Reporte de Simulación en Python: Desplazamiento de Puntos y Cambio de Fondo

Alan Miranda, Cristian Orozco, Marlon Caviedes
Infotep

3 de marzo de 2025

Resumen

Este reporte muestra una simulación sencilla en Python en la que se desplazan puntos hacia la derecha. Si los puntos salen del área definida, se generan nuevos de forma aleatoria dentro del cuadro. Además, se cambia el color de fondo en cada paso de la animación. Aquí se explica el código, se muestran algunas imágenes de la simulación y se analizan brevemente los resultados.

1. Introducción

En este trabajo se exploró el uso de Python y Jupyter para crear animaciones simples con `matplotlib`. Se desarrolló una simulación donde unos puntos se mueven hacia la derecha en una gráfica. Cuando un punto se sale del límite del área, se genera de nuevo en una posición aleatoria. También se implementa un cambio en el color del fondo para hacer la animación más dinámica.

2. Objetivos

Los objetivos de esta actividad fueron:

- Aprender a mover puntos de forma vectorizada en Python.
- Regenerar puntos que se salen del área.
- Cambiar el color del fondo durante la animación.
- Documentar el proceso en un reporte.

3. Descripción de la Actividad y Código

La simulación se basa en una clase llamada `Escena` que:

1. Inicializa la escena con un tamaño específico, color de fondo y una cantidad de puntos generados aleatoriamente.
2. Dibuja la escena de manera estática.
3. Crea una animación en la que los puntos se desplazan hacia la derecha. Si algún punto se sale del límite, se le asigna una nueva posición aleatoria.

El siguiente fragmento muestra el código principal:

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as patches
from matplotlib import animation

class Escena:
    def __init__(self, width, height, color, n_points):
        self.width = width
        self.height = height
        self.color = color
        self.n_points = n_points
```

```
        self.points = np.column_stack((
            np.random.uniform(0, width, n_points),
            np.random.uniform(0, height, n_points)
        ))

    def dibujar(self):
        fig, ax = plt.subplots()
        rect = patches.Rectangle((0, 0), self.width, self.height, facecolor=self.
            color)
        ax.add_patch(rect)
        ax.scatter(self.points[:, 0], self.points[:, 1], c='red')
        ax.set_xlim(0, self.width * 1.1)
        ax.set_ylim(0, self.height * 1.1)
        ax.set_aspect('equal')
        plt.show()

    def animar(self, velocidad, frames=50, interval=100):
        fig, ax = plt.subplots()
        rect = patches.Rectangle((0, 0), self.width, self.height, facecolor=self.
            color)
        ax.add_patch(rect)
        scatter = ax.scatter(self.points[:, 0], self.points[:, 1], c='red')
        ax.set_xlim(0, self.width * 1.1)
        ax.set_ylim(0, self.height * 1.1)
        ax.set_aspect('equal')

        def update(frame):
            self.points[:, 0] += velocidad
            # Si un punto sale del l mite, se le asigna una nueva posici n
                aleatoria
            mask = self.points[:, 0] > self.width
            if np.any(mask):
                num_extras = np.count_nonzero(mask)
                self.points[mask, 0] = np.random.uniform(0, self.width, num_extras)
                self.points[mask, 1] = np.random.uniform(0, self.height, num_extras
                    )
            scatter.set_offsets(self.points)
            # Cambia el color de fondo seg n el frame actual
            nuevo_color = plt.cm.hsv(frame / frames)
            rect.set_facecolor(nuevo_color)
            return scatter, rect

        anim = animation.FuncAnimation(fig, update, frames=frames, interval=
            interval, blit=True)
        plt.close(fig)
        return anim
```

Listing 1: Código en Python para la simulación

4. Resultados

Para la simulación se generaron dos imágenes:

- **Imagen inicial:** Se observa la distribución aleatoria de los puntos en la escena con el fondo en su color original.
- **Imagen durante la animación:** Se muestra un cuadro intermedio donde se aprecia el movimiento de los puntos y el cambio progresivo del color de fondo.

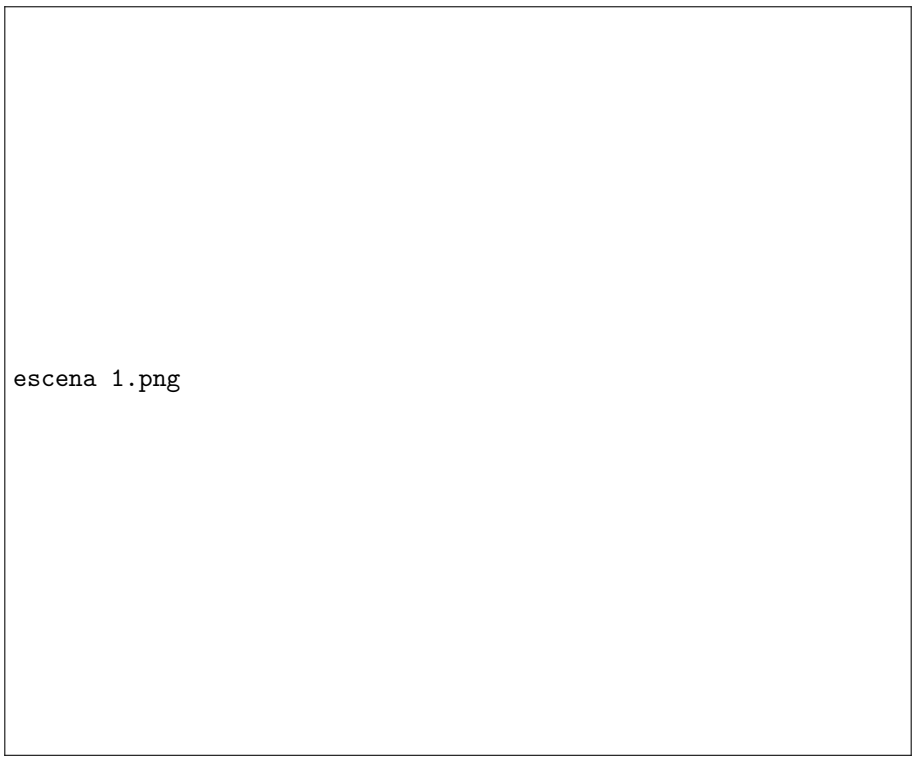

Figura 1: Imagen inicial de la simulación.

Figura 2: Cuadro de la animación con cambio de fondo.

5. Análisis e Interpretación

En la Figura 1 se puede ver la distribución inicial de los puntos, la cual es completamente aleatoria. Durante la animación (Figura 2), los puntos se mueven hacia la derecha de forma uniforme. Cuando alguno sale del límite del área, se le asigna una nueva posición dentro del cuadro, lo que permite que el movimiento sea continuo. Además, el cambio de color del fondo se logra de forma sencilla usando un colormap, lo que añade un efecto visual interesante.

6. Conclusiones

Esta actividad permitió aprender conceptos básicos sobre el manejo de gráficos y animaciones en Python usando Jupyter y `matplotlib`. En resumen:

- Se logró mover puntos de forma vectorizada.
- Se implementó la regeneración de puntos cuando salen del área.
- Se añadió un efecto visual al cambiar el color de fondo.
- Se consolidó el uso de Jupyter y Python para tareas de simulación.