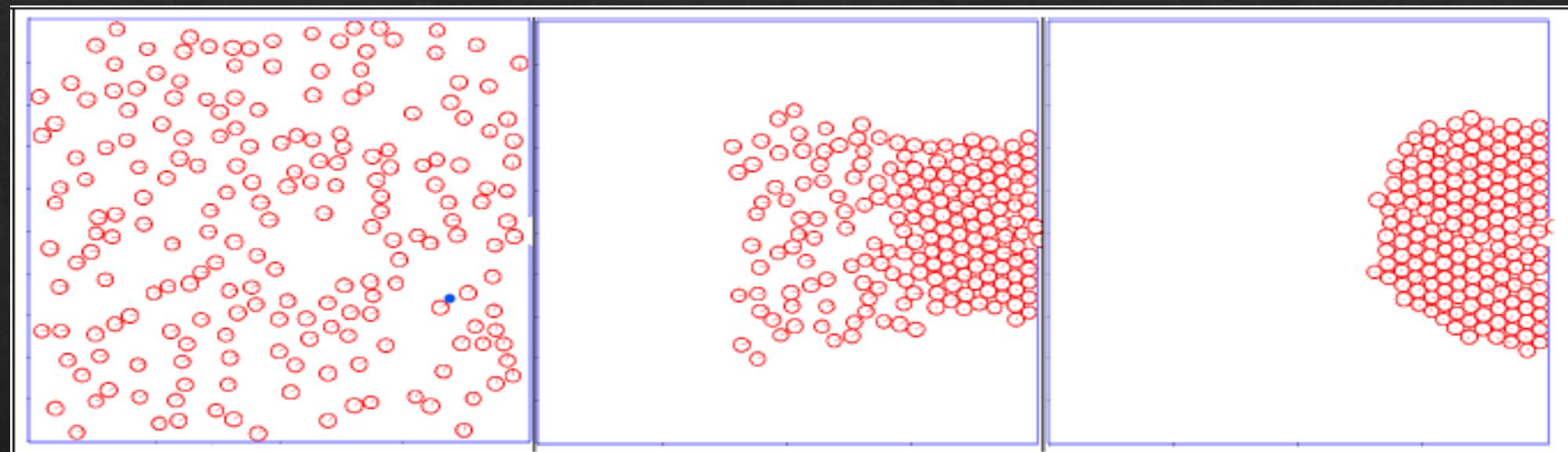


FIGURE 6.3 – Simulation d'une évacuation de salle - Formation d'arche - a.  $t = 0 \text{ s}$ , 200 piétons veulent évacuer; b.  $t = 6 \text{ s}$ , 187 piétons restent à évacuer; c.  $t = 12 \text{ s}$ , 166 piétons restent à évacuer.

En utilisant la méthode NSM 2 il a pu obtenir :

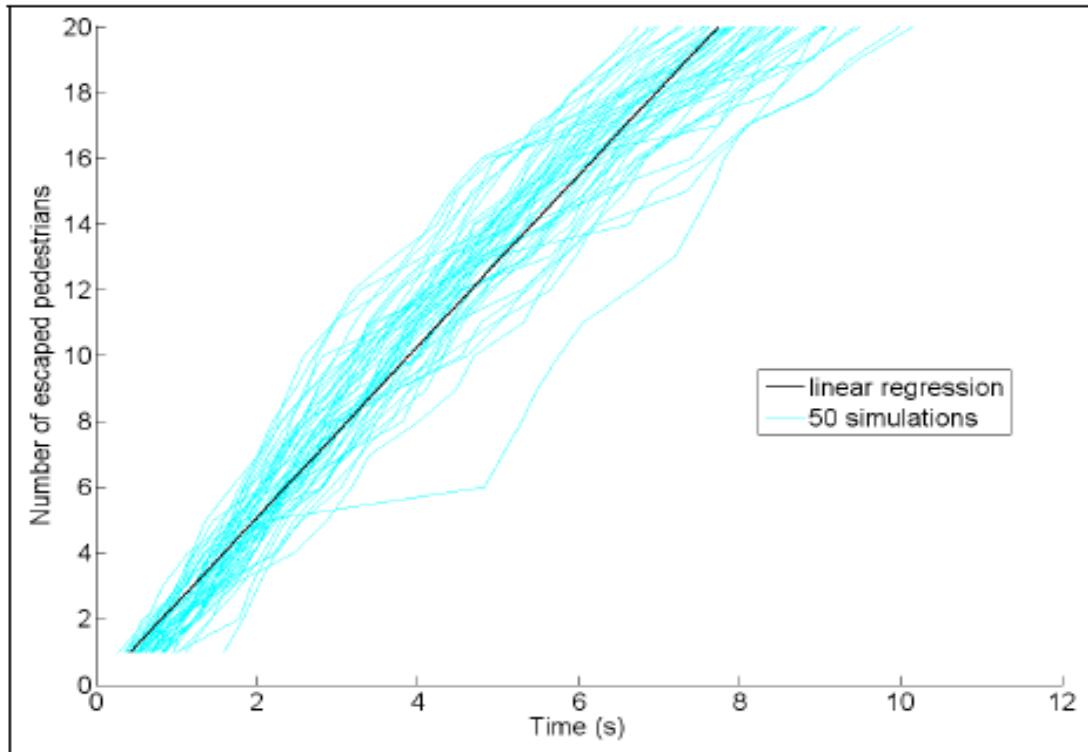


FIGURE 6.4 – Evacuation de salle - Courbes d'évacuation pour NSM2, avec  $h = 10^{-2} \text{ s}$ . Les courbes de couleur cyan résultent des 50 simulations. La régression linéaire des 50 simulations (droite noire) permet d'obtenir le débit moyen de piétons traversant la porte.

# Implémentation de la Régression Linéaire

```
def tracage(nombre_ball, nombre_simulation):
    import numpy as np
    import matplotlib.pyplot as plt
    from sklearn.linear_model import LinearRegression

    Personne_evacuee = np.arange(1, nombre_ball+1)
    Temps_Sortie_Personne = []

    # Effectuer les 10 simulations
    for i in range(nombre_simulation):
        temps_simulation = simulate(nombre_ball) # Remplacez cette ligne par votre fonction simulate
        Temps_Sortie_Personne.append(temps_simulation)

    # Convertir les listes en tableaux numpy
    Personne_evacuee = np.array(Personne_evacuee)
    Temps_Sortie_Personne = np.array(Temps_Sortie_Personne)

    # Tracer les courbes
    plt.figure()
    for i in range(nombre_simulation):
        plt.plot(Temps_Sortie_Personne[i], Personne_evacuee, color='green')

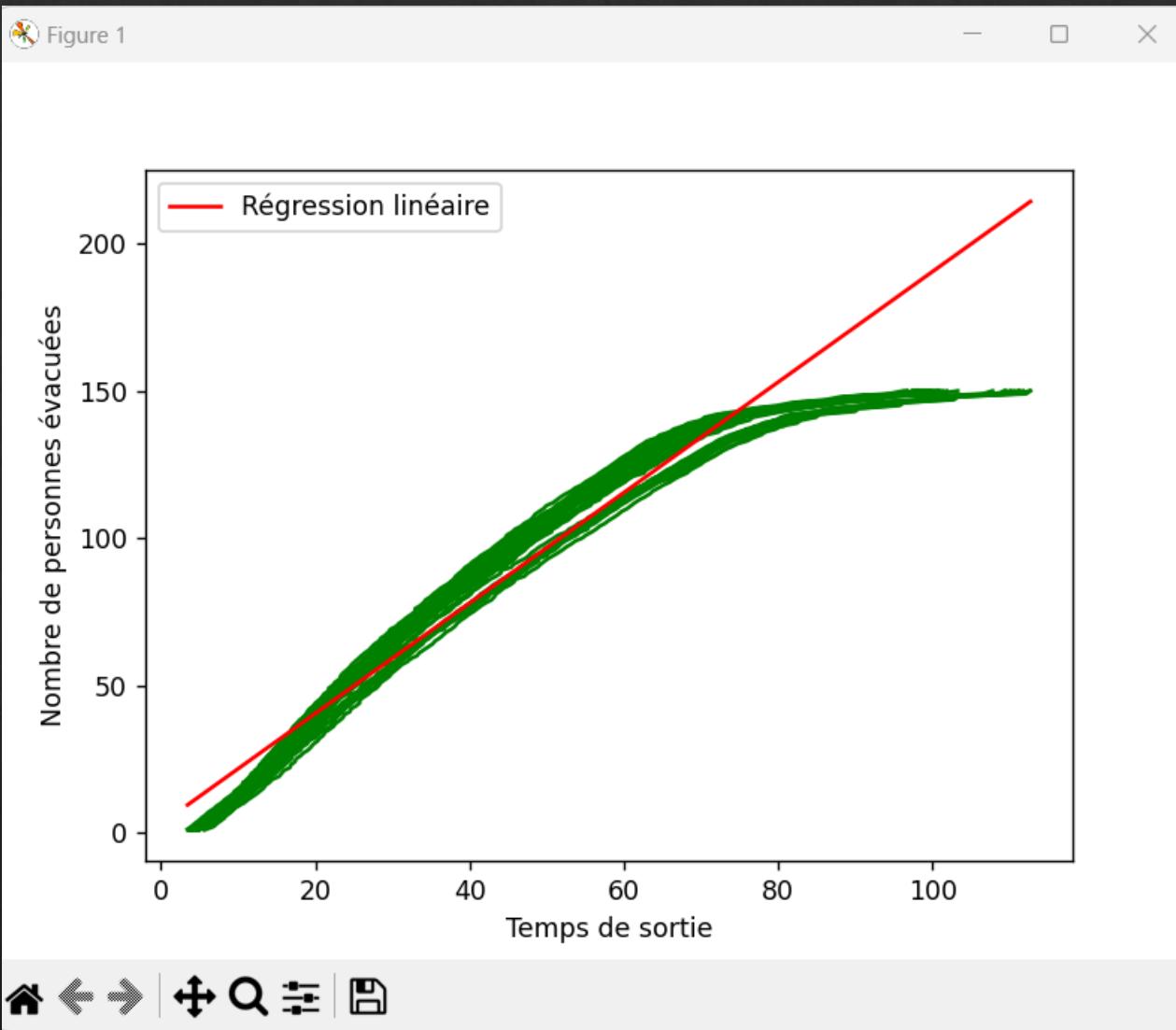
    # Effectuer la régression linéaire
    reg = LinearRegression()
    reg.fit(Temps_Sortie_Personne.mean(axis=0).reshape(-1, 1), Personne_evacuee.reshape(-1, 1))

    # Obtenir les prédictions de la régression linéaire
    x_pred = np.linspace(Temps_Sortie_Personne.min(), Temps_Sortie_Personne.max(), 100)
    y_pred = reg.predict(x_pred.reshape(-1, 1))

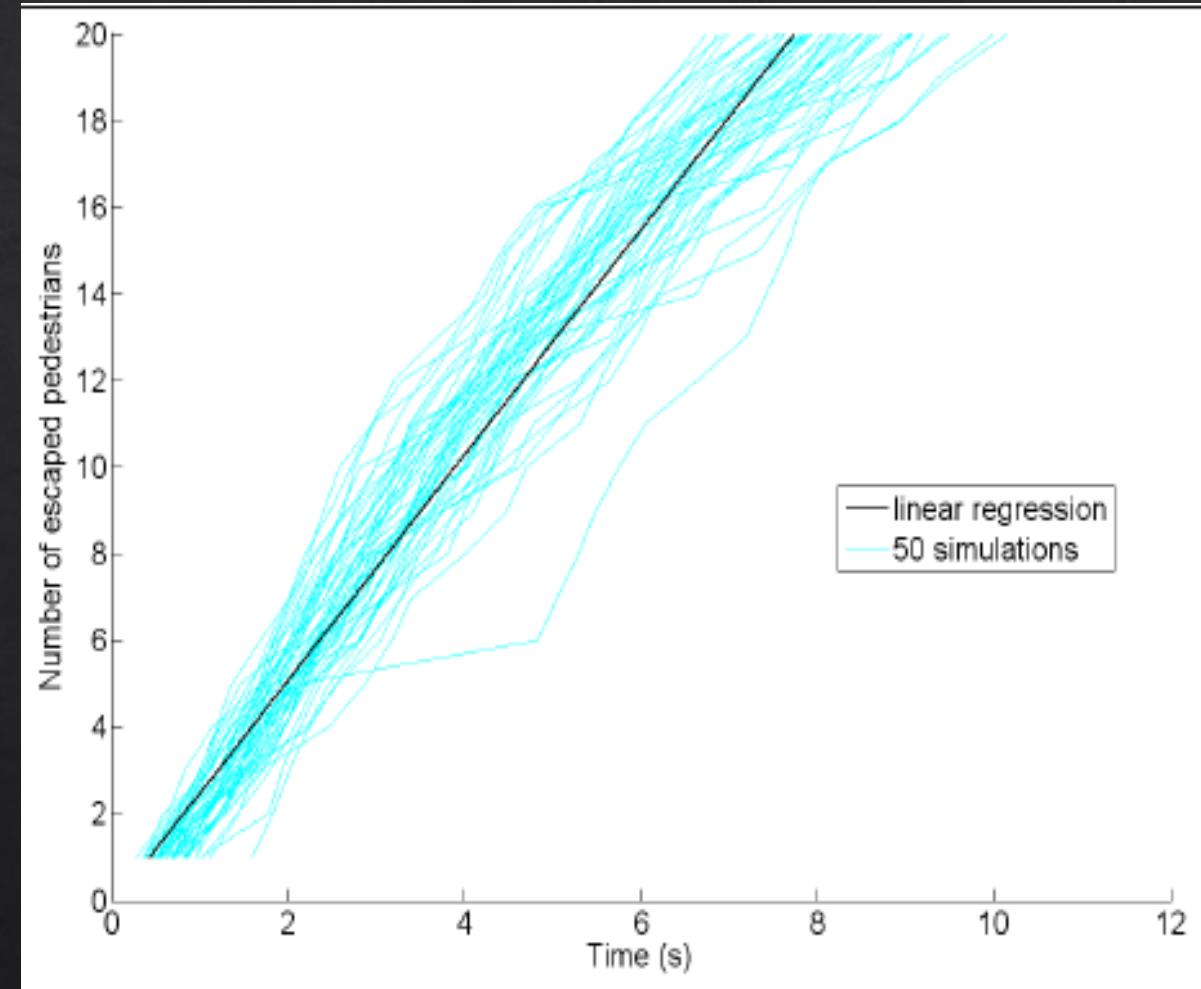
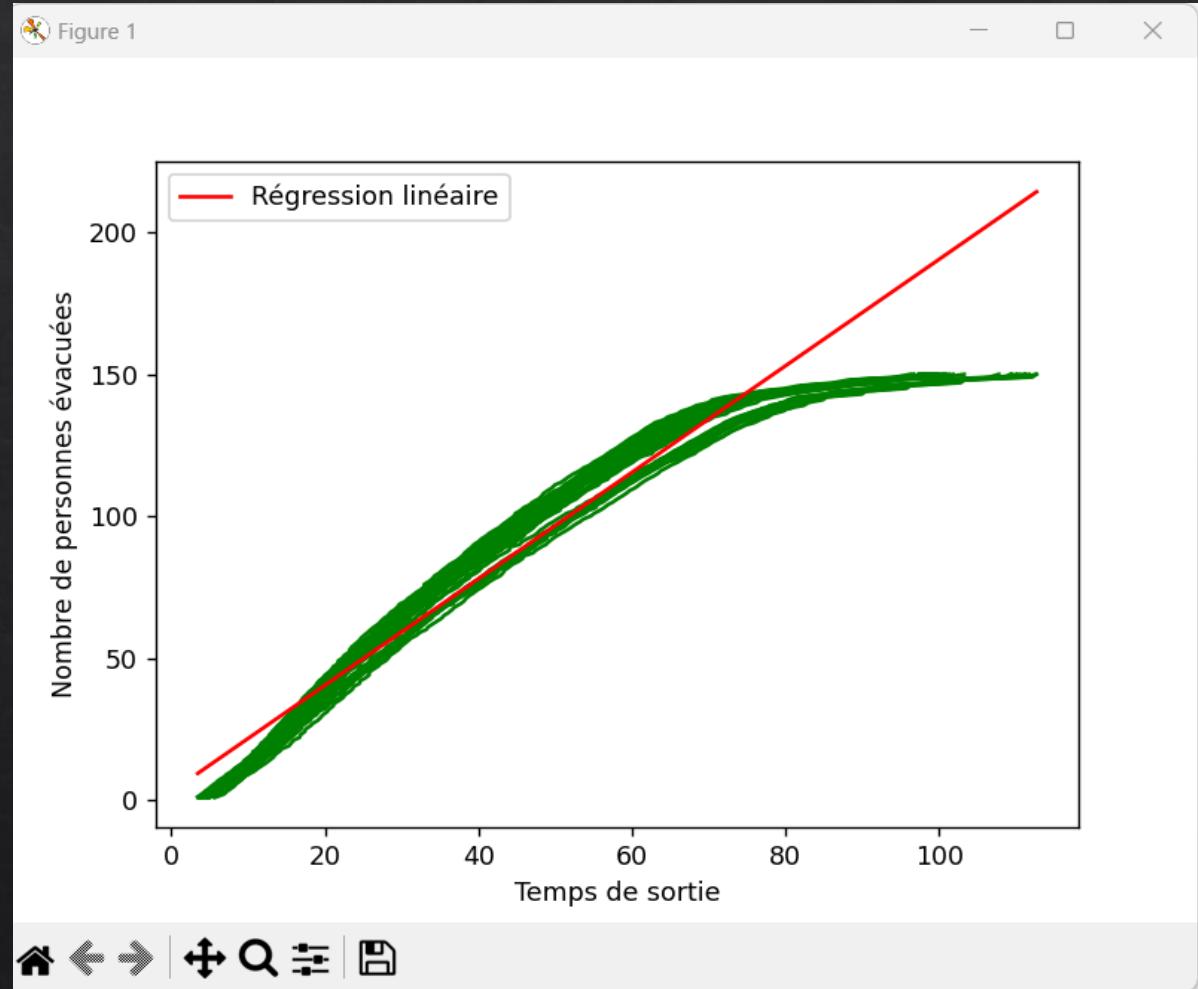
    # Tracer la ligne de régression linéaire
    plt.plot(x_pred, y_pred, color='red', label='Régression linéaire')

    plt.xlabel('Temps de sortie')
    plt.ylabel('Nombre de personnes évacuées')
    plt.legend()
    plt.show()
```

Courbe Obtenu pour : Nombre\_Ball =150 , Nombre Simulation = 50



# Comparons les Résultats



D

- Conclusion

