Simulation d'une foule en fuite

www.mehdimoussaid.com

Photo personnelle

Sommaire

I – Présentation du problème et enjeux

1. Histoire de l'étude des foules

1841 : début de la littérature scientifique (Charles Mackay 1814-1889)

Croissance de l'importance dans la seconde moitié du XX^{ème} siècle

Principaux théoriciens : Scipio Sighele 1868 - 1913
Sigmund Freud 1856 - 1939
Gustave le Bon 1841 - 1931
Gabriel Tarde 1843 - 1904

Théorisation des foules microscopiques

I – Présentation du problème et enjeux

1. Histoire de l’étude des foules

Événements	Date	Nombre de victimes
Bousculades suite à l’effondrement d’un amphithéâtre (Fidènes, Italie)	27 ap-JC	Plusieurs milliers de morts
Bousculade de la rue Royale (Paris)	30 mai 1770	132 morts 800 à 6000 blessés
Tragédie de l'Estadio Nacional (Lima, Pérou)	24 mai 1964	500 morts

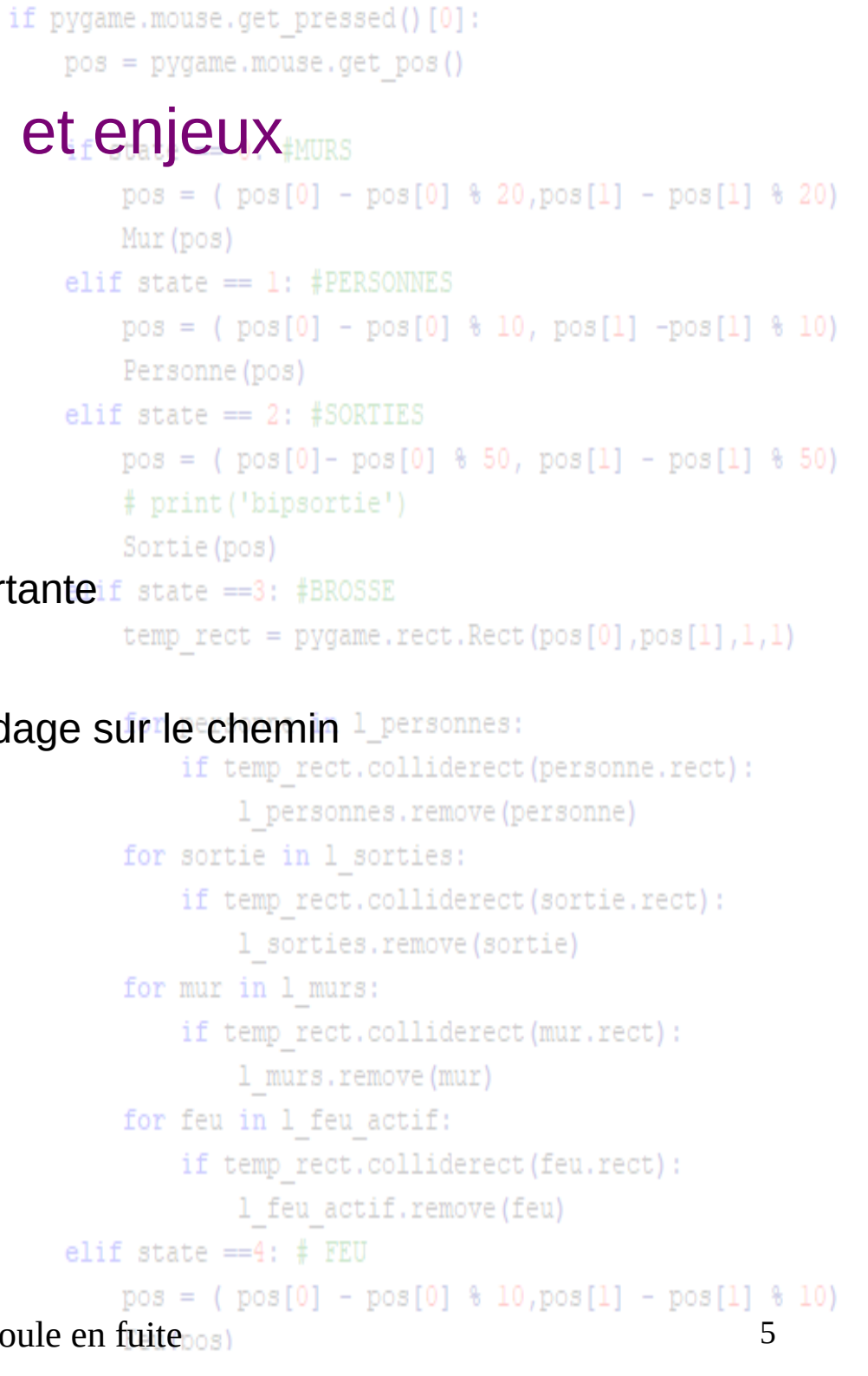
I – Présentation du problème et enjeux

2. Prévention des dommages

- Éviter une densité locale de foule trop importante

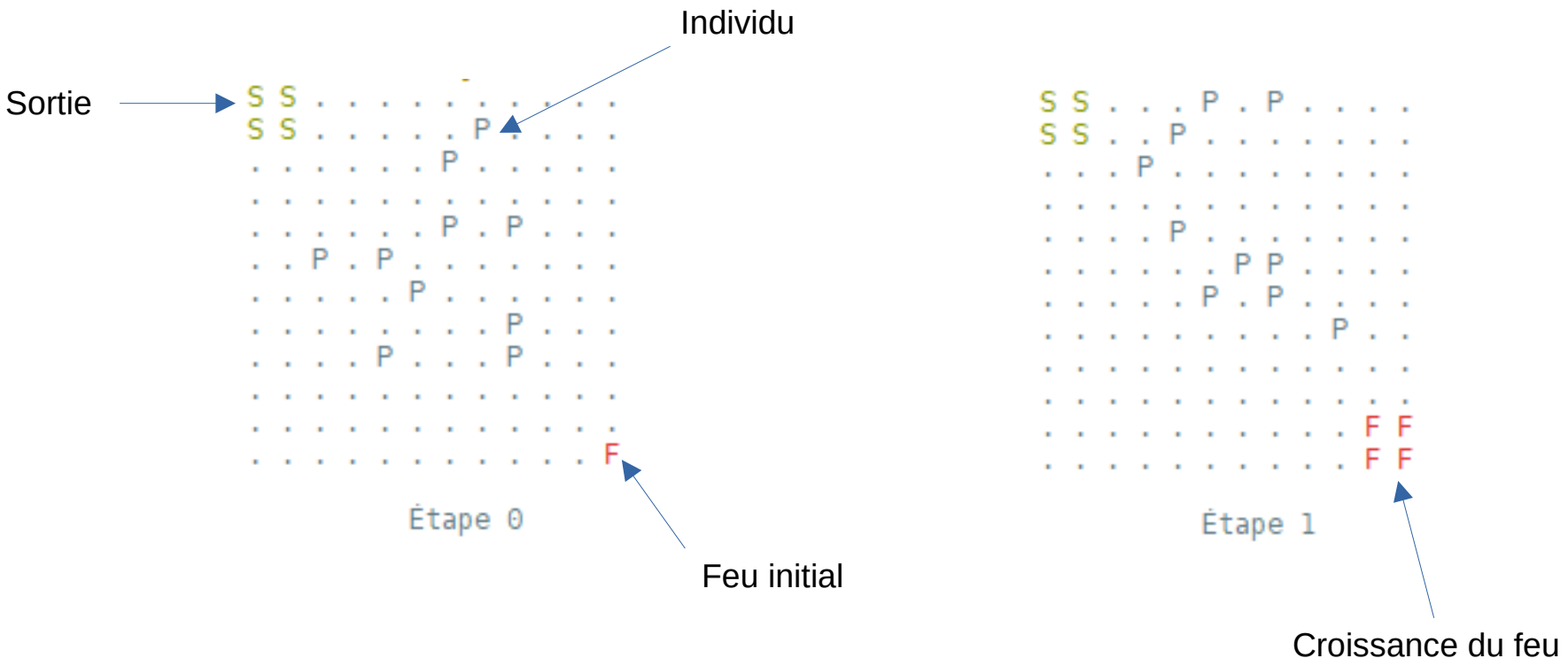
- Fluidifier en disposant des obstacles de guidage sur le chemin

- Placer plusieurs sorties



II - Modélisation et premiers résultats

1. Utilisation de matrices



Fonctionnement du programme matriciel

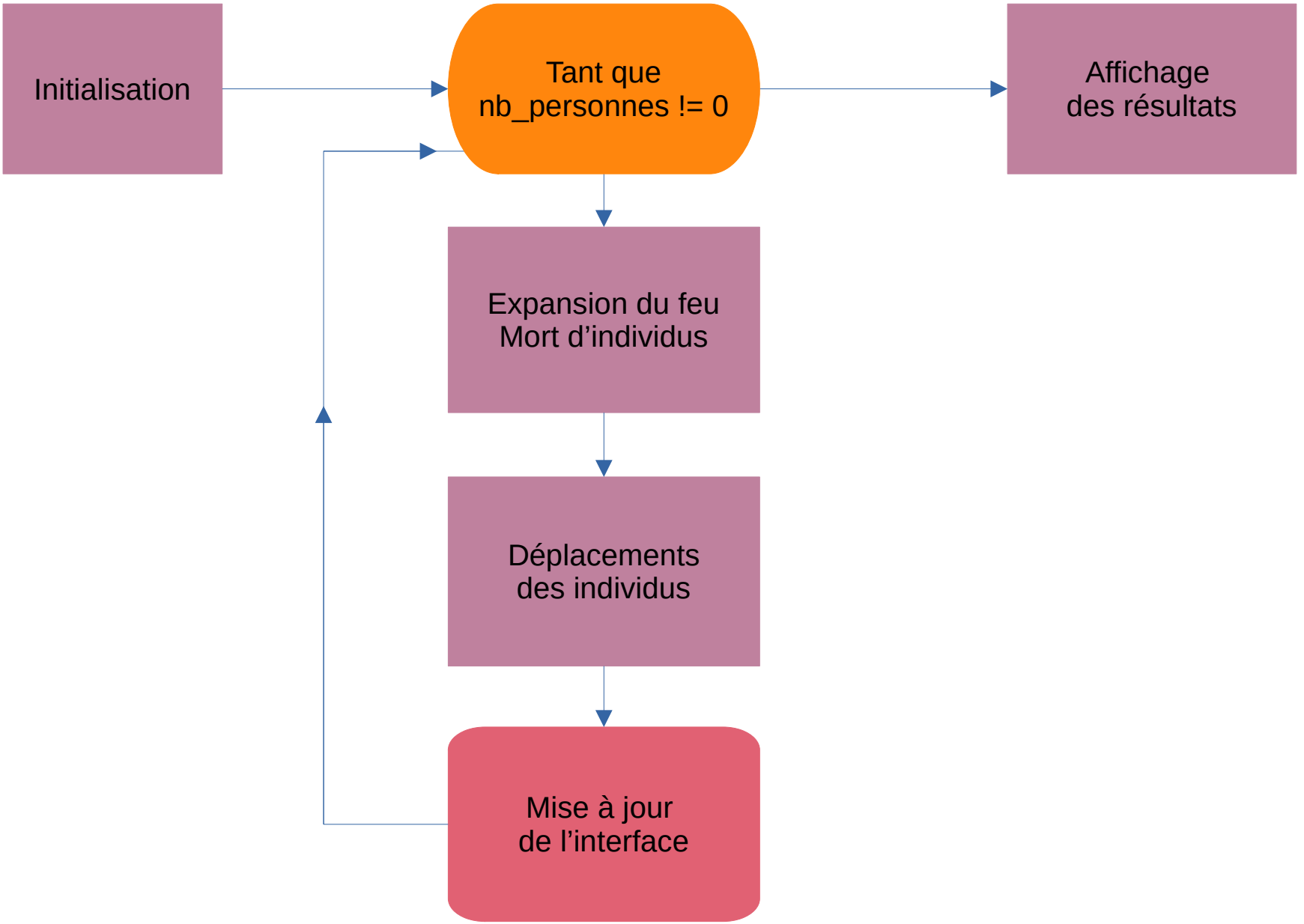
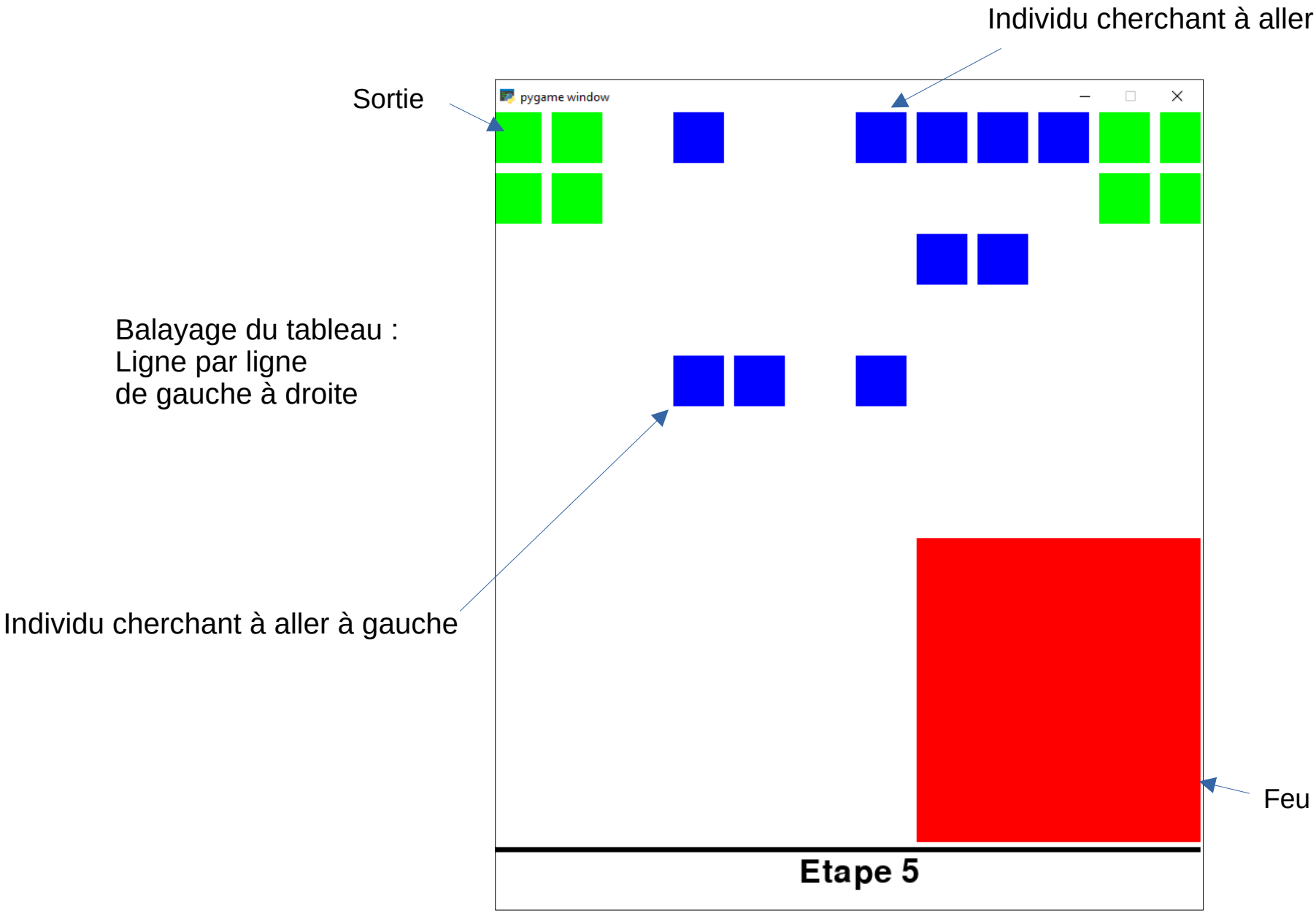
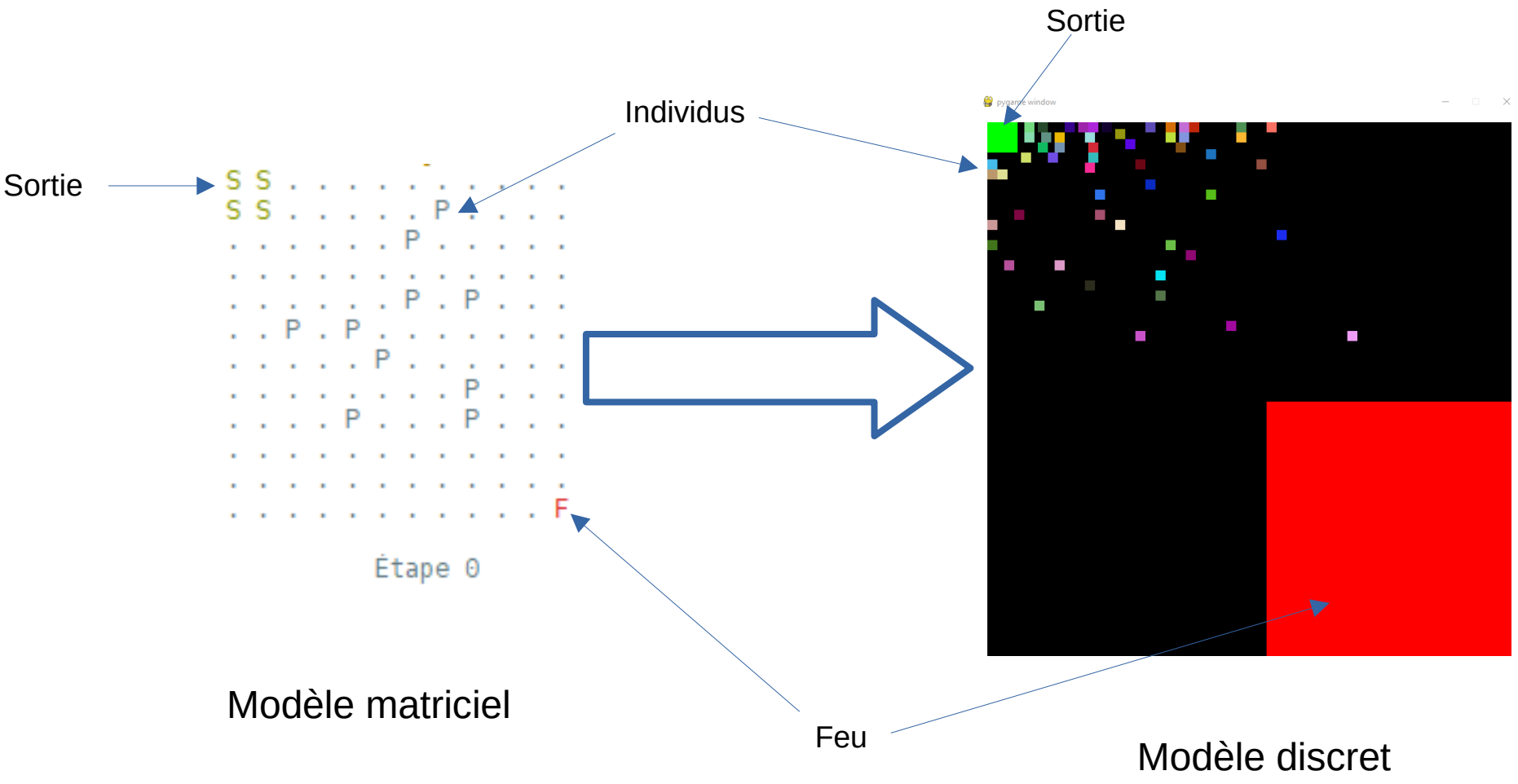


Illustration du problème matriciel



II - Modélisation et premiers résultats

2. Discrétisation de l'espace



III – Programmation orientée objet

1. Histoire et intérêt d'utilisation

- Formalisation en 1990
- C++, python, Java

III – Programmation orientée objet

1. Histoire et intérêt d'utilisation

- Classe
- Objets
- Encapsulation
- Abstraction
- Héritage
- Polymorphisme

```
if pygame.mouse.get_pressed()[0]:
    pos = pygame.mouse.get_pos()

    if state == 0: #MURS
        pos = ( pos[0] - pos[0] % 20,pos[1] - pos[1] % 20)
        Mur(pos)
    elif state == 1: #PERSONNES
        pos = ( pos[0] - pos[0] % 10, pos[1] -pos[1] % 10)
        Personne(pos)
    elif state == 2: #SORTIES
        pos = ( pos[0]- pos[0] % 50, pos[1] - pos[1] % 50)
        # print('bipsortie')
        Sortie(pos)
    elif state ==3: #BROSSE
        temp_rect = pygame.rect.Rect(pos[0],pos[1],1,1)

        for personne in l_personnes:
            if temp_rect.colliderect(personne.rect):
                l_personnes.remove(personne)
        for sortie in l_sorties:
            if temp_rect.colliderect(sortie.rect):
                l_sorties.remove(sortie)
        for mur in l_murs:
            if temp_rect.colliderect(mur.rect):
                l_murs.remove(mur)
        for feu in l_feu_actif:
            if temp_rect.colliderect(feu.rect):
                l_feu_actif.remove(feu)
    elif state ==4: # FEU
        pos = ( pos[0] - pos[0] % 10,pos[1] - pos[1] % 10)
        Feu(pos)
```

III – Programmation orientée objet

2. Implémentation en Python

Définition d'une classe, exemple pour un véhicule

```
class Vehicule:

    def __init__(self, couleur):
        self.couleur = couleur

    def afficher_couleur(self):
        print("La couleur du véhicule est ",self.couleur)
```

III – Programmation orientée objet

2. Implémentation en Python

Création d'un objet

```
class Vehicule:

    def __init__(self, couleur):
        self.__couleur = couleur

    def afficher_couleur(self):
        print("La couleur du véhicule est ",self.__couleur,".", sep = "")

mon_vehicule = Vehicule("jaune")
mon_vehicule.afficher_couleur()
```

Sortie :

```
La couleur du véhicule est jaune.


** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

III – Programmation orientée objet

2. Implémentation en Python

Encapsulation et abstraction

```
class Vehicule:

    def __init__(self, couleur):
        self.__couleur = couleur

    def afficher_couleur(self):
        print("La couleur du véhicule est ",self.__couleur,".", sep = "")

mon_vehicule = Vehicule("jaune")
print(mon_vehicule.__couleur)
```

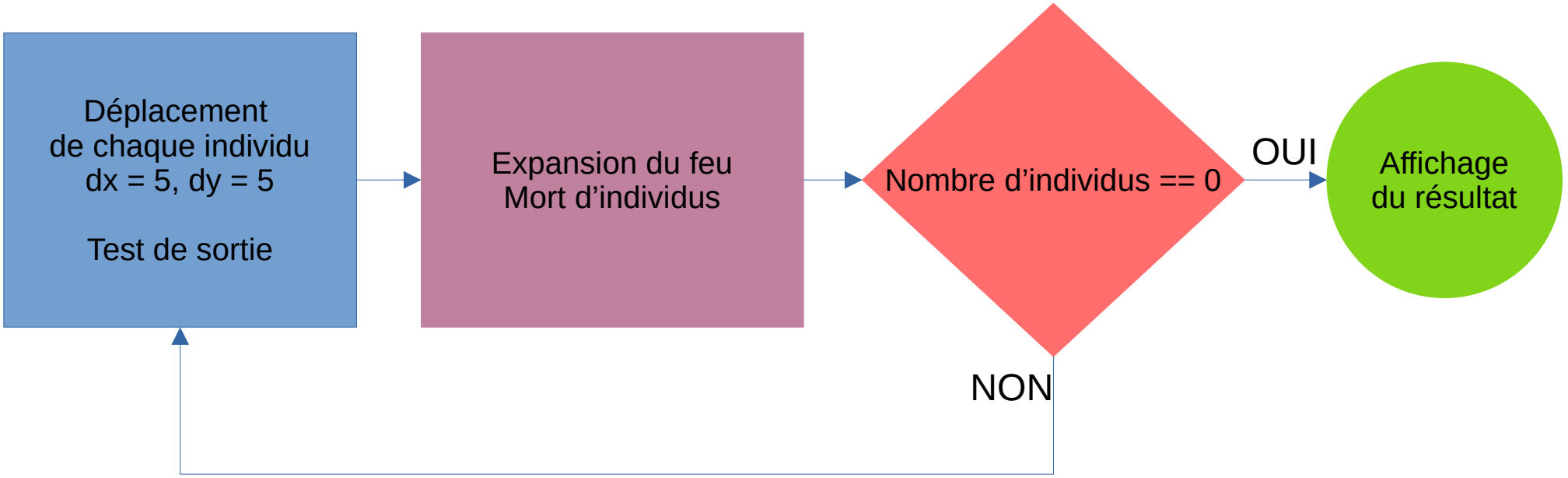
Sortie :

```
Traceback (most recent call last):
  File "main.py", line 11, in <module>
    print(mon_vehicule.__couleur)
AttributeError: 'Vehicule' object has no attribute '__couleur'
```

IV – Algorithmes et résultats

1 -Sortie unique

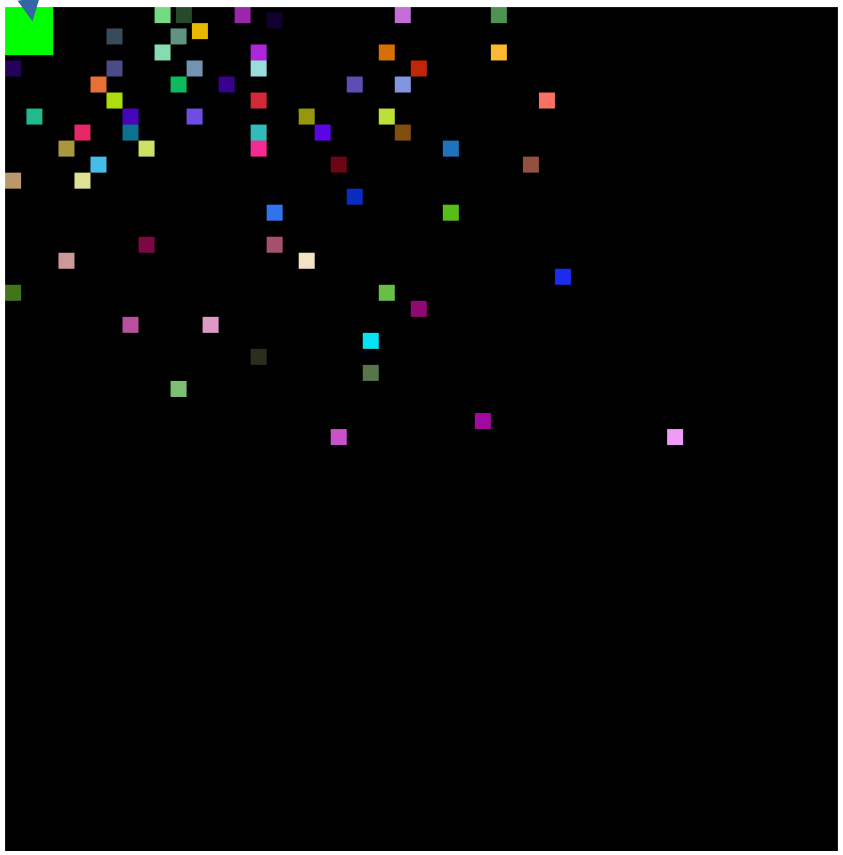
- Algorithme initial



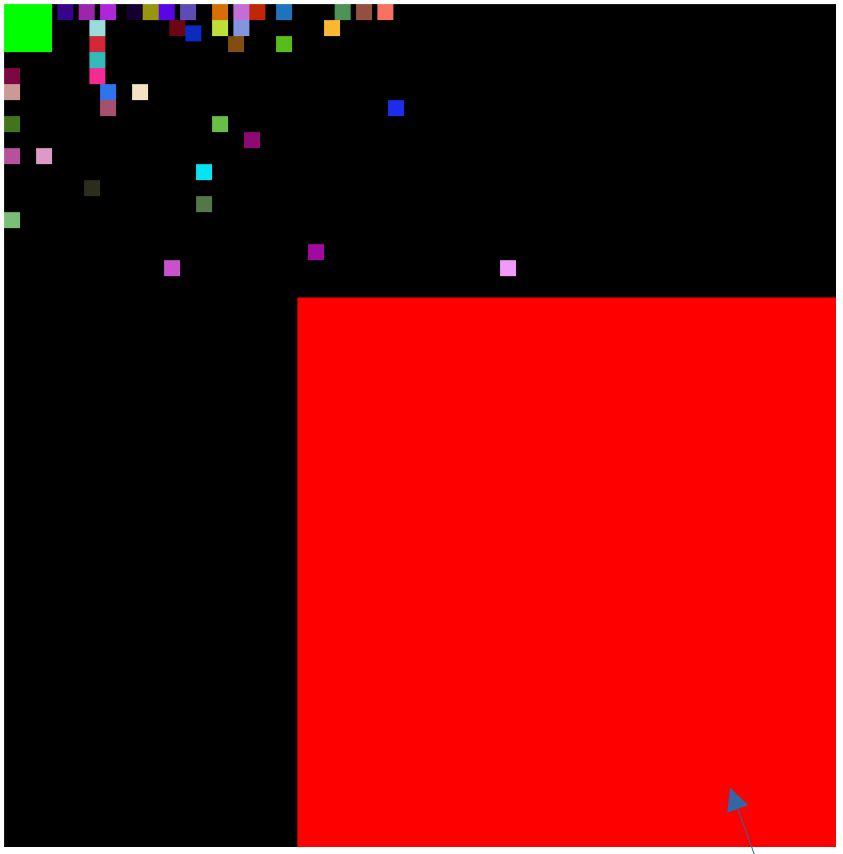
Algorithme initial

Sortie

Individu



Simulation sans feu



Simulation avec feu

Feu

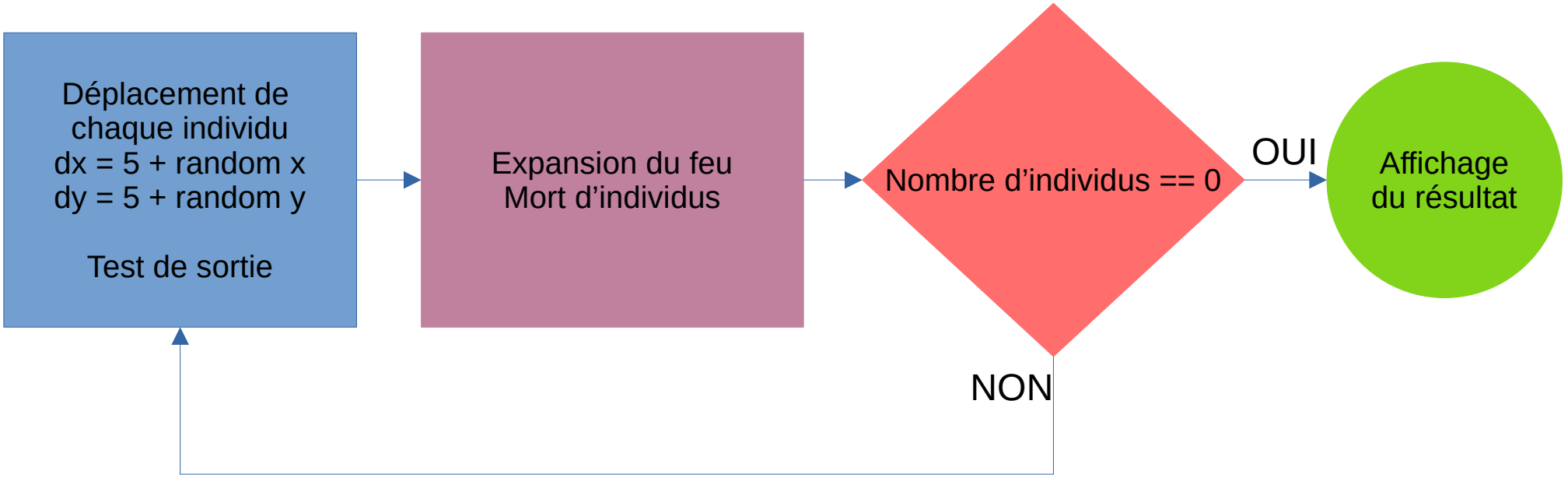
Algorithme initial

Résultat

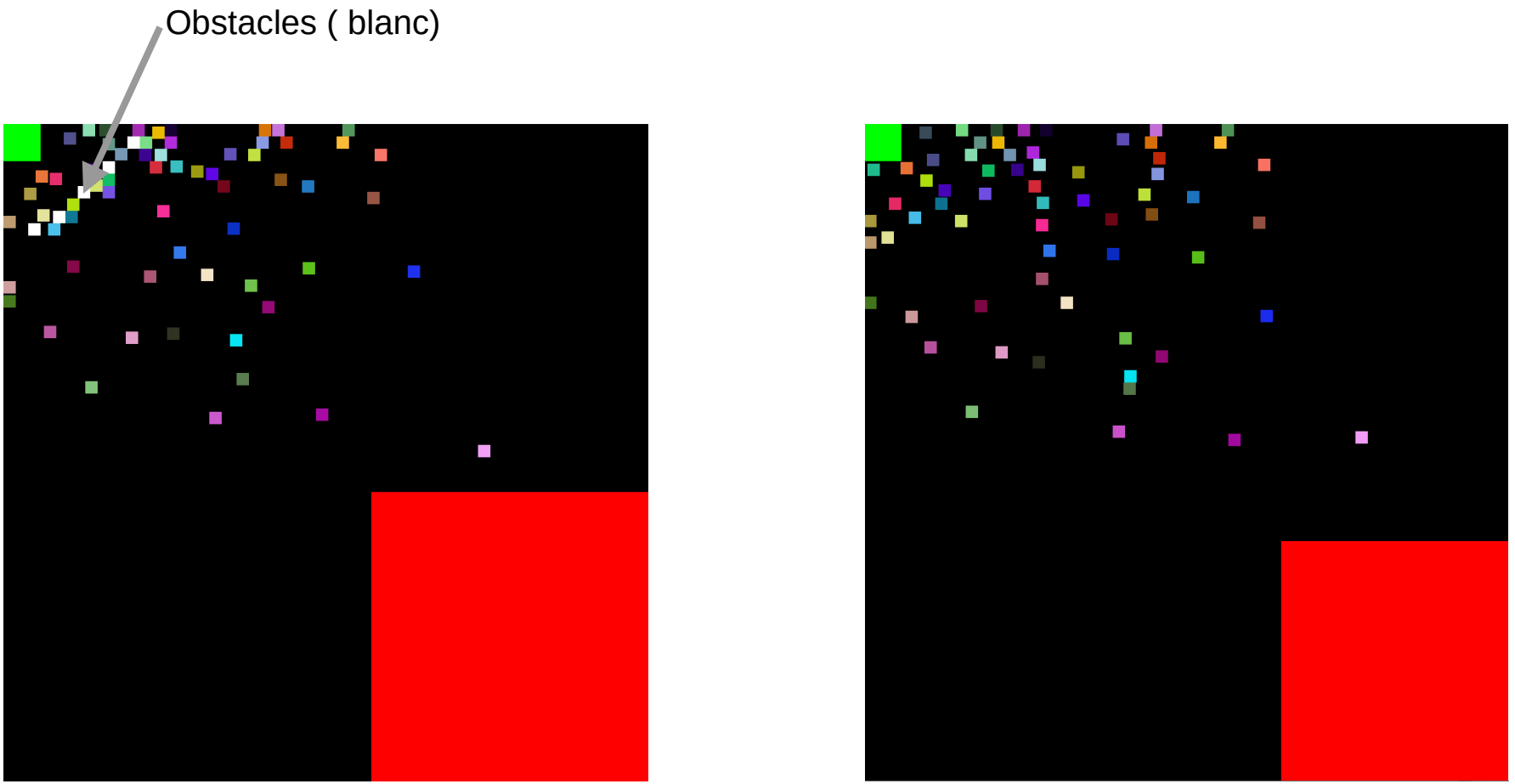
	Avantages	Inconvénients
Premier algorithme	Collisions résolues	Déplacements non naturels

Algorithme avec mouvements aléatoires

- Ensemble de personnes fixé
- Déplacements aléatoires



Algorithme avec mouvements aléatoires



Simulation à deux instants différents

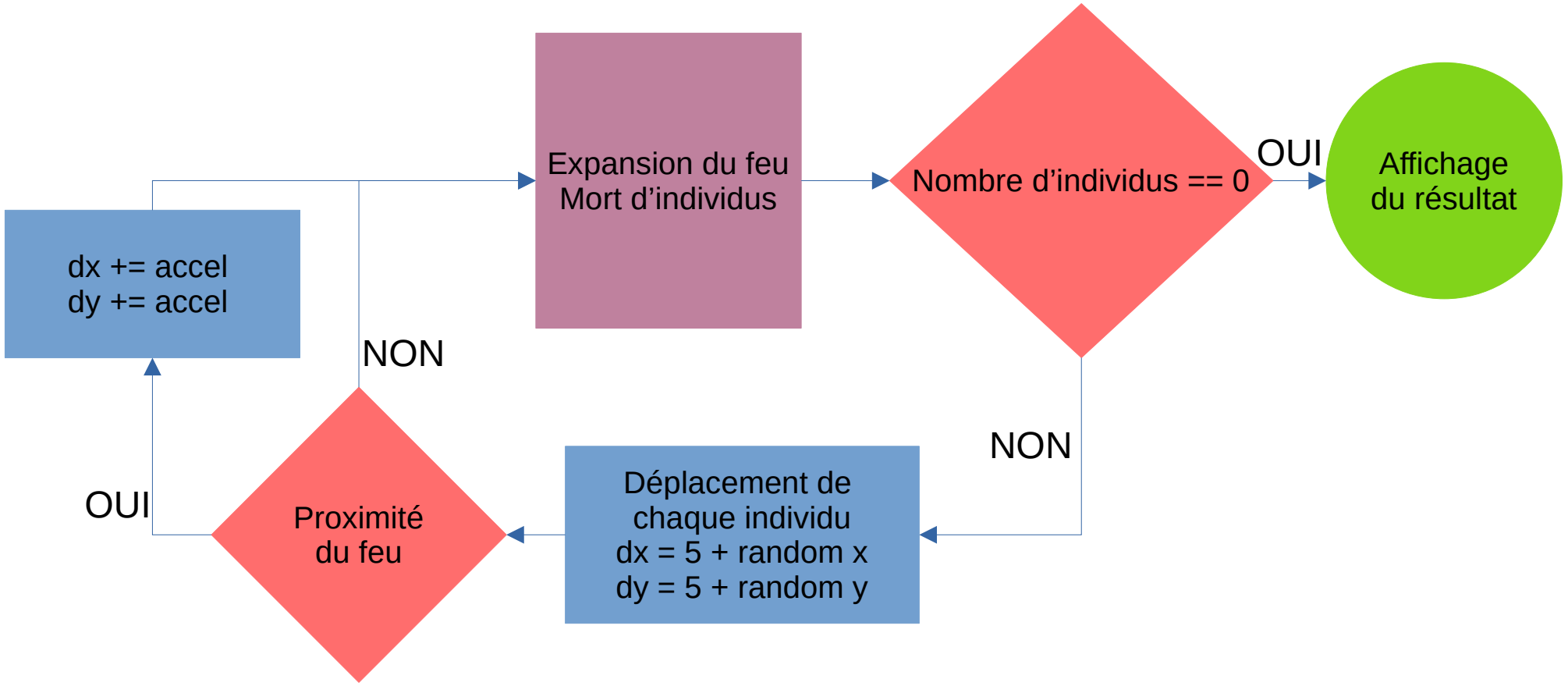
Algorithme avec mouvements aléatoires

Résultats

	Avantages	Inconvénients
Premier algorithme	Collisions résolues	Déplacements non naturels
Deuxième algorithme	Collisions résolues Déplacements plus naturels	Pas de réaction à l'approche du feu

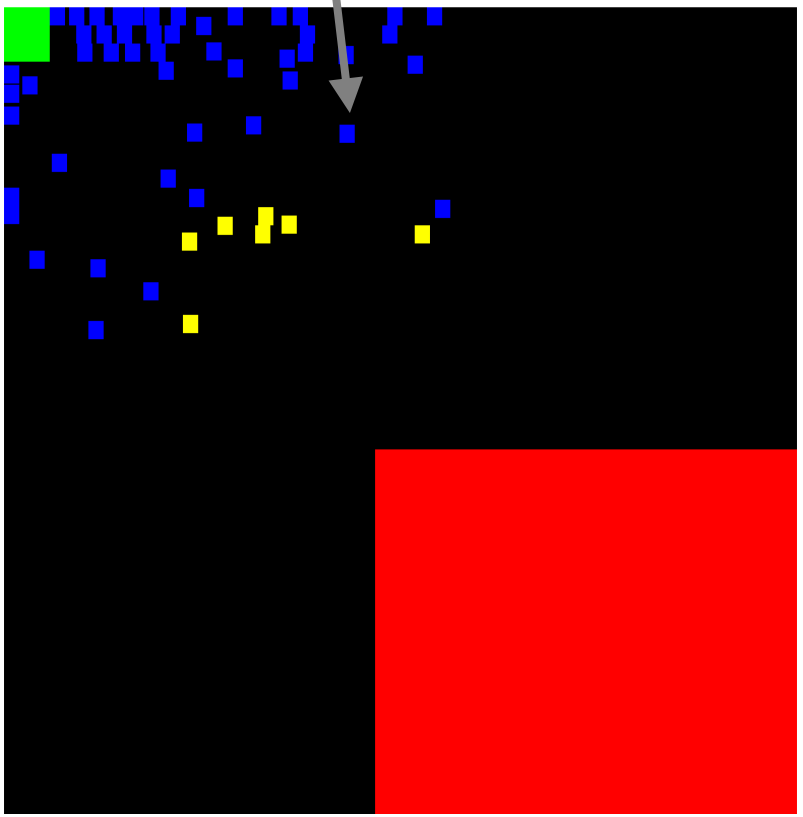
Algorithme avec mouvements aléatoires et accélération (réaction au feu)

- Ensemble de personnes fixé
- Déplacements aléatoires,
- Accélération si proximité du feu

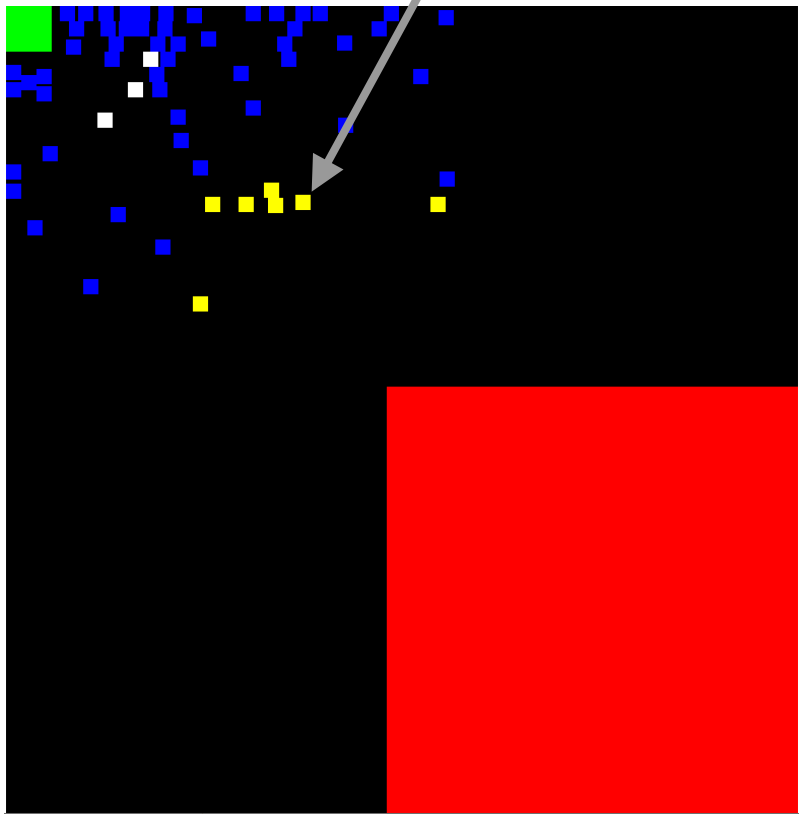


Algorithme avec mouvements aléatoires et accélération

Individu calme (bleu)



Individu en panique (jaune)



Algorithme avec aléatoire et accélération

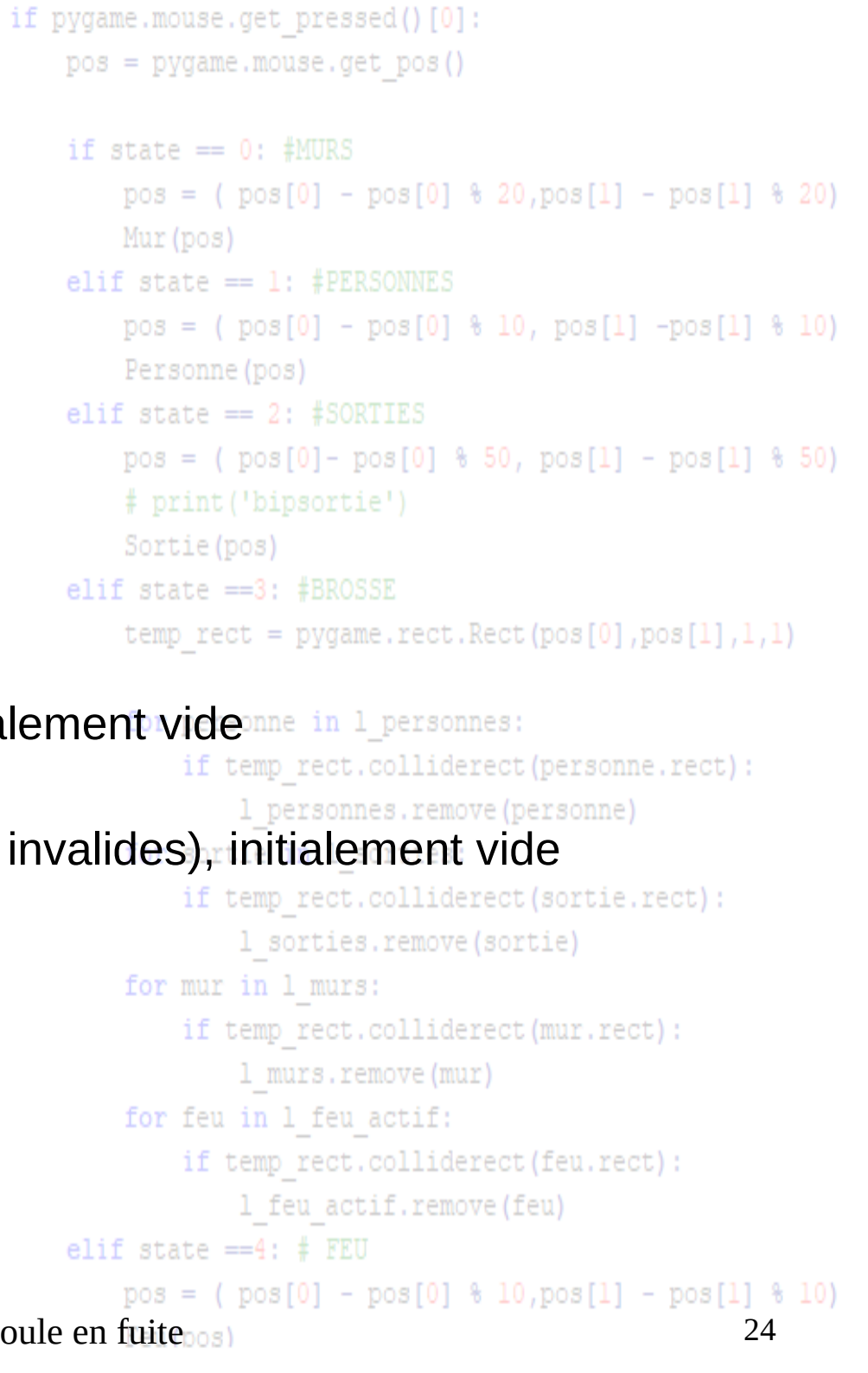
Résultats

	Avantages	Inconvénients
Premier algorithme	Collisions résolues	Déplacements non naturels
Deuxième algorithme	Collisions résolues Déplacements plus naturels	Pas de réaction à l'approche du feu
Troisième algorithme	Collisions résolues Déplacements plus naturels Réaction à la proximité du feu	Une seule sortie

IV - Algorithmes et résultats

2 Sorties multiples

- Recherche de chemin :
  - Adaptation du programme A*
  - Heuristique
  - File ouverte (nœuds à traiter), initialement vide
  - File fermée (nœuds déjà traités ou invalides), initialement vide



On commence par le nœud de départ

Schéma décisionnel de la recherche de chemin

Schéma décisionnel de la recherche de chemin

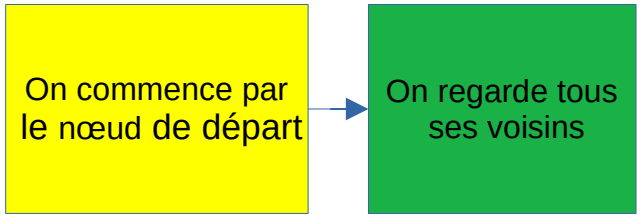


Schéma décisionnel de la recherche de chemin

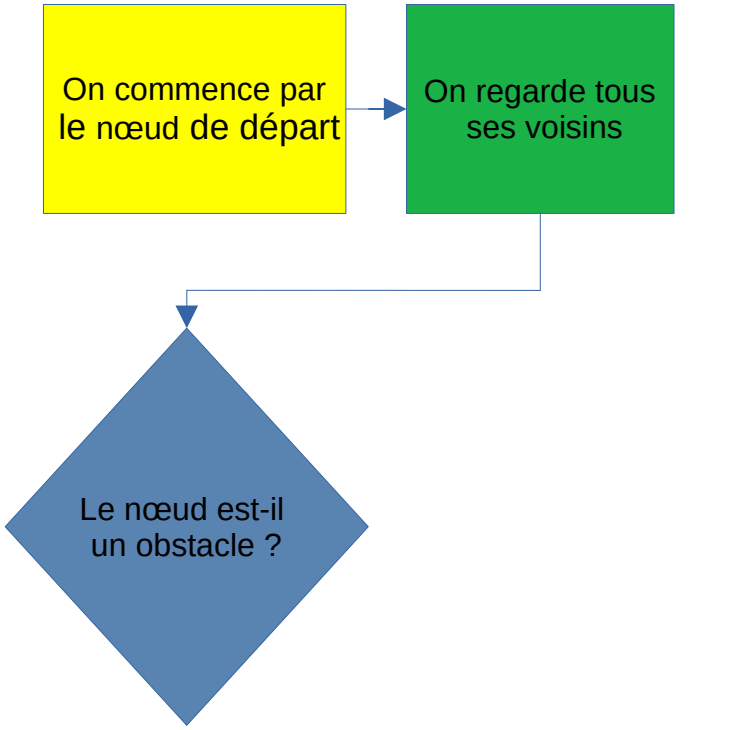


Schéma décisionnel de la recherche de chemin

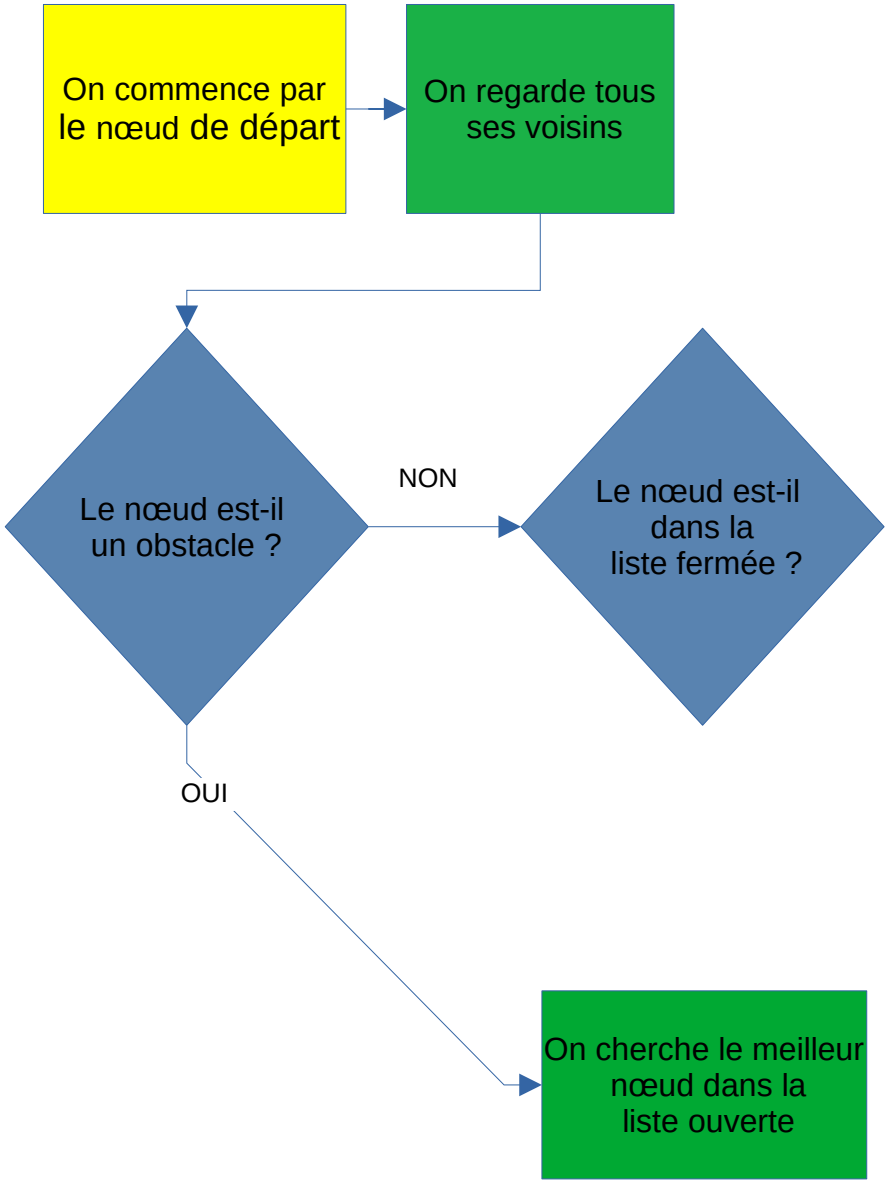


Schéma décisionnel de la recherche de chemin

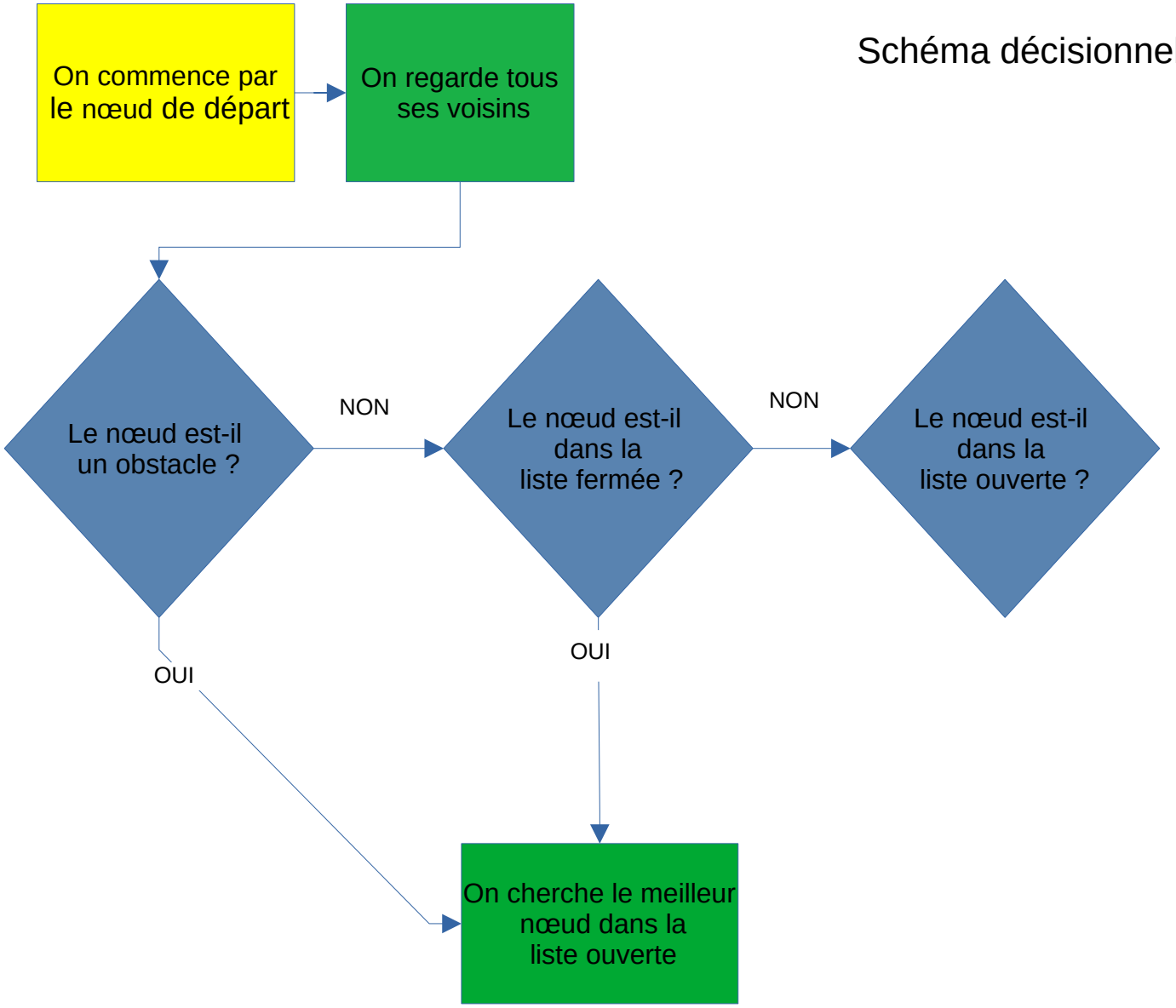


Schéma décisionnel de la recherche de chemin

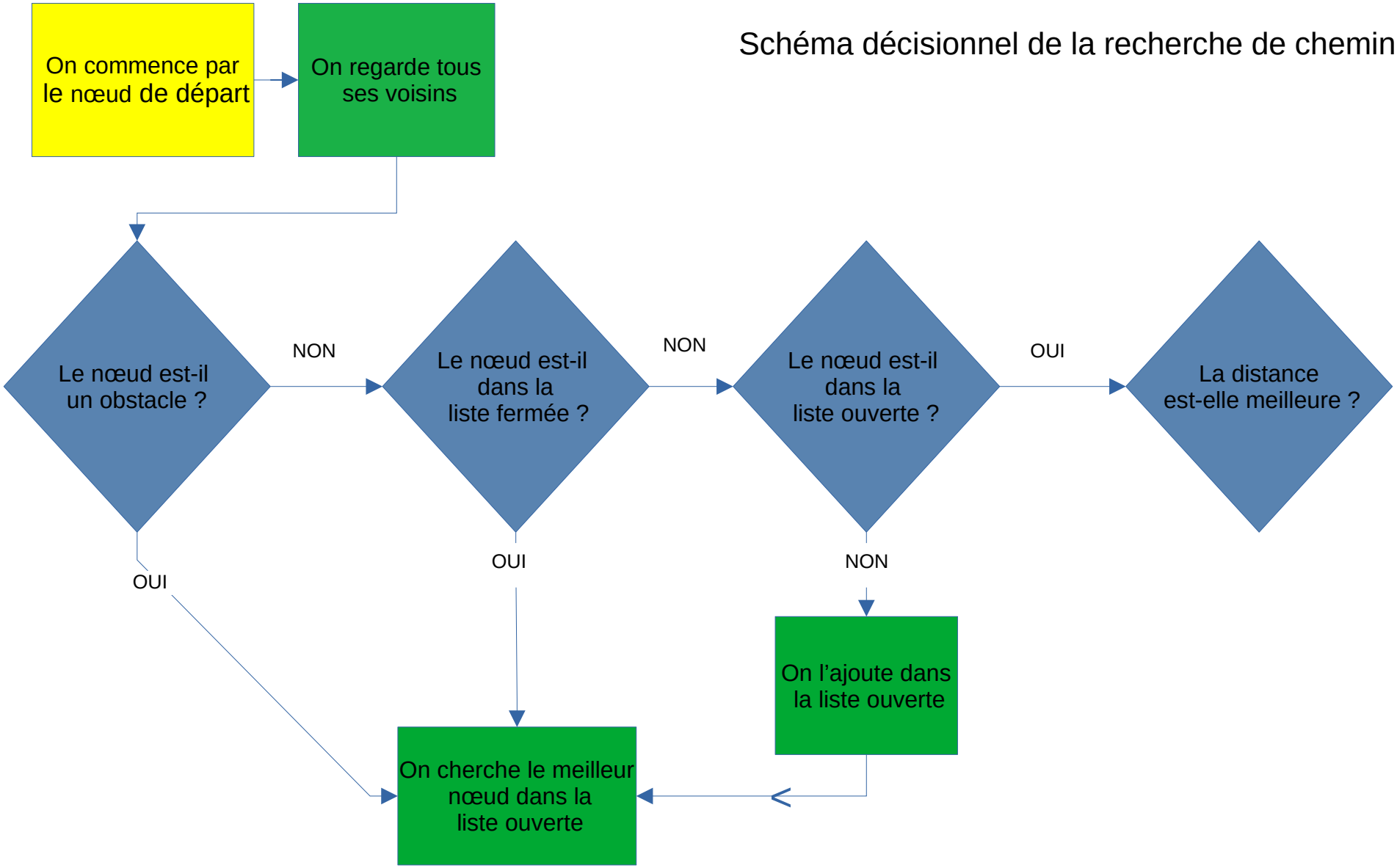


Schéma décisionnel de la recherche de chemin

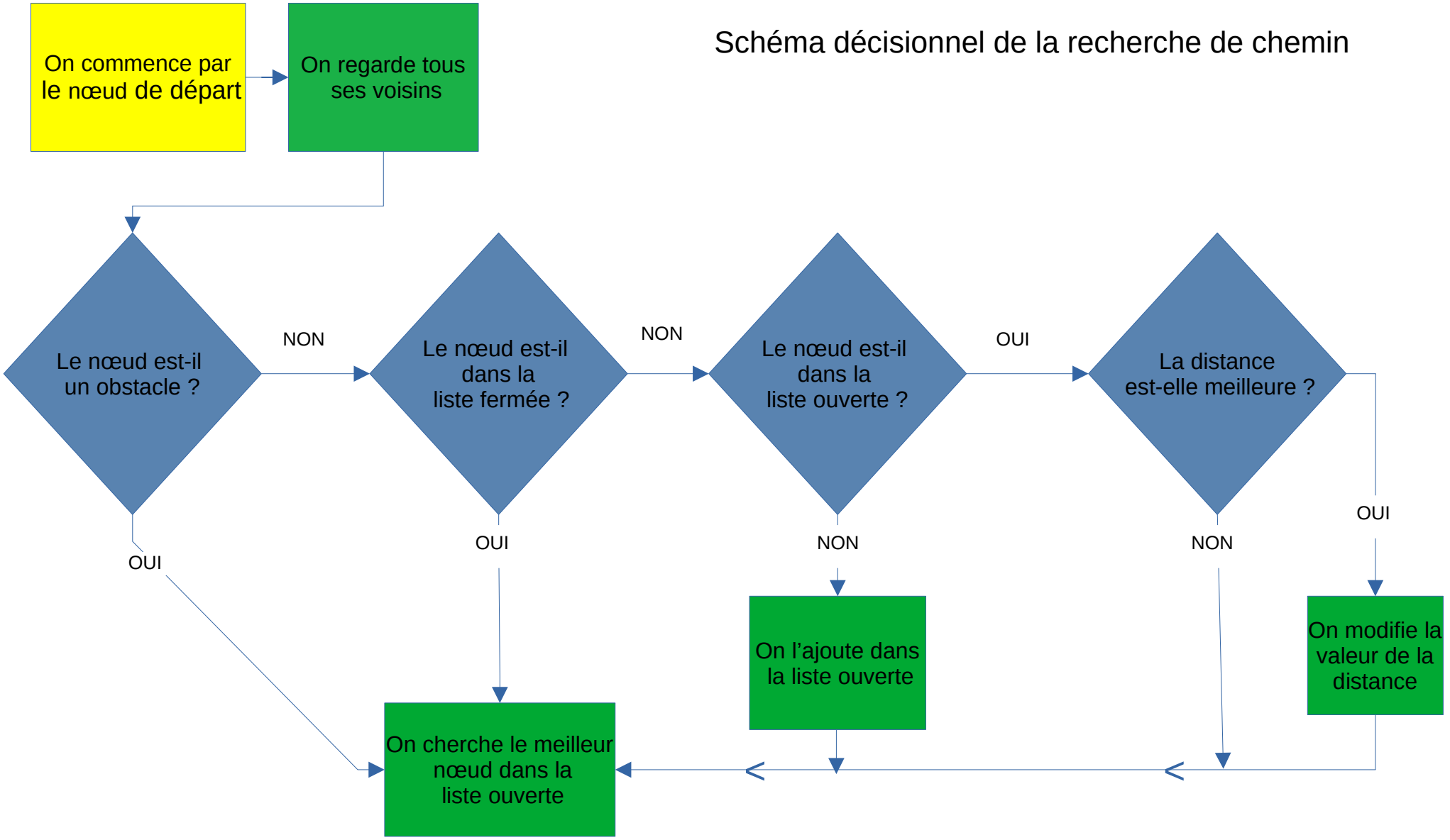


Schéma décisionnel de la recherche de chemin

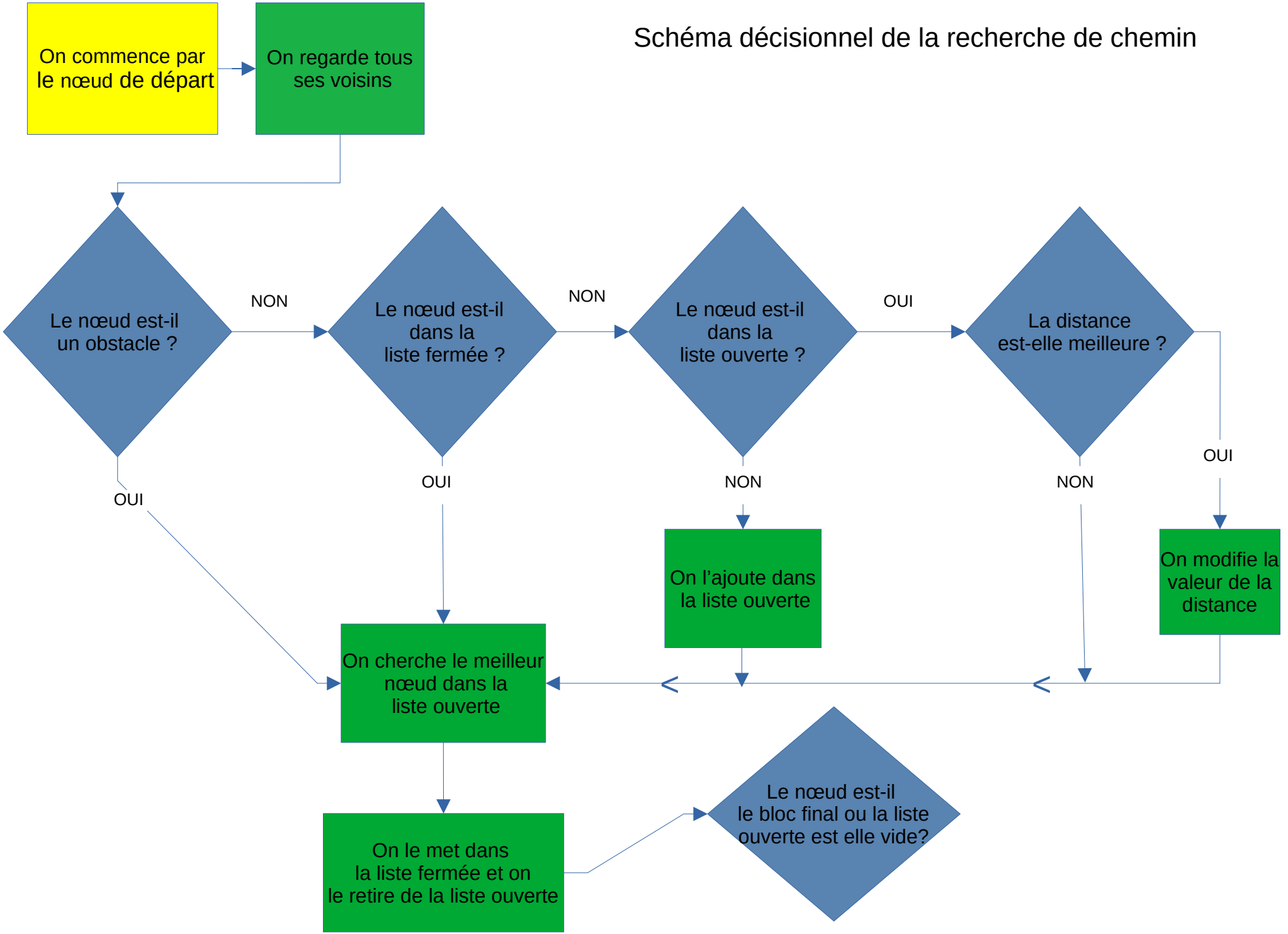
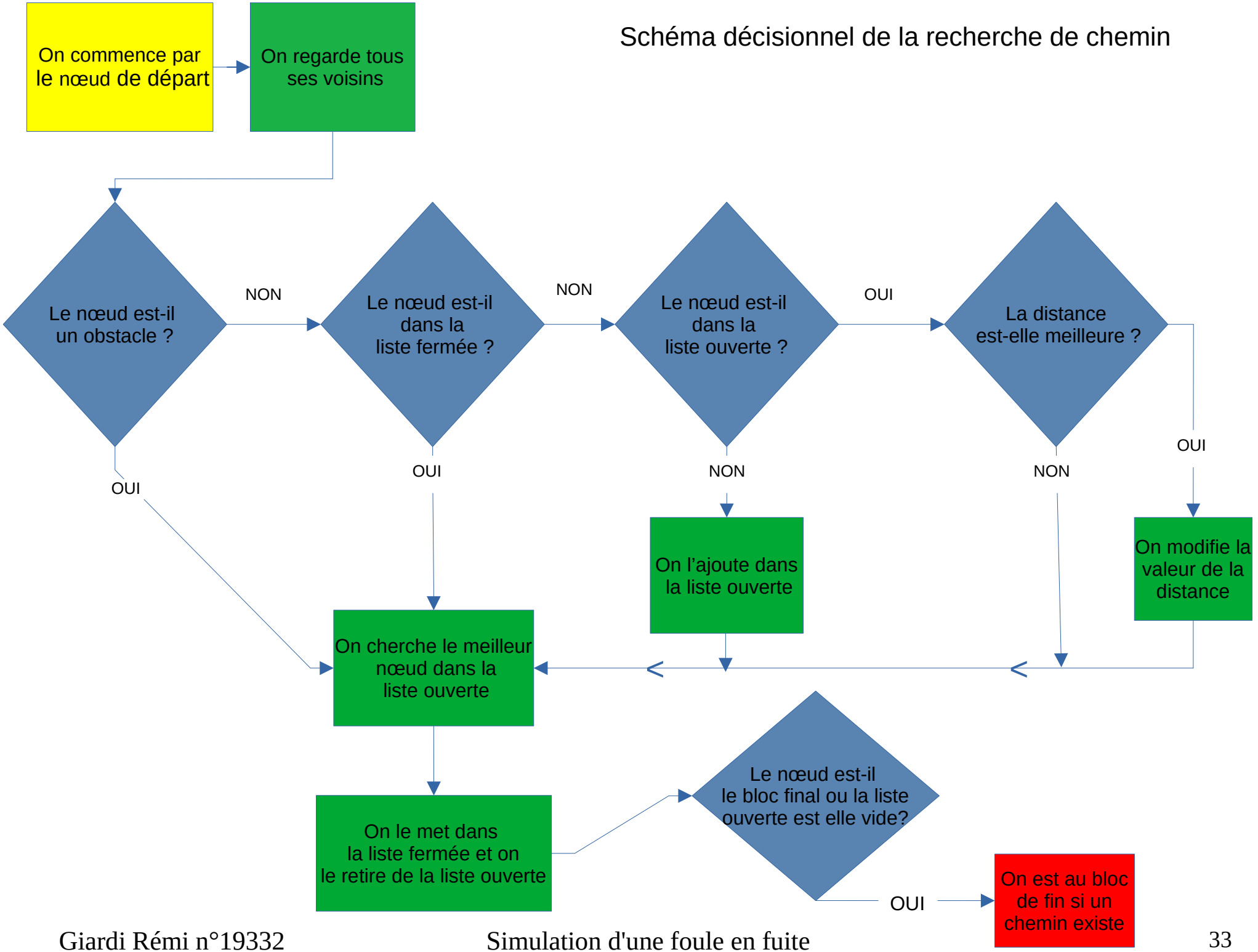


Schéma décisionnel de la recherche de chemin



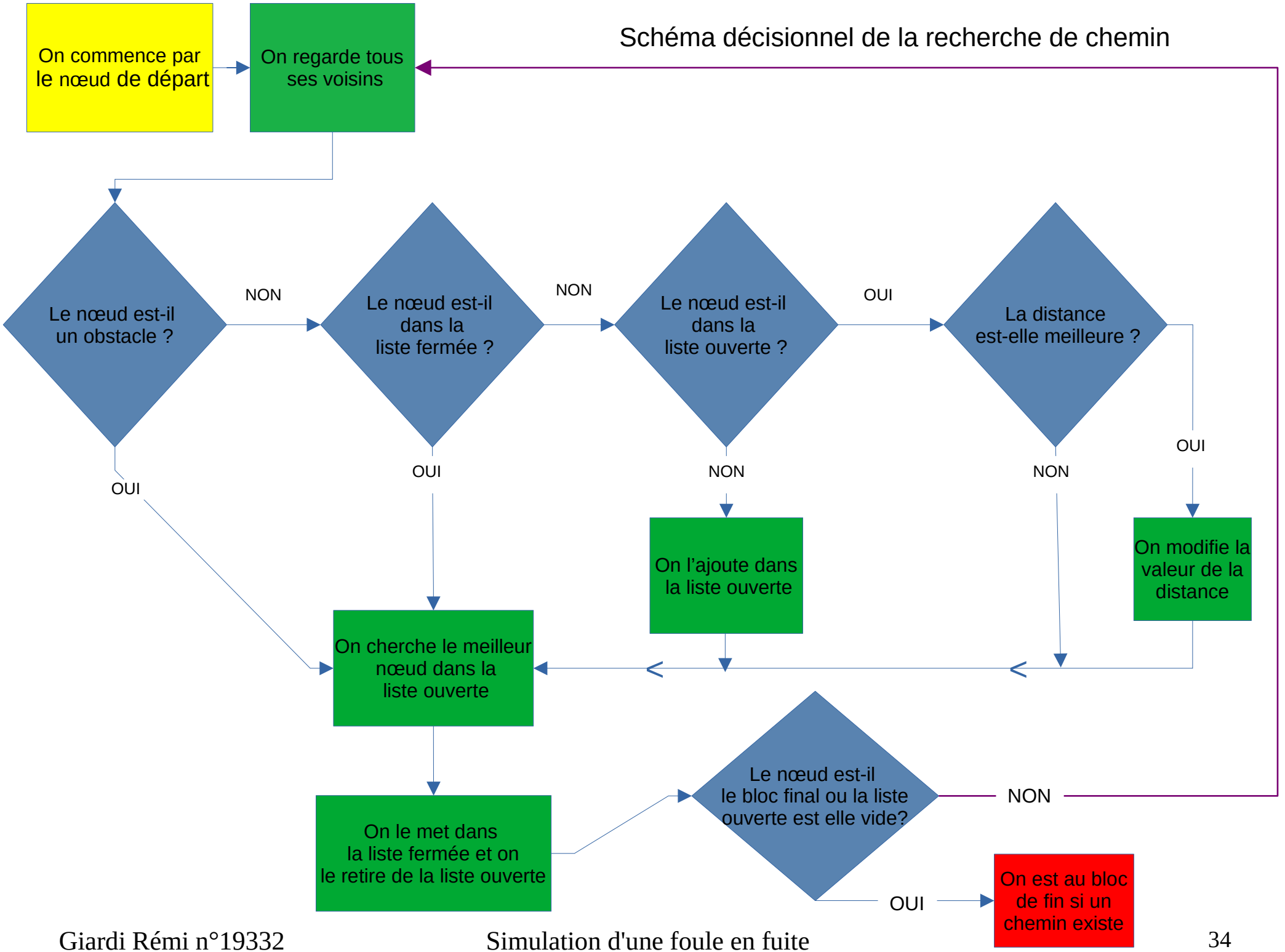
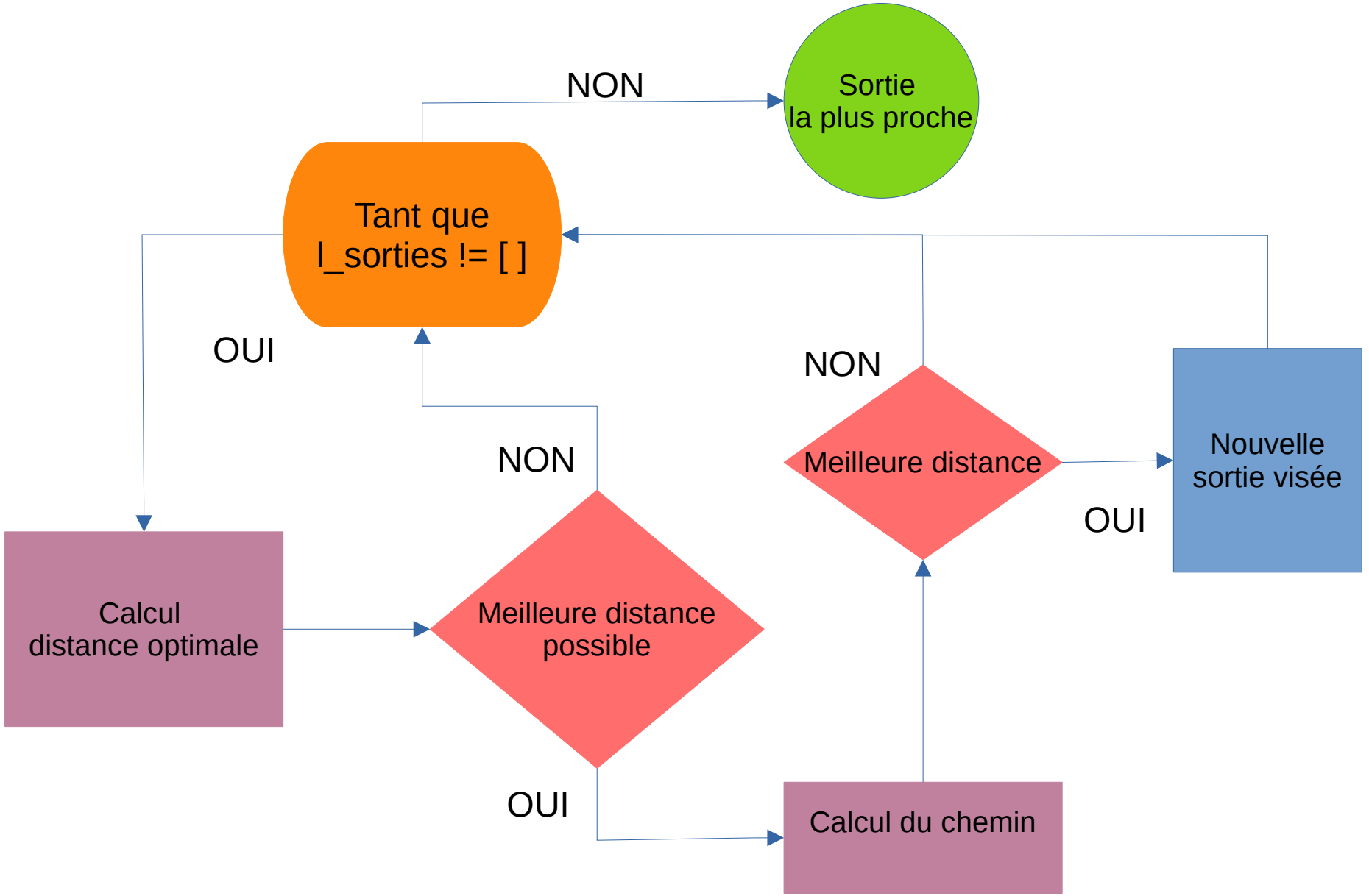
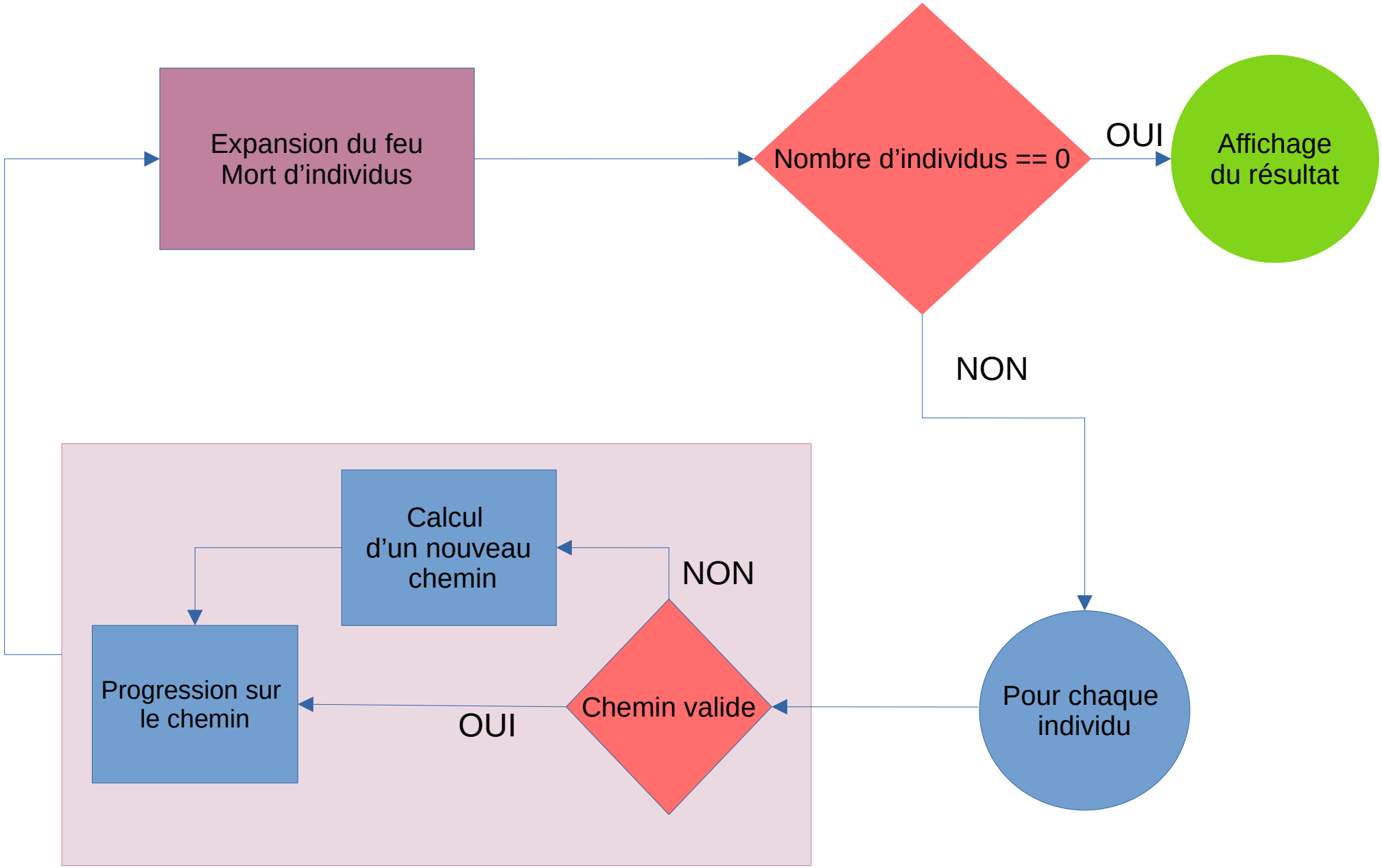


Schéma décisionnel de la recherche de chemin
On commence par le nœud de départ
On regarde tous ses voisins
Le nœud est-il un obstacle ?
NON
Le nœud est-il dans la liste fermée ?
NON
Le nœud est-il dans la liste ouverte ?
OUI
La distance est-elle meilleure ?
OUI
OUI
OUI
NON
NON
On l'ajoute dans la liste ouverte
On modifie la valeur de la distance
On cherche le meilleur nœud dans la liste ouverte
On le met dans la liste fermée et on le retire de la liste ouverte
Le nœud est-il le bloc final ou la liste ouverte est elle vide?
NON
OUI
On est au bloc de fin si un chemin existe

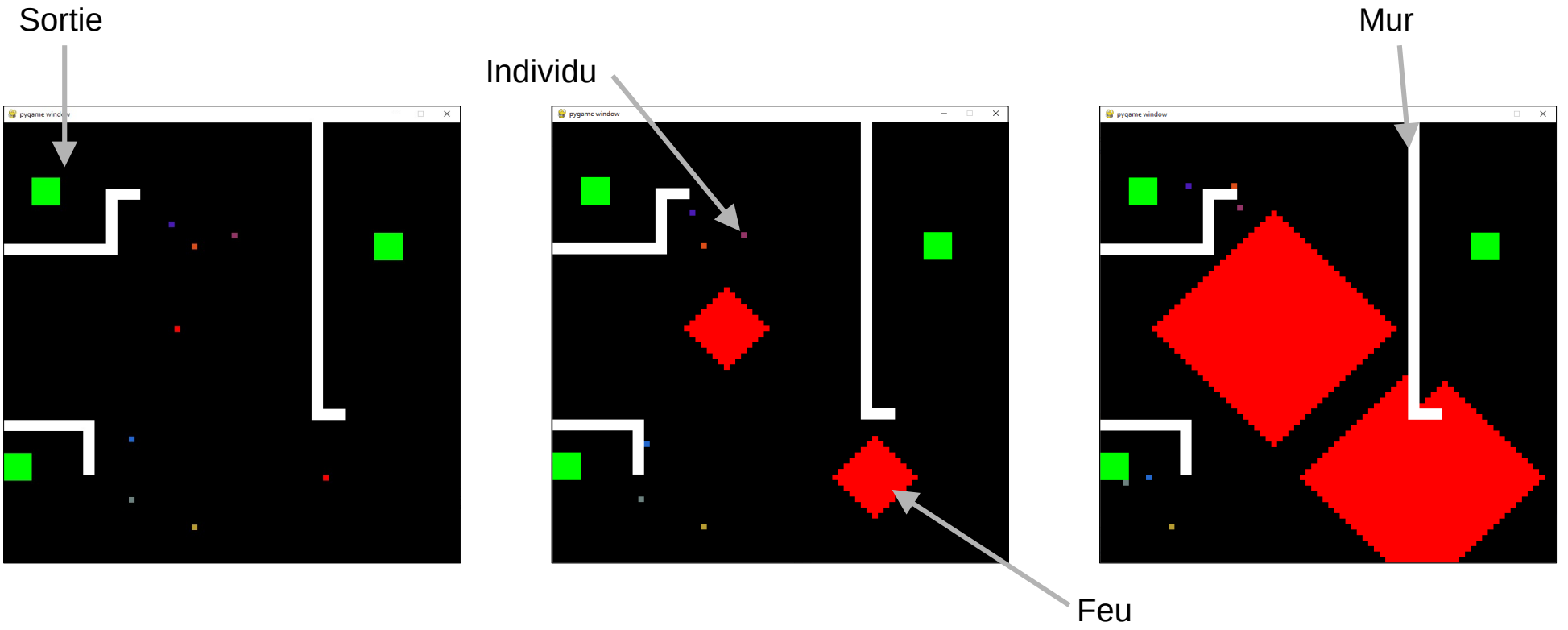
Algorithme à plusieurs sorties : (Recherche de chemin)



Algorithme à plusieurs sorties : (boucle principale)



Algorithme à plusieurs sorties (propagation du feu)



Étape initiale

Étape n°10

Étape n°20

Algorithme à plusieurs sorties

Résultats

	Avantages	Inconvénients
Premier algorithme	Collisions résolues	Déplacements non naturels
Deuxième algorithme	Collisions résolues Déplacements plus naturels	Pas de réaction à l'approche du feu
Troisième algorithme	Collisions résolues Déplacements plus naturels Réaction à la proximité du feu	Une seule sortie
Quatrième algorithme	Collisions résolues Déplacements plus naturels Réaction à la proximité du feu Plusieurs sorties	Moins rapide à exécuter

V - Conclusion

Algorithme utilisé	Résultats (ratio de survivants)	Résultats (temps de calcul)
Algorithme à une sortie sans obstacle	62 %	100 %
Algorithme à une sortie avec obstacles	78 %	90 %
Algorithme à plusieurs sorties sans obstacle	75 %	180 %
Algorithme à plusieurs sorties avec obstacles	89 %	175 %

V - Conclusion

- Simulation microscopique d’une foule
- Modélisation de lieux
- Influence des obstacles sur le guidage

```
if pygame.mouse.get_pressed()[0]:
    pos = pygame.mouse.get_pos()

    if state == 0: #MURS
        pos = ( pos[0] - pos[0] % 20,pos[1] - pos[1] % 20)
        Mur(pos)
    elif state == 1: #PERSONNES
        pos = ( pos[0] - pos[0] % 10, pos[1] -pos[1] % 10)
        Personne(pos)
    elif state == 2: #SORTIES
        pos = ( pos[0]- pos[0] % 50, pos[1] - pos[1] % 50)
        # print('bipsortie')
        Sortie(pos)
    elif state ==3: #BROSSE
        temp_rect = pygame.rect.Rect(pos[0],pos[1],1,1)

        for personne in l_personnes:
            if temp_rect.colliderect(personne.rect):
                l_personnes.remove(personne)
        for sortie in l_sorties:
            if temp_rect.colliderect(sortie.rect):
                l_sorties.remove(sortie)
        for mur in l_murs:
            if temp_rect.colliderect(mur.rect):
                l_murs.remove(mur)
        for feu in l_feu_actif:
            if temp_rect.colliderect(feu.rect):
                l_feu_actif.remove(feu)
    elif state ==4: # FEU
        pos = ( pos[0] - pos[0] % 10,pos[1] - pos[1] % 10)
        Feu(pos)
```

Bibliographie

- L.F. Henderson, « *The statistics of crowd fluids* »

Nature, Nature Publishing Group, vol. 229, 1971, p. 381–383

- Aude Roudneff. « *Modélisation macroscopique de mouvements de foule* ».

Université Paris Sud - Paris XI, 2011.

- Nicolas Bain, Denis Bartolo. « *Dynamic response and hydrodynamics of polarized crowds* » Science,

4 janvier 2019.

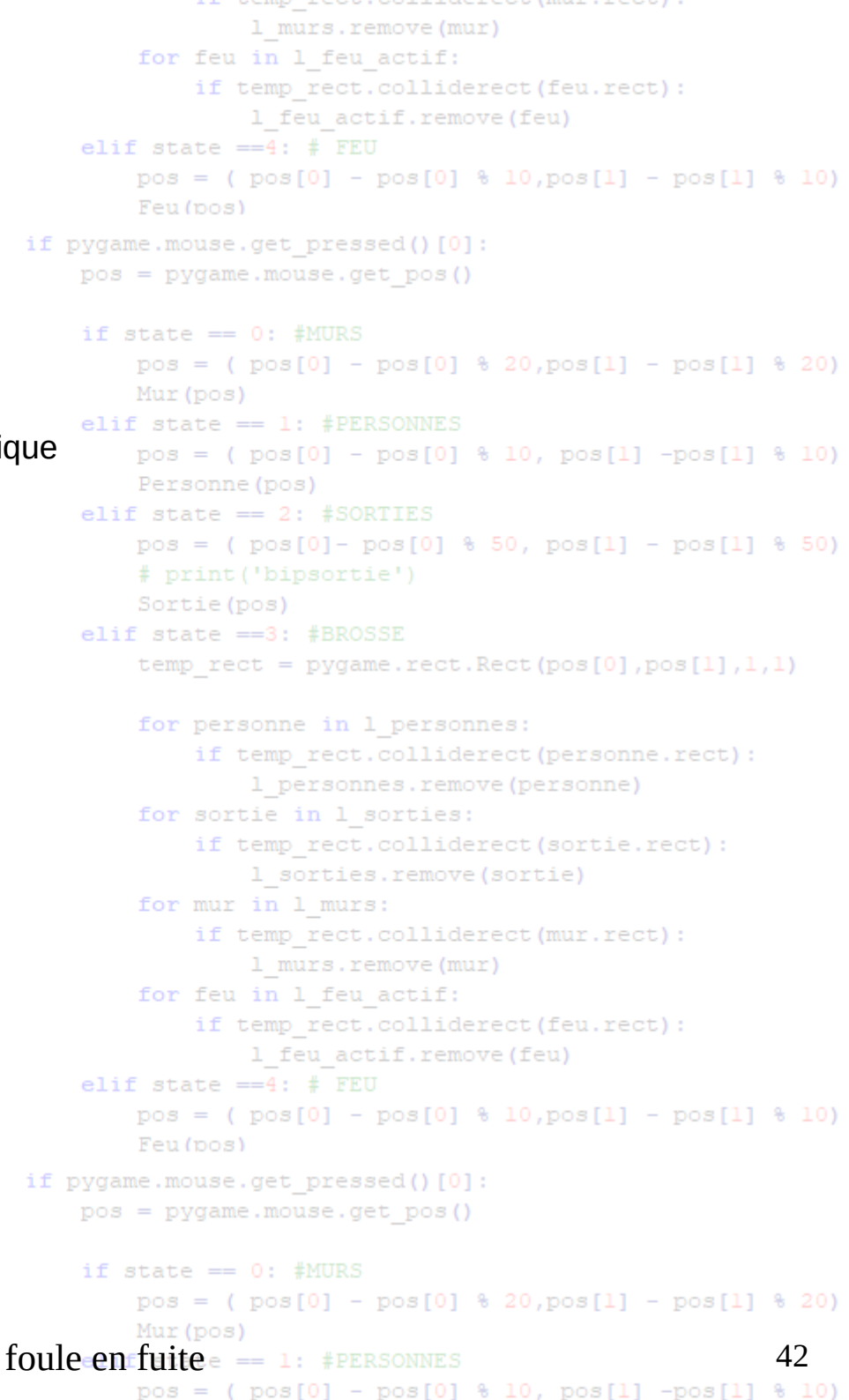
- Craig W. Reynolds, « *Flocks, herds and schools: A distributed behavioral model* »

ACM SIGGRAPH Computer Graphics, vol. 21, no 4, Juillet 1987, p. 25-34

- Dirk Helbing, Illés J. Farkas, Péter Molnár, Tamás Vicsek « *Simulation of Pedestrian Crowds in Normal and Evacuation Situations* »

Research Gate, Janvier 2002, p. 5-6

Sommaire de l’annexe



```
                l_murs.remove(mur)
            for feu in l_feu_actif:
                if temp_rect.colliderect(feu.rect):
                    l_feu_actif.remove(feu)
    elif state ==4: # FEU
        pos = ( pos[0] - pos[0] % 10,pos[1] - pos[1] % 10)
        Feu(pos)
if pygame.mouse.get_pressed()[0]:
    pos = pygame.mouse.get_pos()

    if state == 0: #MURS
        pos = ( pos[0] - pos[0] % 20,pos[1] - pos[1] % 20)
        Mur(pos)
    elif state == 1: #PERSONNES
        pos = ( pos[0] - pos[0] % 10, pos[1] -pos[1] % 10)
        Personne(pos)
    elif state == 2: #SORTIES
        pos = ( pos[0]- pos[0] % 50, pos[1] - pos[1] % 50)
        # print('bipsortie')
        Sortie(pos)
    elif state ==3: #BROSSE
        temp_rect = pygame.rect.Rect(pos[0],pos[1],1,1)

        for personne in l_personnes:
            if temp_rect.colliderect(personne.rect):
                l_personnes.remove(personne)
        for sortie in l_sorties:
            if temp_rect.colliderect(sortie.rect):
                l_sorties.remove(sortie)
        for mur in l_murs:
            if temp_rect.colliderect(mur.rect):
                l_murs.remove(mur)
        for feu in l_feu_actif:
            if temp_rect.colliderect(feu.rect):
                l_feu_actif.remove(feu)
    elif state ==4: # FEU
        pos = ( pos[0] - pos[0] % 10,pos[1] - pos[1] % 10)
        Feu(pos)
if pygame.mouse.get_pressed()[0]:
    pos = pygame.mouse.get_pos()

    if state == 0: #MURS
        pos = ( pos[0] - pos[0] % 20,pos[1] - pos[1] % 20)
        Mur(pos)
    elif state == 1: #PERSONNES
        pos = ( pos[0] - pos[0] % 10, pos[1] -pos[1] % 10)
```

Version 1

```
### Imports et déclaration des variables
from termcolor import colored
import random
import os
import pyautogui
tabl = [ ["." for i in range (12)] for j in range(12)]

nbmorts = 0
nbpersonnes = 0
nbsurvivants = 0

### Fonctions
def affichage(tabl):
    res = ""
    for i in range(12):
        for j in range(12):
            res += str(tabl[i][j] )+ " "
        res += '\n'
    print (res)

def crea_sortie(tabl):
    for i in range(2):
        for j in range(2):
            tabl[i][j]=colored('S','green')


def crea_personnes(tabl,densite):
    global nbpersonnes
    for i in range(2,10):
        for j in range(2,10):
            nombre = random.randint(1,100)
            if nombre <=densite:
                tabl[i][j] = 'P'
                nbpersonnes +=1

```

Version 1

```
35
36  def dep_personne(tabl):
37      global nbpersonnes
38      global nbsurvivants
39      for i in range(12):
40          for j in range(12):
41              if tabl[i][j] == 'P':
42                  if tabl[i][j-1] == colored('S','green') or tabl[i-1][j] == colored('S','green'):
43                      tabl[i][j] = '.'
44                      nbpersonnes = nbpersonnes -1
45                      nbsurvivants += 1
46
47
48                  elif i>0 and  tabl[i-1][j]== '.':
49                      tabl[i-1][j] = 'P'
50                      tabl[i][j] = '.'
51
52                  elif j>0 and tabl[i][j-1]== '.':
53                      tabl[i][j-1] = 'P'
54                      tabl[i][j] = '.'
55
56  def rependre_feu(tabl,taillepre):
57      global nbpersonnes
58      global nbmorts
59      for i in range(11-taillepre,12):
60          if tabl[11-taillepre][i]== 'P':
61              nbpersonnes= nbpersonnes -1
62              nbmorts += 1
63          tabl[11-taillepre][i]=colored('F','red')
64      for j in range(11-taillepre,12):
65          if tabl[j][11-taillepre]== 'P':
66              nbpersonnes= nbpersonnes -1
67              nbmorts += 1
68          tabl[j][11-taillepre]=colored('F','red')
69
70
71  def tour(i):
72      rependre_feu(tabl,i)
73      dep_personne(tabl,nbpersonnes)
74      affichage(tabl)
75      print("        Étape " +str(i))
76
```

Version 1

```
76
77  ### MAIN
78  def main():
79      crea_sortie(tabl)
80      crea_personnes(tabl,20)
81      i=-1
82      while nbpersonnes !=0:
83          i+=1
84          rependre_feu(tabl,i)
85          dep_personne(tabl)
86          affichage(tabl)
87          print("          Étape " + str(i))
88          pyautogui.sleep(2)
89          # affichage(tabl)
90          # print("        Étape " +str(2*i))
91          # pyautogui.sleep(2)
92          #
93          # dep_personne(tabl)
94          #
95          # affichage(tabl)
96          # print("        Étape " +str(2*i+1))
97          #
98          #
99          # pyautogui.sleep(2)
100
101
102 main()
103
104 ### AFFICHAGE RÉSULTATS
105
106 print("nb_morts=", nbmorts)
107 print("nb_survivants=", nbsurvivants)
108
109
```

Version 2

```
### Imports et déclaration des variables
import pygame
from pygame.locals import *

import pyautogui

SIZE = 700, 800

nbmorts = 0
nbpersonnes = 0
nbsurvivants = 0


pygame.init()
screen = pygame.display.set_mode(SIZE)


screen.fill("white")


tabl =[
['S', 'S', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.'],
['S', 'S', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.'],
['.', '.', '.', '.', '.', '.', 'P', '.', '.', '.', '.', '.'],
['.', '.', '.', '.', '.', '.', '.', 'P', '.', '.', '.', '.'],
['.', '.', 'P', '.', '.', '.', '.', 'P', 'P', '.', '.', '.'],
['.', '.', '.', '.', '.', '.', 'P', '.', '.', '.', '.', '.'],
['.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.'],
['.', '.', '.', '.', '.', '.', '.', 'P', 'P', '.', '.', '.'],
['.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.'],
['.', '.', '.', 'P', 'P', '.', 'P', '.', '.', '.', '.', '.'],
['S', 'S', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.'],
['S', 'S', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.']
]

### Fonctions

def rependre_feu(tabl,taillepre):
    global nbpersonnes
    global nbmorts
    for i in range(11-taillepre,12):
        if tabl[11-taillepre][i]== 'P':
            nbpersonnes= nbpersonnes -1
            nbmorts += 1
        tabl[11-taillepre][i]='F'
    for j in range(11-taillepre,12):
        if tabl[j][11-taillepre]== 'P':
            nbpersonnes= nbpersonnes -1
            nbmorts += 1
        tabl[j][11-taillepre]='F'

```

Version 2

```
def dep_personne1(tabl):
    global nbpersonnes
    global nbsurvivants
    for i in range(12):
        for j in range(12):
            if tabl[i][j] == 'P':
                if tabl[i][j-1] == "S" or tabl[i-1][j] == 'S':
                    tabl[i][j] = '.'
                    nbpersonnes = nbpersonnes -1
                    nbsurvivants += 1

                elif j>0 and tabl[i][j-1]== '.':
                    tabl[i][j-1] = 'P'
                    tabl[i][j] = '.'

                elif i>0 and  tabl[i-1][j]== '.':
                    tabl[i-1][j] = 'P'
                    tabl[i][j] = '.'


def dep_personne2(tabl):
    for i in range(12):
        for j in range(12):
            if tabl[i][j] == 'P':
                if (j>0 and tabl[i][j-1] ==  'S')  or tabl[i-1][j] == 'S' or ( j<12 and tabl[i][j+1] == 'S' ):
                    tabl[i][j] = '.'



                elif i>0 and  tabl[i-1][j]== '.':
                    tabl[i-1][j] = 'P'
                    tabl[i][j] = '.'

                elif j >6 and j<10:
                    if j>0 and tabl[i][j+1]== '.':
                        tabl[i][j+1] = 'P'
                        tabl[i][j] = '.'
                else:
                    if j>0 and tabl[i][j-1]== '.':
                        tabl[i][j-1] = 'P'
                        tabl[i][j] = '.'
```

Version 2

```
### DEBUT BOUCLE PRINCIPALE
font = pygame.font.Font('freesansbold.ttf', 32)
text = font.render('Etape', True, 'black')
textRect = text.get_rect()
textRect.center = (700 // 2, 750)

etape = 0
running = True
while running:
    for event in pygame.event.get():
        if event.type == QUIT:
            running = False
    text = font.render("Etape "+str(etape), True, 'black')
    ligne = Rect(0,725,800,5)
    pygame.draw.rect(screen, 'black', ligne)


    screen.blit(text, textRect)


    for i in range(12):
        for j in range(12):
            if tabl[i][j]== 'P':
                rect = Rect(j*60,i*60,50,50)
                pygame.draw.rect(screen, 'blue',rect)
            elif tabl[i][j] == 'F':
                rect = Rect(j*60,i*60,60,60)
                pygame.draw.rect(screen, 'red', rect)
            elif tabl[i][j] == 'S':
                rect = Rect(i*60,j*60,50,50)
                pygame.draw.rect(screen, 'green', rect)


    dep_personne2(tabl)
    rependre_feu(tabl,etape)
    pygame.display.flip()
    pyautogui.sleep(1)
    screen.fill( (255,255,255) )
    etape +=1
    if etape >12:
        running = False

```

```
141 ### AFFICHAGE RÉSULTATS
142
143 screen.fill( 'white')
144 texte1 = font.render("nb personnes : " + str(-nbpersonnes), True, 'black')
145 txtRect1 = texte1.get_rect()
146 txtRect1.center =(350,200)
147 texte2 = font.render("nb survivants : " + str(nbsurvivants), True, 'black')
148 txtRect2 = texte2.get_rect()
149 txtRect2.center= (350,400)
150 texte3 = font.render("nb morts : " + str(nbmorts), True, 'black')
151 txtRect3 = texte3.get_rect()
152 txtRect3.center= (350,600)
153
154
155 screen.blit(texte1, txtRect1)
156 screen.blit(texte2, txtRect2)
157 screen.blit(texte3, txtRect3)
158
159 pygame.display.flip()
160
161 input()
162
```

Version 2

Version 3

```python
### Imports et déclaration des variables
import pygame
import os
import random
import pyautogui

pygame.init()
global survivants
survivants = 0
global morts
morts= 0

#creation écran
screen = pygame.display.set_mode((810,810))

resultats = []
print("Avec ou sans feu ? ")
B_feu = input() == 'avec'


### Classes
class Feu(object):

    def __init__(self):
        self.rect = pygame.Rect(800,800,10,10)

    def etendre(self):
        global morts
        self.rect.x += -10
        self.rect.y += -10
        self.rect.width +=10
        self.rect.height +=10


        #on enlève les personnes qui touchent le feu et on incrémentent les morts
        for personne in individus:
            if self.rect.colliderect(personne.rect):
                for personne2 in range(personne.numero+1,len(individus)):
                    individus[personne2].numero -=1
                personne.kill(individus)
                morts += 1
```

Version 3

```
43
44
45  class Individus(object):
46      est_present = True
47      numero = 0
48
49      couleur1 = 0
50      couleur2 = 0
51      couleur3 = 0
52
53      objectif_y = '.'
54      objectif_x = '.'
55
56      def __init__(self,pos,i):
57          individus.append(self)
58          self.numero = i
59          self.couleur1 = (self.numero * 39) % 255
60          self.couleur2 = (self.numero * 73) % 255
61          self.couleur3 = (self.numero * 43) % 255
62
63          self.rect = pygame.Rect(pos[0],pos[1], 15,15)
64
65      def move(self,dx,dy):
66          self.move_x(dx)
67          self.move_y(dy)
68          #print(self.est_present)
69
70      def move_x(self,dx):
71          global survivants
72          if dx >0:
73              sens = 1
74          else:
75              sens = -1
76          for i in range(abs(dx)):
77              self.rect.x += 1 *sens
78
79
80
81
82              #detection colisions avec les murs et arret
83
84              for mur in murs:
85                  if self.rect.colliderect(mur.rect):
86                      self.rect.x += -1 * sens
87                      break
88              #detection colisions avec les murs et arret
89              for personne in individus:
90                  if personne != self and self.rect.colliderect(personne.rect):
91                      self.rect.x += -1* sens
92                      break
```

Version 3

```
93
94      def move_y(self,dx):
95          if dx >0:
96              sens = 1
97          else:
98              sens = -1
99
100         flag = True
101         for i in range(abs(dx)):
102             self.rect.y += 1 * sens
103
104
105
106             # détection des sorties, incrémentation et disparaition de la liste des individis
107             for sortie in sorties:
108                 if self.rect.colliderect(sortie.rect):
109                     self.est_present=False
110                      #print("sortie")
111                      #print(individus)
112                     if flag:
113                         for personne in range(self.numero+1,len(individus)):
114                             individus[personne].numero -=1
115                         self.kill(individus)
116
117
118                         flag = False
119
120
121             for mur in murs:
122                 if self.rect.colliderect(mur.rect):
123                     self.rect.y += -1 * sens
124                     break
125
126             for personne in individus:
127                 if personne != self and self.rect.colliderect(personne.rect):
128                     self.rect.y +=  -1 *sens
129                     break
130
131     def kill(self,individus):
132         #on enlève de la liste des individus l'objet actuel pour ne plus le traiter
133         individus.pop(self.numero)
134
```

```
147
148
149
150
151 ### DEBUT BOUCLE TOURS
152 for tour in range(1,3):
153
154     #lecture du fichier et transformation en listes d'individus, de murs et de sorties
155     os.chdir(r"C:\Users\girem\Desktop\lycée et prépa\Prépa\TIPE\CARTES DE PERSONNES")
156     fichier = open(str(tour) + '.txt','r')
157     tabl = [""] * 56
158     for i in range(56):
159         contenu = fichier.readline()
160
161         tabl[i] = contenu.strip()
162     map = tabl
163
164
165     individus = []
166     murs = []
167     sorties = []
168     survivants = 0
169     morts= 0
170
171
172     x = y = 0
173     count = 0
174     for row in map:
175         for col in row:
176             if col == "W":
177                 Murs((x, y))
178             if col == 'S':
179                 Sortie((x,y))
180             if col == "P":
181                 Individus((x,y),count)
182                 count += 1
183             x += 15
184         y += 15
185         x = 0
186
187
```

```
188
189 ### APPUYEZ SUR ENTREE POUR TEMPORISER
190     font = pygame.font.Font('freesansbold.ttf', 32)
191     screen.fill("white")
192
193
194     #affichage
195
196     texte1 = font.render("Appuyez sur Entrée", True, 'black')
197     txtRect1 = texte1.get_rect()
198     txtRect1.center =(400,400)
199
200     texte2 = font.render("Tour " + str(tour),True, "red")
201     txtRect2 = texte2.get_rect()
202     txtRect2.center =(400,500)
203
204     screen.blit(texte1, txtRect1)
205     screen.blit(texte2, txtRect2)
206
207     pygame.display.flip()
208
209     #attente appuis entrée
210
211     flag = True
212     while flag:
213         for e in pygame.event.get():
214             if e.type == pygame.KEYDOWN and e.key == pygame.K_RETURN:
215                 flag = False
216
217
```

```
### FIN TEMPORISATION
### DEBUT BOUCLE PRINCIPALE

    running = True
    etape =0
    os.chdir(r"C:\Users\girem\Desktop\lycée et prépa\Prépa\TIPE\par tableau python\Simulation "+ str(5 + tour) + " images")

    #afficher ou non le feu
    if B_feu:
        feu = Feu()

    while running :
        #permet de quitter si besoin
        for e in pygame.event.get():
            if e.type == pygame.QUIT:
                running = False
            if e.type == pygame.KEYDOWN and e.key == pygame.K_ESCAPE:
                running = False

        #déplacements des personnes
        for personne in individus:

            if personne.est_present:
                dx = dy =-5
                if oriente == 1:
                    if personne.objectif_y == 'bas':
                        dy = 5
                    if personne.objectif_x == 'droite':
                        dx = 5
                personne.move(dx,dy)
        #print(individus)

        #répendre ou non le feu
        if B_feu:
                feu.etendre()









        screen.fill((0, 0, 0))
        #affichage à l'écran du feu (éventuellement) des sorties et des murs
        if B_feu:
            pygame.draw.rect(screen,"red", feu.rect)
        for sortie in sorties:
            pygame.draw.rect(screen, "green", sortie.rect)
        for mur in murs:
            pygame.draw.rect(screen, (255, 255, 255), mur.rect)
```

Version 3

```
269
270         for personne in individus:
271     #affichage des personnes si ils sont dans la liste d'individus
272
273             if personne.est_present:
274                 # pygame.draw.rect(screen, (150,75,0), personne.rect )
275                 pygame.draw.rect(screen, (personne.couleur1,personne.couleur2,personne.couleur3), personne.rect )
276
277         pygame.display.flip()
278         pyautogui.sleep(0.05)
279         #pour les captures d'écrans
280         # pyautogui.screenshot("etape " + str(etape) +".png",( 400,50,800,800))
281         etape +=1
282         if individus == []:
283             #quitte la boucle si il n'y a plus d'indivdus restants
284             pyautogui.sleep(0.5)
285
286             running = False
287
288     resultats.append((survivants,morts))
289
290 ### AFFICHAGE RESULTATS
291
292 font = pygame.font.Font('freesansbold.ttf', 32)
293 screen.fill("white")
294
295 for i in range(len(resultats)):
296     #affichage de texte ,le nombre de survivants et de morts à chaque tour
297     texte1 = font.render("Tour "+ str(i+1)+ " :", True, 'black')
298     txtRect1 = texte1.get_rect()
299     txtRect1.midtop =(200 + 275 * i ,300)
300
301     texte2 = font.render("Rescapés : " + str(resultats[i][0]),True, "black")
302     txtRect2 = texte2.get_rect()
303     txtRect2.midtop =(200 + 275 * i ,400)
304
305     texte3 = font.render("Victimes : " + str(resultats[i][1]),True, "black")
306     txtRect3 = texte2.get_rect()
307     txtRect3.midtop =(200 + 275 * i ,500 )
308
309     screen.blit(texte1, txtRect1)
310     screen.blit(texte2, txtRect2)
311     screen.blit(texte3, txtRect3)
312
313 pygame.display.flip()
314
315
316 flag = True
317 while flag:
318     #reste affiché tant qu'on ne quitte pas
319     for e in pygame.event.get():
320         if e.type == pygame.KEYDOWN and e.key == pygame.K_RETURN:
321             flag = False
```

Version 4

```
### Imports et déclaration des variables
import pygame
import os
import random
import pyautogui

pygame.init()
global survivants
survivants = 0
global morts
morts= 0

#creation écran
screen = pygame.display.set_mode((810,810))

resultats = []
print("Avec ou sans feu ? ")
B_feu = input() == 'avec'


### Classes
class Feu(object):

    def __init__(self):
        self.rect = pygame.Rect(800,800,10,10)

    def etendre(self):
        global morts
        self.rect.x += -10
        self.rect.y += -10
        self.rect.width +=10
        self.rect.height +=10


        #on enlève les personnes qui touchent le feu et on incrémentent les morts
        for personne in individus:
            if self.rect.colliderect(personne.rect):
                for personne2 in range(personne.numero+1,len(individus)):
                    individus[personne2].numero -=1
                personne.kill(individus)
                morts += 1

```

Version 4

```
class Individus(object):
    est_present = True
    numero = 0

    couleur1 = 0
    couleur2 = 0
    couleur3 = 0

    objectif_y = '.'
    objectif_x = '.'

    def __init__(self,pos,i):
        individus.append(self)
        self.numero = i
        self.couleur1 = (self.numero * 39) % 255
        self.couleur2 = (self.numero * 73) % 255
        self.couleur3 = (self.numero * 43) % 255

        self.rect = pygame.Rect(pos[0],pos[1], 15,15)


    def move(self,dx,dy):
        self.move_x(dx)
        self.move_y(dy)
        #print(self.est_present)

    def move_x(self,dx):
        global survivants
        if dx >0:
            sens = 1
        else:
            sens = -1
        for i in range(abs(dx)):
            self.rect.x += 1 *sens



            #detection colisions avec les murs et arret

            for mur in murs:
                if self.rect.colliderect(mur.rect):
                    self.rect.x += -1 * sens
                    break
            #detection colisions avec les murs et arret
            for personne in individus:
                if personne != self and self.rect.colliderect(personne.rect):
                    self.rect.x += -1* sens
                    break
```

Version 4

```
94
95      def move_y(self,dx):
96          global survivants
97          if dx >0:
98              sens = 1
99          else:
100             sens = -1
101
102         flag = True
103         for i in range(abs(dx)):
104             self.rect.y += 1 * sens
105
106
107
108             # détection des sorties, incrémentation et disparaition de la liste des individis
109             for sortie in sorties:
110                 if self.rect.colliderect(sortie.rect):
111                     self.est_present=False
112                      #print("sortie")
113                      #print(individus)
114                     if flag:
115                         for personne in range(self.numero+1,len(individus)):
116                             individus[personne].numero -=1
117                         self.kill(individus)
118                         survivants += 1
119
120
121                         flag = False
122
123
124             for mur in murs:
125                 if self.rect.colliderect(mur.rect):
126                     self.rect.y += -1 * sens
127                     break
128
129             for personne in individus:
130                 if personne != self and self.rect.colliderect(personne.rect):
131                     self.rect.y +=  -1 *sens
132                     break
133
134     def kill(self,individus):
135         #on enlève de la liste des individus l'objet actuel pour ne plus le traiter
136         individus.pop(self.numero)
137
138
139 class Murs(object):
140
141     def __init__(self,pos):
142         murs.append(self)
143         self.rect = pygame.Rect(pos[0],pos[1], 15,15)
144
```

Version 4

```
class Sortie(object):

    def __init__(self,pos):
        sorties.append(self)
        self.rect = pygame.Rect(pos[0],pos[1], 15,15)









### DEBUT BOUCLE TOURS
for tour in range(1,3):

    #lecture du fichier et transformation en listes d'individus, de murs et de sorties
    os.chdir(r"C:\Users\girem\Desktop\lycée et prépa\Prépa\TIPE\CARTES DE PERSONNES")
    fichier = open(str(tour) + '.txt','r')
    tabl = [""] * 56
    for i in range(56):
        contenu = fichier.readline()

        tabl[i] = contenu.strip()
    map = tabl


    individus = []
    murs = []
    sorties = []
    survivants = 0
    morts= 0


    x = y = 0
    count = 0
    for row in map:
        for col in row:
            if col == "W":
                Murs((x, y))
            if col == 'S':
                Sortie((x,y))
            if col == "P":
                Individus((x,y),count)
                count += 1
            x += 15
        y += 15
        x = 0


    # print("orieté?  1 pour oui, 0 pour non")
    oriente = 0

```

```
196
197 ### APPUYEZ SUR ENTREE POUR TEMPORISER
198     font = pygame.font.Font('freesansbold.ttf', 32)
199     screen.fill("white")
200
201
202     #affichage
203
204     texte1 = font.render("Appuyez sur Entrée", True, 'black')
205     txtRect1 = texte1.get_rect()
206     txtRect1.center =(400,400)
207
208     texte2 = font.render("Tour " + str(tour),True, "red")
209     txtRect2 = texte2.get_rect()
210     txtRect2.center =(400,500)
211
212     screen.blit(texte1, txtRect1)
213     screen.blit(texte2, txtRect2)
214
215     pygame.display.flip()
216
217     #attente appuis entrée
218
219     flag = True
220     while flag:
221         for e in pygame.event.get():
222             if e.type == pygame.KEYDOWN and e.key == pygame.K_RETURN:
223                 flag = False
224
225
226
```

```
226
227 ### FIN TEMPORISATION
228 ### DEBUT BOUCLE PRINCIPALE
229
230     running = True
231     etape =0
232     os.chdir(r"C:\Users\girem\Desktop\lycée et prépa\Prépa\TIPE\par tableau python\Simulation "+ str(5 + tour) + " images")
233
234     #afficher ou non le feu
235     if B_feu:
236         feu = Feu()
237
238     while running :
239         #permet de quitter si besoin
240         for e in pygame.event.get():
241             if e.type == pygame.QUIT:
242                 running = False
243             if e.type == pygame.KEYDOWN and e.key == pygame.K_ESCAPE:
244                 running = False
245
246         #déplacements des personnes
247         for personne in individus:
248
249             if personne.est_present:
250                 nombrex = random.randint(-3,4)
251                 nombrey = random.randint(-3,4)
252                 dx = -5 + nombrex
253                 dy = -5 + nombrey
254                 if oriente == 1:
255                     if personne.objectif_y == 'bas':
256                         dy = 5
257                     if personne.objectif_x == 'droite':
258                         dx = 5
259                 personne.move(dx,dy)
260         #print(individus)
261
262         #répendre ou non le feu
263         if B_feu:
264                 feu.etendre()
265
266
267
268
269
270
271         screen.fill((0, 0, 0))
272         #affichage à l'écran du feu (éventuellement) des sorties et des murs
273         if B_feu:
274             pygame.draw.rect(screen,"red", feu.rect)
275         for sortie in sorties:
276             pygame.draw.rect(screen, "green", sortie.rect)
277         for mur in murs:
278             pygame.draw.rect(screen, (255, 255, 255), mur.rect)
279
```

Version 4

```
280         for personne in individus:
281             if personne.est_present:
282                 pygame.draw.rect(screen, (personne.couleur1,personne.couleur2,personne.couleur3), personne.rect)
283 
284         pygame.display.flip()
285         pyautogui.sleep(0.05)
286 
287         etape +=1
288         if individus == []:
289             #quitte la boucle si il n'y a plus d'indivdus restants
290             pyautogui.sleep(0.5)
291 
292             running = False
293 
294     resultats.append((survivants,morts,etape))
```

```
295
296 ### AFFICHAGE RESULTATS
297
298 font = pygame.font.Font('freesansbold.ttf', 32)
299 screen.fill("white")
300
301 for i in range(len(resultats)):
302     #affichage de texte ,le nombre de survivants et de morts à chaque tour
303     texte1 = font.render("Tour "+ str(i+1)+ " :", True, 'black')
304     txtRect1 = texte1.get_rect()
305     txtRect1.midtop =(200 + 275 * i ,300)
306
307     texte2 = font.render("Rescapés : " + str(resultats[i][0]),True, "black")
308     txtRect2 = texte2.get_rect()
309     txtRect2.midtop =(200 + 275 * i ,400)
310
311     texte3 = font.render("Victimes : " + str(resultats[i][1]),True, "black")
312     txtRect3 = texte2.get_rect()
313     txtRect3.midtop =(200 + 275 * i ,500 )
314
315     texte4 = font.render("Temps : " + str(resultats[i][2]),True, "black")
316     txtRect4 = texte2.get_rect()
317     txtRect4.midtop =(200 + 275 * i ,600 )
318
319     screen.blit(texte1, txtRect1)
320     screen.blit(texte2, txtRect2)
321     screen.blit(texte3, txtRect3)
322     screen.blit(texte4, txtRect4)
323
324 pygame.display.flip()
325
326
327 flag = True
328 while flag:
329     #reste affiché tant qu'on ne quitte pas
330     for e in pygame.event.get():
331         if e.type == pygame.KEYDOWN and e.key == pygame.K_RETURN:
332             flag = False
333
```

Version 5

```
### Imports et déclaration des variables
import pygame
import os
import random
import pyautogui

pygame.init()
global survivants
survivants = 0
global morts
morts= 0

#creation écran
screen = pygame.display.set_mode((810,810))

resultats = []
print("Avec ou sans feu ? ")
B_feu = input() == 'avec'


### Classes
class Feu(object):

    def __init__(self):
        self.rect = pygame.Rect(800,800,10,10)

    def etendre(self):
        global morts
        self.rect.x += -10
        self.rect.y += -10
        self.rect.width +=10
        self.rect.height +=10


        #on enlève les personnes qui touchent le feu et on incrémentent les morts
        for personne in individus:
            if self.rect.colliderect(personne.rect):
                for personne2 in range(personne.numero+1,len(individus)):
                    individus[personne2].numero -=1
                personne.kill(individus)
                morts += 1
```

Version 5

```
class Individus(object):
    est_present = True
    numero = 0

    couleur1 = 0
    couleur2 = 0
    couleur3 = 0
    couleur = "blue"


    objectif_y = '.'
    objectif_x = '.'

    def __init__(self,pos,i):
        individus.append(self)
        self.numero = i
        self.couleur1 = (self.numero * 39) % 255
        self.couleur2 = (self.numero * 73) % 255
        self.couleur3 = (self.numero * 43) % 255

        self.rect = pygame.Rect(pos[0],pos[1], 15,15)



    def f_distance(self, feu):
        x,y = self.rect.center

        dist_x = -x + feu.rect.x
        dist_y = -y + feu.rect.y
        if dist_x < 200 and dist_y < 200:
            # print(min(dist_x,dist_y))
            self.couleur = "yellow"
            return True, (210-max( dist_x, dist_y) )//5

        else:
            # print(self.couleur)
            if self.couleur == "yellow":
                # print(0)
                self.couleur = "brown"
            return False, 0


```

Version 5

```
87
88      def move(self,dx,dy):
89          self.move_x(dx)
90          self.move_y(dy)
91          #print(self.est_present)
92
93      def move_x(self,dx):
94          global survivants
95          if dx >0:
96              sens = 1
97          else:
98              sens = -1
99          for i in range(abs(dx)):
100             self.rect.x += 1 *sens
101
102
103
104
105             #detection colisions avec les murs et arret
106
107             for mur in murs:
108                 if self.rect.colliderect(mur.rect):
109                     self.rect.x += -1 * sens
110                     break
111             #detection colisions avec les murs et arret
112             for personne in individus:
113                 if personne != self and self.rect.colliderect(personne.rect):
114                     self.rect.x += -1* sens
115                     break
116
```

Version 5

```
116
117     def move_y(self,dx):
118         global survivants
119         if dx >0:
120             sens = 1
121         else:
122             sens = -1
123
124         flag = True
125         for i in range(abs(dx)):
126             self.rect.y += 1 * sens
127
128
129
130             # détection des sorties, incrémentation et disparaition de la liste des individis
131             for sortie in sorties:
132                 if self.rect.colliderect(sortie.rect):
133                     self.est_present=False
134                      #print("sortie")
135                      #print(individus)
136                     if flag:
137                         for personne in range(self.numero+1,len(individus)):
138                             individus[personne].numero -=1
139                         self.kill(individus)
140                         survivants += 1
141
142
143                         flag = False
144
145
146             for mur in murs:
147                 if self.rect.colliderect(mur.rect):
148                     self.rect.y += -1 * sens
149                     break
150
151             for personne in individus:
152                 if personne != self and self.rect.colliderect(personne.rect):
153                     self.rect.y +=  -1 *sens
154                     break
155
156     def kill(self,individus):
157         #on enlève de la liste des individus l'objet actuel pour ne plus le traiter
158         individus.pop(self.numero)
159
```

Version 5

```
class Murs(object):

    def __init__(self,pos):
        murs.append(self)
        self.rect = pygame.Rect(pos[0],pos[1], 15,15)

class Sortie(object):

    def __init__(self,pos):
        sorties.append(self)
        self.rect = pygame.Rect(pos[0],pos[1], 15,15)









### DEBUT BOUCLE TOURS
for tour in range(1,3):

    #lecture du fichier et transformation en listes d'individus, de murs et de sorties
    os.chdir(r"C:\Users\girem\Desktop\TIPE à montrer\Cartes 2")
    fichier = open(str(tour) + '.txt','r')
    tabl = [""] * 56
    for i in range(56):
        contenu = fichier.readline()

        tabl[i] = contenu.strip()
    map = tabl


    individus = []
    murs = []
    sorties = []
    survivants = 0
    morts= 0


    x = y = 0
    count = 0
    for row in map:
        for col in row:
            if col == "W":
                Murs((x, y))
            if col == 'S':
                Sortie((x,y))
            if col == "P":
                Individus((x,y),count)
                count += 1
            x += 15
        y += 15
        x = 0

```

Version 5

```
215
216     # print("orieté?  1 pour oui, 0 pour non")
217     oriente = 0
218
219 ### APPUYEZ SUR ENTREE POUR TEMPORISER
220     font = pygame.font.Font('freesansbold.ttf', 32)
221     screen.fill("white")
222
223
224     #affichage
225
226     texte1 = font.render("Appuyez sur Entrée", True, 'black')
227     txtRect1 = texte1.get_rect()
228     txtRect1.center =(400,400)
229
230     texte2 = font.render("Tour " + str(tour),True, "red")
231     txtRect2 = texte2.get_rect()
232     txtRect2.center =(400,500)
233
234     screen.blit(texte1, txtRect1)
235     screen.blit(texte2, txtRect2)
236
237     pygame.display.flip()
238
239     #attente appuis entrée
240
241     flag = True
242     while flag:
243         for e in pygame.event.get():
244             if e.type == pygame.KEYDOWN and e.key == pygame.K_RETURN:
245                 flag = False
246
247
248
249 ### FIN TEMPORISATION
```

Version 5

```
### DEBUT BOUCLE PRINCIPALE

    running = True
    etape =0
    # os.chdir(r"C:\Users\girem\Desktop\lycée et prépa\Prépa\TIPE\par tableau python\Simulation "+ str(5 + tour) + " images")

    #afficher ou non le feu
    if B_feu:
        feu = Feu()

    while running :
        #permet de quitter si besoin
        for e in pygame.event.get():
            if e.type == pygame.QUIT:
                running = False
            if e.type == pygame.KEYDOWN and e.key == pygame.K_ESCAPE:
                running = False

        #déplacements des personnes
        for personne in individus:

            if personne.est_present:
                moinsde100 = False
                rand_x = random.randint(-1,2)
                rand_y = random.randint(-1,2)
                if B_feu:
                    moinsde100,accel  = personne.f_distance(feu)


                dx = -5 + rand_x
                dy = -5 + rand_y

                if oriente == 1:
                    if personne.objectif_y == 'bas':
                        dy = 4
                    if personne.objectif_x == 'droite':
                        dx = 4
                if moinsde100:
                    personne.move(dx-accel,dy-accel)
                else:
                    personne.move(dx,dy)
        #print(individus)

        #répendre ou non le feu
        if B_feu:
                feu.etendre()
```

Version 5

```
297
298
299
300
301         screen.fill((0, 0, 0))
302         #affichage à l'écran du feu (éventuellement) des sorties et des murs
303         if B_feu:
304             pygame.draw.rect(screen,"red", feu.rect)
305         for sortie in sorties:
306             pygame.draw.rect(screen, "green", sortie.rect)
307         for mur in murs:
308             pygame.draw.rect(screen, (255, 255, 255), mur.rect)
309
310
311         for personne in individus:
312     #affichage des personnes si ils sont dans la liste d'individus
313
314             if personne.est_present:
315                 # pygame.draw.rect(screen, (150,75,0), personne.rect )
316                 #pygame.draw.rect(screen, (personne.couleur1,personne.couleur2,personne.couleur3), personne.rect )
317                 pygame.draw.rect(screen, personne.couleur, personne.rect)
318
319         pygame.display.flip()
320         pyautogui.sleep(0.05)
321         #pour les captures d'écrans
322         # pyautogui.screenshot("etape " + str(etape) +".png",( 400,50,800,800))
323         etape +=1
324         if individus == []:
325             #quitte la boucle si il n'y a plus d'indivdus restants
326             pyautogui.sleep(0.5)
327
328             running = False
329
330     resultats.append((survivants,morts,etape))
```

```
331
332 ### AFFICHAGE RESULTATS
333
334 font = pygame.font.Font('freesansbold.ttf', 32)
335 screen.fill("white")
336
337 for i in range(len(resultats)):
338     #affichage de texte ,le nombre de survivants et de morts à chaque tour
339     texte1 = font.render("Tour "+ str(i+1)+ " :", True, 'black')
340     txtRect1 = texte1.get_rect()
341     txtRect1.midtop =(200 + 275 * i ,300)
342
343     texte2 = font.render("Rescapés : " + str(resultats[i][0]),True, "black")
344     txtRect2 = texte2.get_rect()
345     txtRect2.midtop =(200 + 275 * i ,400)
346
347     texte3 = font.render("Victimes : " + str(resultats[i][1]),True, "black")
348     txtRect3 = texte2.get_rect()
349     txtRect3.midtop =(200 + 275 * i ,500 )
350
351     texte4 = font.render("Temps : " + str(resultats[i][2]),True, "black")
352     txtRect4 = texte2.get_rect()
353     txtRect4.midtop =(200 + 275 * i ,600 )
354
355     screen.blit(texte1, txtRect1)
356     screen.blit(texte2, txtRect2)
357     screen.blit(texte3, txtRect3)
358     screen.blit(texte4, txtRect4)
359
360 pygame.display.flip()
361
362
363 flag = True
364 while flag:
365     #reste affiché tant qu'on ne quitte pas
366     for e in pygame.event.get():
367         if e.type == pygame.KEYDOWN and e.key == pygame.K_RETURN:
368             flag = False
369
```

```python
import pygame
import os
import time

pygame.init()

screen = pygame.display.set_mode((1000,1000))

### Font
number_font = pygame.font.SysFont( None, 16 )

### Fonctions

def insertion_croissante_blocs(liste, elem_bloc):
    for i_bloc in range(len(liste)):
        if liste[i_bloc].distance_f > elem_bloc.distance_f:
            liste = liste[:i_bloc] + [elem_bloc] + liste[i_bloc:]
            return liste
    liste.append(elem_bloc)
    return liste

def insertion_croissante(liste, elem_bloc):
    for i_bloc in range(len(liste)):
        if liste[i_bloc] > elem_bloc:
            liste = liste[:i_bloc] + [elem_bloc] + liste[i_bloc:]

            return liste

def affichageblocs(liste):
    for bloc in liste:
        print(bloc.rect.centerx, bloc.rect.centery)
    print()

def chercher_trajet_retour(bloc, liste):
    if not( bloc.rect.colliderect(bloc_depart)):
        liste = [bloc] + chercher_trajet_retour(bloc.parent, liste)
    return liste

```

```
### Classes
class Debut:
    color  = "yellow"

    def __init__(self,pos):
        self.rect = pygame.rect.Rect(pos[0],pos[1],50,50)

class Fin:
    color = "red"

    def __init__(self,pos):
        self.rect = pygame.rect.Rect(pos[0],pos[1],50,50)

class Murs:
    color = "white"

    def __init__(self,pos):
        self.rect = pygame.rect.Rect(pos[0],pos[1],50,50)
        l_murs.append(self)

class Bloc_Trajet:
    distance_g = 0
    distance_h = 0

    distance_f = distance_g + distance_h

    def __init__(self,pos,parent, flag = True ) :
        self.rect = pygame.rect.Rect(pos[0],pos[1],50,50)
        if flag :
            self.parent = parent
            self.distance_g = parent.distance_g + 50
        else :
            self_distance = 0
        # self.distance_g = abs( pos[0] - debut.rect.x) + abs( pos[1] - debut.rect.y )
        self.distance_h = abs( pos[0] - fin.rect.x) + abs( pos[1] - fin.rect.y )
        self.distance_f = self.distance_h + self.distance_g



    def affichage(self,couleur):
        pygame.draw.rect(screen, couleur, self.rect)
        image_x = number_font.render( str(self.rect.centerx), True, (0,0,0), couleur)
        image_y = number_font.render( str(self.rect.centery), True, (0,0,0), couleur)
        screen.blit(image_x, (self.rect.centerx  - 20,self.rect.centery))
        screen.blit(image_y, (self.rect.centerx  + 7,self.rect.centery))

```

```
85      def etendre(self):
86          global l_ouverte
87          global l_fermee
88          global running
89          global trajet
90
91
92          bloc_h = Bloc_Trajet((  self.rect.left, self.rect.top - self.rect.height), self)
93
94          bloc_d = Bloc_Trajet((  self.rect.left + self.rect.width, self.rect.top ), self)
95
96          bloc_b = Bloc_Trajet((  self.rect.left, self.rect.top + self.rect.height ), self)
97
98          bloc_g = Bloc_Trajet(( self.rect.left - self.rect.width, self.rect.top  ), self)
99
100
101         liste1 = [bloc_g, bloc_h,bloc_d,bloc_b]
102         affichageblocs(liste1)
103         #on enlève tous les blocs qui sont déjà dans la liste fermée
104         for bloc in liste1:
105             if bloc.rect.colliderect(bloc_fin.rect):
106                 print("youpi")
107                 trajet = chercher_trajet_retour(bloc, [])
108                 running = False
109             flag_liste_fermee = False
110             for bloc_ferme in l_fermee:
111                 flag_liste_fermee = flag_liste_fermee or bloc.rect.colliderect(bloc_ferme.rect)
112
113             for mur in l_murs:
114                 flag_liste_fermee = flag_liste_fermee or bloc.rect.colliderect(mur.rect)
115
116             if not(flag_liste_fermee):
117                 flag_liste_ouverte = False
118                 for bloc_ouvert in l_ouverte:
119                     flag_liste_ouverte = flag_liste_ouverte or bloc.rect.colliderect(bloc_ouvert)
120
121                     if bloc.rect.colliderect(bloc_ouvert):
122                         if bloc.distance_g <  bloc_ouvert.distance_g :
123
124                             l_ouverte.remove(bloc_ouvert)
125                             l_ouverte = insertion_croissante_blocs(l_ouverte, bloc)
126
127                 if not(flag_liste_ouverte):
128                     l_ouverte = insertion_croissante_blocs(l_ouverte, bloc)
129
130
```

```
130
131 ### Boucle Principale
132
133 global running
134 running = True
135
136 global l_ouverte
137 global l_fermee
138 l_ouverte = []
139 l_fermee = []
140 l_murs= []
141 trajet = []
142
143
144 os.chdir(r"C:\Users\girem\Desktop\TIPE à montrer\pathfinding")
145
146 fichier = open(r"C:\Users\girem\Desktop\TIPE à montrer\pathfinding\Carte 2.txt",'r')
147 tabl = [""] * 20
148 for i in range(20):
149     contenu = fichier.readline()
150
151     tabl[i] = contenu.strip()
152 map = tabl
153
154
155 fin = Fin((1000,1000))
156 x = y = 0
157 count = 0
158 for row in map:
159     for col in row:
160         if col == "M":
161             Murs((x, y))
162         if col == 'D':
163             debut = Debut((x,y))
164             bloc_depart = Bloc_Trajet((x,y) ,debut, False)
165         if col == "F":
166             fin = Fin((x,y))
167             bloc_fin = Bloc_Trajet((x,y), fin, False)
168         x += 50
169     y += 50
170     x = 0
171
172
```

```
172
173 l_ouverte.append(bloc_depart)
174
175 running2 = True
176 temps1 = time.time()
177 while running:
178
179     for e in pygame.event.get():
180         if e.type == pygame.QUIT:
181             running = False
182             running2 = False
183         if e.type == pygame.KEYDOWN and e.key == pygame.K_ESCAPE:
184             running = False
185             running2 = False
186         ## partie traitement
187         if e.type == pygame.KEYDOWN and e.key == pygame.K_SPACE:
188             current_square = l_ouverte[0]
189
190             l_ouverte.pop(0)
191
192
193
194             l_fermee.append(current_square)
195             current_square.etendre()
196
197         if e.type == pygame.KEYDOWN and e.key == pygame.K_l:
198             while running:
199                 current_square = l_ouverte[0]
200
201                 l_ouverte.pop(0)
202
203
204
205                 l_fermee.append(current_square)
206                 current_square.etendre()
207
208
209     for bloc in l_ouverte:
210         bloc.affichage("gray")
211     for bloc in l_fermee:
212         bloc.affichage("green")
213     for mur in l_murs:
214         pygame.draw.rect(screen, mur.color, mur)
215
216     # bloc_depart.affichage()
217
218     pygame.draw.rect(screen, debut.color, debut.rect)
219     pygame.draw.rect(screen, fin.color, fin.rect)
220
221
222
223
224
```

```
222
223
224
225
226
227
228         pygame.display.flip()
229
230     temps2= time.time()
231     print(temps2-temps1)
232     a_ete_print = False
233     while running2:
234         for e in pygame.event.get():
235             if e.type == pygame.QUIT:
236                 running = False
237                 running2 = False
238             if e.type == pygame.KEYDOWN and e.key == pygame.K_ESCAPE:
239                 running = False
240                 running2 = False
241
242         for bloc in l_ouverte:
243             bloc.affichage("gray")
244         for bloc in l_fermee:
245             bloc.affichage("green")
246         for mur in l_murs:
247             pygame.draw.rect(screen, mur.color, mur)
248
249         # bloc_depart.affichage()
250
251         pygame.draw.rect(screen, debut.color, debut.rect)
252         pygame.draw.rect(screen, fin.color, fin.rect)
253         if not(a_ete_print):
254             print("liste des blocs de trajets :")
255         for bloc in trajet[1:]:
256             bloc.affichage("blue")
257             if not(a_ete_print):
258                 print(bloc.rect.x,bloc.rect.y,sep = "  ")
259         a_ete_print = True
260
261         pygame.display.flip()
```

Version 6

```
##Init

import pygame
pygame.init()
screen = pygame.display.set_mode((800,800))
l_murs = []
l_personnes =[]
l_sorties = []
l_bool = [True,True]
state = 0

l_feu_entoures = []
l_feu_actif    = []

##Classes
class Feu:

    def __init__(self,pos):
        self.rect=pygame.rect.Rect(pos[0],pos[1],10,10)
        l_feu_actif.append(self)
        for personne in l_personnes:
            if self.rect.colliderect(personne.rect):
                l_personnes.remove(personne)
        for sortie in l_sorties:
            if self.rect.colliderect(sortie.rect):
                l_sorties.remove(sortie)

        self.color = 'red'

    def etendre(self):

        rect_g = pygame.rect.Rect(self.rect.x - 10 , self.rect.y,10,10)
        rect_d = pygame.rect.Rect(self.rect.x + 10 , self.rect.y,10,10)
        rect_h = pygame.rect.Rect(self.rect.x, self.rect.y - 10,10,10)
        rect_b = pygame.rect.Rect(self.rect.x, self.rect.y + 10,10,10)
        list = [rect_g ,rect_d ,rect_h ,rect_b ]
        plein = True
        for rect in list:
            if rect.x <=0 or rect.y <=0:
                pass
            if rect.collidelist(l_feu_entoures+l_feu_actif) == -1 and rect.collidelist(l_murs) == -1:
                plein = False
                Feu((rect.x,rect.y))

        if plein:
            l_feu_actif.remove(self)
            l_feu_entoures.append(self)
```

Version 6

```
52
53  class Personne:
54
55      def __init__(self,pos):
56          self.rect= pygame.rect.Rect(pos[0],pos[1],10,10)
57          if self.rect.collidelist(l_personnes) == -1 and self.rect.collidelist(l_murs) == -1 and self.rect.collidelist(l_sorties) == -1:
58              l_personnes.append(self)
59
60          taille = len(l_personnes)
61          r,g,b = taille*73 % 256, taille* 26 %256,taille*179 % 256
62
63
64          self.color = (r,g,b)
65          self.trajet = []
66
67  class Mur:
68
69
70      def __init__(self,pos):
71          self.rect= pygame.rect.Rect(pos[0],pos[1],20,20)
72          if self.rect.collidelist(l_personnes) == -1 and self.rect.collidelist(l_murs) == -1 and self.rect.collidelist(l_sorties) == -1:
73              l_murs.append(self)
74
75          self.color = "white"
76  class Sortie:
77
78      def __init__(self,pos):
79          self.rect= pygame.rect.Rect(pos[0],pos[1],50,50)
80          self.color = "green"
81          if self.rect.collidelist(l_personnes) == -1 and self.rect.collidelist(l_murs) == -1 and self.rect.collidelist(l_sorties) == -1:
82              # print(0)
83              l_sorties.append(self)
84      def majpos(self,npos):
85          temp = pygame.rect.Rect(npos[0],pos[1],50,50)
86          if temp.collidelist(l_personnes) == -1 and temp.collidelist(l_murs) == -1:
87              self.rect.x,self.rect.y = npos
88
89  class Bloc_Trajet:
90      color = "brown"
91      distance_g = 0
92      distance_h = 0
93      distance_f = distance_g + distance_h
94
95      def __init__(self,pos,parent,taille, sortie, flag = True):
96          self.taille = taille
97          self.sortie = sortie
98          self.rect = pygame.rect.Rect(pos[0],pos[1],taille,taille)
99          if flag :
100             self.parent = parent
101             self.distance_g = parent.distance_g + taille
102         else :
103             self_distance = 0
104         self.distance_h = abs( pos[0] - sortie.rect.x) + abs( pos[1] - sortie.rect.y )
105         self.distance_f = self.distance_h + self.distance_g
```

Version 6

```
106
107     def etendre(self,l_ouverte,l_fermee,personne,sortie):
108         # la fin est sortie
109
110         def chercher_trajet_retour(bloc, liste):
111             if not( bloc.rect.colliderect(personne)):
112                 liste = [bloc] + chercher_trajet_retour(bloc.parent, liste)
113             return liste
114
115         bloc_h = Bloc_Trajet((  self.rect.left, self.rect.top - self.rect.height), self, self.taille,self.sortie)
116         bloc_d = Bloc_Trajet((  self.rect.left + self.rect.width, self.rect.top ), self, self.taille,self.sortie)
117         bloc_b = Bloc_Trajet((  self.rect.left, self.rect.top + self.rect.height ), self, self.taille,self.sortie)
118         bloc_g = Bloc_Trajet(( self.rect.left - self.rect.width, self.rect.top  ), self, self.taille,self.sortie)
119
120         liste1 = [bloc_g, bloc_h,bloc_d,bloc_b]
121         for bloc in liste1:
122             if bloc.rect.colliderect(sortie.rect):
123                 # print("youpi")
124                 trajet = chercher_trajet_retour(bloc, [])
125                 print(len(trajet))
126                 return (True,trajet)
127             flag_liste_fermee = False
128             for bloc_ferme in l_fermee:
129                 flag_liste_fermee = flag_liste_fermee or bloc.rect.colliderect(bloc_ferme.rect)
130
131             for mur in l_murs+l_personnes+l_feu_actif:
132                 flag_liste_fermee = flag_liste_fermee or bloc.rect.colliderect(mur.rect)
133
134
135             # for personne in l_personnes:
136                 # flag_liste_fermee = flag_liste_fermee or bloc.rect.colliderect(personne.rect)
137
138             if not(flag_liste_fermee):
139                 flag_liste_ouverte = False
140                 for bloc_ouvert in l_ouverte:
141                     flag_liste_ouverte = flag_liste_ouverte or bloc.rect.colliderect(bloc_ouvert)
142
143                     if bloc.rect.colliderect(bloc_ouvert):
144                         if bloc.distance_g <  bloc_ouvert.distance_g :
145
146                             l_ouverte.remove(bloc_ouvert)
147                             l_ouverte = insertion_croissante_blocs(l_ouverte, bloc)
148
149                 if not(flag_liste_ouverte): #TODO Attention ajouter comme il faut
150                     l_ouverte = insertion_croissante_blocs(l_ouverte, bloc)
151
152         return (False,l_ouverte,l_fermee)
153
154
```

Version 6

```
154
155 ## Fonctions
156
157
158 def insertion_croissante_blocs(liste, elem_bloc):
159     for i_bloc in range(len(liste)):
160         if liste[i_bloc].distance_f > elem_bloc.distance_f:
161             liste = liste[:i_bloc] + [elem_bloc] + liste[i_bloc:]
162             return liste
163     liste.append(elem_bloc)
164     return liste
165
166
167 def affichage(*kargs):
168     # screen.fill(0)
169     for list in kargs:
170         for bloc in list:
171             pygame.draw.rect(screen, bloc.color,bloc.rect)
172
```

Version 6

```
def plus_court_chemin(personne):
    l_trajet =[]
    min = 3000
    trajet_f =-1
    for temp_sortie in l_sorties:
        dist = abs( pos[0] - temp_sortie.rect.centerx)//10 + abs( pos[1] - temp_sortie.rect.centery )//10
        print('calcul : ', dist)
        if dist > min:
            print('pass')
        else:
            l_ouverte = []
            l_fermee  = []
            trajet    = []
            bool = True
            Bloc = Bloc_Trajet(personne.rect.topleft,personne,10,temp_sortie,False)
            l_ouverte.append(Bloc)
            # print(0)
            while bool:

                res = l_ouverte[0].etendre(l_ouverte,l_fermee,personne,temp_sortie)
                if res[0]:
                    trajet = res[1]
                    if len(trajet)<min:
                        min = len(trajet)
                        trajet_f = trajet
                    break
                else:
                    l_ouverte = res[1]
                    l_ouverte.pop(0)
                    l_fermee  = res[2]



    for trajet in l_trajet:
        if len(trajet) < min:
            min = len(trajet)
            trajet_f = trajet

    return trajet_f


list = []

count = 0
```

Version 6

```
## boucle 1
while l_bool[0]:
    screen.fill(0)

    for e in pygame.event.get():
        if e.type == pygame.KEYDOWN and e.key == pygame.K_ESCAPE:
            print('bip')
            for i in range(len(l_bool)):
                l_bool[i] = False
        if e.type == pygame.KEYDOWN and e.key == pygame.K_m:
            state = 0
        elif e.type == pygame.KEYDOWN and e.key == pygame.K_p:
            state = 1
        elif e.type == pygame.KEYDOWN and e.key == pygame.K_s:
            state = 2
        elif e.type == pygame.KEYDOWN and e.key == pygame.K_c:
            state = 3
        elif e.type == pygame.KEYDOWN and e.key == pygame.K_f:
            state = 4
        if e.type == pygame.KEYDOWN and e.key == pygame.K_RETURN:
            l_bool[0] = False
        if e.type == pygame.KEYDOWN and e.key == pygame.K_w:
            print(len(l_feu_actif))
        if e.type == pygame.KEYDOWN and e.key == pygame.K_SPACE:
            # print('start feu')
            # count_feu =0
            for personne in l_personnes:
                # print(personne.trajet)


                #TODO  test si le trajet est en feu
                bool = False
                for bloc in personne.trajet:
                    if bloc.rect.collidelist(l_feu_actif) != -1:
                        bool = True
                        break
                if bool:
                    personne.trajet = plus_court_chemin(personne)

                affichage(personne.trajet)


```

Version 6

```
262
263             #TODO fin preced todo
264             temp_rect = personne.trajet[-1].rect
265             if temp_rect.collidelist(l_personnes) == -1:
266                 personne.rect.center = personne.trajet[-1].rect.center
267                 personne.trajet.pop(-1)
268                 if personne.rect.collidelist(l_sorties) != -1:
269                     l_personnes.remove(personne)
270                 if personne.rect.collidelist(l_feu_actif) != -1:
271                     l_personnes.remove(personne)
272         temp = l_feu_actif.copy()
273         for feu in temp:
274             # print('feu n°:',count_feu)
275             # count_feu +=1
276             feu.etendre()
277         # print('start pers')
278
279     if e.type == pygame.KEYDOWN and e.key == pygame.K_r:
280         for personne in l_personnes:
281             print('go')
282             personne.trajet = plus_court_chemin(personne)
283         print("READY !")
284
285
286
```

Version 6

```
285
286
287     if pygame.mouse.get_pressed()[0]:
288         pos = pygame.mouse.get_pos()
289
290         if state == 0: #MURS
291             pos = ( pos[0] - pos[0] % 20,pos[1] - pos[1] % 20)
292             Mur(pos)
293         elif state == 1: #PERSONNES
294             pos = ( pos[0] - pos[0] % 10, pos[1] -pos[1] % 10)
295             Personne(pos)
296         elif state == 2: #SORTIES
297             pos = ( pos[0]- pos[0] % 50, pos[1] - pos[1] % 50)
298             # print('bipsortie')
299             Sortie(pos)
300         elif state ==3: #BROSSE
301             temp_rect = pygame.rect.Rect(pos[0],pos[1],1,1)
302
303             for personne in l_personnes:
304                 if temp_rect.colliderect(personne.rect):
305                     l_personnes.remove(personne)
306             for sortie in l_sorties:
307                 if temp_rect.colliderect(sortie.rect):
308                     l_sorties.remove(sortie)
309             for mur in l_murs:
310                 if temp_rect.colliderect(mur.rect):
311                     l_murs.remove(mur)
312             for feu in l_feu_actif:
313                 if temp_rect.colliderect(feu.rect):
314                     l_feu_actif.remove(feu)
315         elif state ==4: # FEU
316             pos = ( pos[0] - pos[0] % 10,pos[1] - pos[1] % 10)
317             Feu(pos)
318
319
320
321     for rect in list:
322         # print('mais ? ')
323         pygame.draw.rect(screen, 'brown', rect)
324
325
326     affichage(l_murs,l_personnes,l_sorties,l_feu_actif) #l_feu_entoures
327     # if l_personnes != []:
328     #     pygame.draw.rect(screen,'red',l_personnes[count])
329     # pygame.draw.rect(screen,'green',sortie)
330     pygame.display.flip()
331
332
333
```