LE GOUVERNEMENT
DU GRAND-DUCHÉ DE LUXEMBOURG
Ministère de l'Éducation nationale,
de l'Enfance et de la Jeunesse

**EXAMEN DE FIN D'ÉTUDES SECONDAIRES CLASSIQUES**
**2019**
**CORRIGÉ – BARÈME**

| BRANCHE | SECTION(S) | ÉPREUVE ÉCRITE |
|---|---|---|
| **Informatique** | **B** | *Durée de l'épreuve :* 3h<br>*Date de l'épreuve :* |

```
#-------------------------------------------------------------------------------------------
import pygame, sys                                                                  # 1 p.
from pygame.locals import *
from random import randint, random

#-------------------------------------------------------------------------------------------
class Particle:
#-------------------------------------------------------------------------------------------
    def __init__(self, x, y, dx=0, dy=0, color=Color('white')):                    # 5 p.
        self.x, self.y = x, y
        self.dx, self.dy = dx, dy
        self.color = color
        self.trail = [(self.x, self.y)]
        self.dx += randint(-8, 8)
        self.dy += randint(-8,0)
        self.size = randint(10, 15)

#-------------------------------------------------------------------------------------------
    def draw(self):                                                                 # 6 p.
        x, y = self.x, self.y
        size = self.size
        color = self.color
        if color != Color('white'):
            pygame.draw.circle(screen, color, (round(x), round(y)), round(size))
            for i in range(len(self.trail) - 1):
                pygame.draw.line(screen, color, self.trail[i], self.trail[i+1])
        else:
            probability = random()
            if probability < 0.5:
                x += randint(-2, 2)
                y += randint(-2, 2)
                pygame.draw.line(screen, color, (x - size // 2, y - size // 2),
                                                (x + size // 2, y + size // 2))
                pygame.draw.line(screen, color, (x + size // 2, y - size // 2),
                                                (x - size // 2, y + size // 2))

#-------------------------------------------------------------------------------------------
    def move(self):                                                                 # 7 p.
        new_x, new_y = self.x + self.dx, self.y + self.dy
        if new_x < 0 or new_x >= WIDTH:
            self.dx = -self.dx
        if new_y < 0 or new_y >= HEIGHT:
            self.dy = -self.dy
        self.x += self.dx
        self.y += self.dy
        self.size -= 0.2
        self.dy += 0.4
        self.trail.append((self.x, self.y))
        while len(self.trail) > 20:
            self.trail.pop(0)

#-------------------------------------------------------------------------------------------
```

```
#-------------------------------------------------------------------------------
class Firework:
#-------------------------------------------------------------------------------
    def __init__(self):                                                  # 2 p.
        self.list = []

#-------------------------------------------------------------------------------
    def add_cracker(self, x, y, dx, dy, kind):                           # 4 p.
        if kind == 1:
            c = randint(1, 3)
            if c == 1:
                color = Color('red')
            elif c == 2:
                color = Color('green')
            else:
                color = Color('blue')
            for i in range(60):
                self.list.append(Particle(x, y, dx, dy, color))
        else:
            for i in range(120):
                self.list.append(Particle(x, y, dx, dy))

#-------------------------------------------------------------------------------
    def draw(self):                                                      # 2 p.
        for p in self.list:
            p.draw()

#-------------------------------------------------------------------------------
    def move(self):                                                      # 2 p.
        for p in self.list:
            p.move()

#-------------------------------------------------------------------------------
    def boost(self):                                                     # 2 p.
        for p in self.list:
            if p.color != Color('white'):
                p.dy -= 10

#-------------------------------------------------------------------------------
    def purge(self):                                                     # 5 p.
        for p in self.list:
            if p.color != Color('white') and p.size < 2:
                probability = random()
                if probability < 0.01:
                    self.add_cracker(p.x, p.y, 0, 0, 1)
        self.list = [part for part in self.list if part.size >= 2]

#-------------------------------------------------------------------------------
    def clear(self):                                                     # 2 p.
        self.list = []

#-------------------------------------------------------------------------------
```

```
#-------------------------------------------------------------------------------------------
pygame.init()                                                                    # 3 p.
WIDTH, HEIGHT = 1000, 700
FPS = 25
clock = pygame.time.Clock()
screen = pygame.display.set_mode((WIDTH, HEIGHT))
screen.fill(Color('black'))
fireworks = Firework()
done = False
x, y = 0, 0
dx, dy = 0, 0
auto = False
frame_count = 0
#-------------------------------------------------------------------------------------------
while not done:                                                                  # 1 p.
    for event in pygame.event.get():
        if event.type == QUIT:
            done = True
#-------------------------------------------------------------------------------------------
        elif event.type == MOUSEBUTTONDOWN:                                      # 3 p.
            button_pressed = pygame.mouse.get_pressed()
            x, y = pygame.mouse.get_pos()
            dx, dy = 0, 0
#-------------------------------------------------------------------------------------------
        elif event.type == MOUSEMOTION:                                          # 2 p.
            new_x, new_y = pygame.mouse.get_pos()
            dx, dy = (new_x - x) / 10, (new_y - y) / 10
#-------------------------------------------------------------------------------------------
        elif event.type == MOUSEBUTTONUP:                                        # 4 p.
            if button_pressed == (True, False, False):
                fireworks.add_cracker(new_x, new_y, dx, dy, 1)
            elif button_pressed == (False, False, True):
                fireworks.add_cracker(new_x, new_y, dx, dy, 2)
#-------------------------------------------------------------------------------------------
        if event.type == KEYDOWN:                                                #3 p.
            if event.key == K_a:
                auto = not (auto)
            elif event.key == K_b:
                fireworks.boost()
            elif event.key == K_s:
                fireworks.clear()
            elif event.key == K_ESCAPE:
                done = True
#-------------------------------------------------------------------------------------------
    frame_count = (frame_count + 1) % FPS                                        # 3 p.
    if auto and frame_count == 0:
        x_auto, y_auto = randint(0,WIDTH), randint(0, HEIGHT // 4)
        kind_auto = randint(1, 2)
        fireworks.add_cracker(x_auto, y_auto, 0, 0, kind_auto)
#-------------------------------------------------------------------------------------------
    fireworks.move()                                                             # 2 p.
    fireworks.purge()
    screen.fill(Color('black'))
    fireworks.draw()
    pygame.display.update()
    clock.tick(FPS)
#-------------------------------------------------------------------------------------------
pygame.quit()                                                                    # 1 p.
sys.exit()
```