TP9: MODELOS ESTOCÁSTICOS DE CRECIMIENTO POR AGREGACIÓN EN CONFINAMIENTO

Estos modelos se utilizan para estudiar el depósito de partículas en conductos de diferentes formas y secciones. Se trabaja en el plano, simulando movimiento aleatorio de partículas que se adhieren a las paredes del conducto, o a otras partículas que ya se han depositado previamente.
La simulación a desarrollar debe:
1) Solicitar forma de la sección y dimensiones del conducto. Las formas pueden ser circulares, cuadradas o rectangulares. Y sus dimensiones deben estar comprendidas entre 1 y 1000 mm.
2) Las partículas serán de forma cuadrada y se establecerán sus dimensiones entre 1 y 10 mm de lado.
3) Las partículas se generan en el centro del conducto y están dotadas de movimiento aleatorio, hacia arriba, abajo, izquierda y derecha.
4) Cuando se encuentran suficientemente próximas, con cierta tolerancia, a las paredes del conducto, deben permanecer adheridas.
5) Cuando se encuentran suficientemente próximas, con cierta tolerancia, a otras partículas, previamente adheridas al conducto, también deben permanecer adheridas.
6) La simulación se detiene cuando el crecimiento de las partículas alcanza una cierta distancia al centro, establecida previamente.

En esta simulación podemos ver cómo será el crecimiento de partículas en un conducto, podemos elegir entre 3 formas distintas de conducto cuadrada, rectangular y circular, además de poder decidir qué dimensiones tiene entre 1 y 1000 mm. También podemos elegir el tamaño de las partículas que pueden ser de entre 1 y 10 mm

El código utilizado fue:

```
import pygame
import random
import math

# Función para obtener la forma del conducto
def get_conducto_shape():
    while True:
          shape = input("Ingrese la forma del conducto (circle, rectangle, square): ").strip().lower()
        if shape in ["circle", "rectangle", "square"]:
            return shape
        else:
                    print("Forma no válida. Por favor, ingrese 'circle', 'rectangle' o 'square'.")

# Función para obtener las dimensiones del conducto
def get_dimension_mm():
    while True:
        try:
```

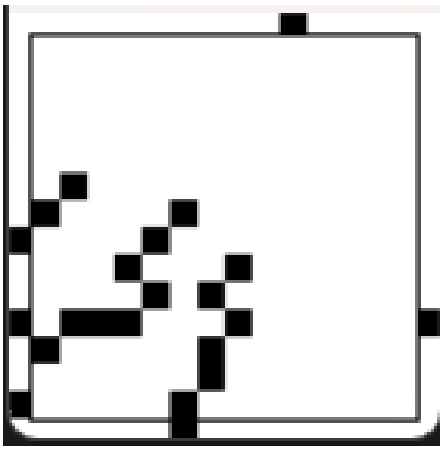
```
            dimension = float(input("Ingrese la dimensión del conducto en
mm (entre 1 y 1000): "))
            if 1 <= dimension <= 1000:
                return dimension
            else:
                 print("Dimensión fuera de rango. Debe estar entre 1 y 1000
mm.")
        except ValueError:
             print("Entrada no válida. Por favor, ingrese un número entre 1
y 1000.")

# Función para obtener las dimensiones de las partículas
def get_particle_size():
    while True:
        try:
             size = float(input("Ingrese el tamaño de las partículas en mm
(entre 1 y 10): "))
            if 1 <= size <= 10:
                return size
            else:
                print("Tamaño fuera de rango. Debe estar entre 1 y 10 mm.")
        except ValueError:
             print("Entrada no válida. Por favor, ingrese un número entre 1
y 10.")

# Obtener forma y dimensiones del conducto y de las partículas
conducto_shape = get_conducto_shape()
dimension_mm = get_dimension_mm()
particle_size_mm = get_particle_size()

# Configuración inicial de Pygame
pygame.init()

# Parámetros del conducto y partículas
tolerance_mm = 10
num_particles = 1000

# Conversión de mm a píxeles (ajustar según la escala deseada)
mm_to_px = 1  # 1 mm = 10 píxeles

# Conversión de dimensiones
dimension_px = int(dimension_mm * mm_to_px)
particle_size_px = int(particle_size_mm * mm_to_px)
```

```
tolerance_px = int(tolerance_mm * mm_to_px)
extra_space_px = int(5 * mm_to_px)  # Espacio adicional de 5 mm alrededor
del conducto

# Configuración de la pantalla
screen_width = dimension_px if conducto_shape != "circle" else dimension_px
* 2
screen_height  =  dimension_px  if  conducto_shape  !=  "circle"  else
dimension_px * 2

# Añadir espacio adicional alrededor del conducto
screen_width += 2 * extra_space_px
screen_height += 2 * extra_space_px

screen = pygame.display.set_mode((screen_width, screen_height))
pygame.display.set_caption("Modelo  Estocástico  de  Crecimiento  por
Agregación en Confinamiento")

# Centro del conducto ajustado con espacio extra
center_x = screen_width // 2
center_y = screen_height // 2

# Función para verificar si una partícula está dentro del conducto
def is_in_conducto(x, y):
    if conducto_shape == "circle":
        radius = dimension_px
        return (x - center_x) ** 2 + (y - center_y) ** 2 <= radius ** 2
    elif conducto_shape in ["rectangle", "square"]:
          return extra_space_px <= x < screen_width - extra_space_px and
extra_space_px <= y < screen_height - extra_space_px

# Inicializar la lista de partículas
particles = []

# Función para mover las partículas
def move_particle(particle):
        directions  =  [(0,  particle_size_px),  (0,  -particle_size_px),
(particle_size_px, 0), (-particle_size_px, 0)]
    move = random.choice(directions)
    return (particle[0] + move[0], particle[1] + move[1])

# Función para verificar si una partícula está adherida
def is_stuck(particle):
```

```
    x, y = particle
    if not is_in_conducto(x, y):
        return True
    for p in particles:
        if math.hypot(p[0] - x, p[1] - y) <= tolerance_px:
            return True
    return False

# Loop principal de la simulación
running = True
clock = pygame.time.Clock()
particle_count = 0

while running:
    screen.fill((255, 255, 255))  # Limpiar la pantalla con color blanco

    # Dibujar el contorno del conducto
    if conducto_shape == "circle":
            pygame.draw.circle(screen, (0, 0, 0), (center_x, center_y),
dimension_px, 1)
    elif conducto_shape in ["rectangle", "square"]:
                pygame.draw.rect(screen, (0, 0, 0), (extra_space_px,
extra_space_px, screen_width - 2 * extra_space_px, screen_height - 2 *
extra_space_px), 1)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Generar y mover una nueva partícula si no se ha alcanzado el número
máximo
    if particle_count < num_particles:
        new_particle = (center_x, center_y)
        while not is_stuck(new_particle):
            new_particle = move_particle(new_particle)
             screen.fill((255, 255, 255))  # Limpiar la pantalla con color
blanco

            # Dibujar el contorno del conducto
            if conducto_shape == "circle":
                pygame.draw.circle(screen, (0, 0, 0), (center_x, center_y),
dimension_px, 1)
            elif conducto_shape in ["rectangle", "square"]:
```

```
                    pygame.draw.rect(screen, (0, 0, 0), (extra_space_px,
extra_space_px, screen_width - 2 * extra_space_px, screen_height - 2 *
extra_space_px), 1)

            # Dibujar todas las partículas
            for particle in particles:
                        pygame.draw.rect(screen, (0, 0, 0), (particle[0],
particle[1], particle_size_px, particle_size_px))

            # Dibujar la nueva partícula en movimiento
                    pygame.draw.rect(screen, (0, 0, 0), (new_particle[0],
new_particle[1], particle_size_px, particle_size_px))

            pygame.display.flip()  # Actualizar la pantalla
            clock.tick(30)  # Ajustar la velocidad de la animación

        particles.append(new_particle)
        particle_count += 1

# Mantener la ventana abierta para observar el resultado final
finished = True
while finished:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            finished = False
    pygame.time.wait(100)

pygame.quit()
```

La simulación una vez terminada para un conducto cuadrado de 100 mm con un tamaño de partículas de 7 mm:



La simulación una vez terminada para un conducto circular de 100 mm con un tamaño de partículas de 10 mm: