SEMANA 7

📘 Lectura: Visualización de Datos con Processing

1. Introducción y Conceptos Generales

La visualización de datos es una disciplina que combina elementos de la programación, el diseño gráfico y el análisis de información para convertir datos complejos en representaciones visuales comprensibles. Estas pueden tomar la forma de gráficos, diagramas, mapas interactivos o animaciones, y son fundamentales para facilitar la toma de decisiones, identificar patrones o comunicar ideas complejas de manera intuitiva.

Processing es una herramienta ideal para este tipo de trabajo. Se trata de un entorno de desarrollo (IDE) y lenguaje de programación basado en Java, creado específicamente para artistas digitales, diseñadores y desarrolladores creativos. Su sintaxis sencilla y su potente capacidad gráfica lo convierten en una herramienta perfecta tanto para aprender como para crear proyectos avanzados de visualización de datos.

Una característica clave de Processing es que permite crear gráficos dinámicos e interactivos con pocos recursos computacionales, lo que lo hace ideal para estudiantes y profesionales que buscan explorar visualizaciones en tiempo real.

---

2. Importación y Manipulación de Datos

Para poder visualizar datos, primero debemos importarlos a nuestro entorno de trabajo. En Processing, esto se logra principalmente utilizando funciones integradas como `loadTable()` para archivos CSV y TSV, o librerías externas para formatos como JSON o XML.

Un **archivo CSV** (Comma-Separated Values) es uno de los formatos más comunes para almacenar datos tabulares. Por ejemplo:

```
Nombre,Edad,Ciudad
Ana,25,Madrid
```

```
Carlos,30,Bogotá
Lucía,28,Lima
```

En Processing, podemos cargar este archivo con una línea:

```
Table datos = loadTable("datos.csv", "header");
```

Esto carga el archivo y reconoce la primera fila como encabezado. Una vez cargado, podemos recorrerlo y acceder a los valores:

```
for (TableRow fila : datos.rows()) {
  String nombre = fila.getString("Nombre");
  int edad = fila.getInt("Edad");
  // Hacer algo con esos datos
}
```

Además del uso directo de datos estáticos, también podemos manipular, filtrar, ordenar o agrupar estos datos dentro del propio programa antes de proceder a su representación visual.

---

3. Creación de Gráficos y Animaciones

Processing no solo permite mostrar datos, sino también darles vida mediante gráficos y animaciones. Esto permite mostrar cómo cambian los datos a través del tiempo o responder a acciones del usuario.

Tipos de gráficos básicos

- **Gráfico de barras**: Ideal para comparar magnitudes.
- **Gráfico circular**: Útil para mostrar proporciones.
- **Gráfico de líneas**: Excelente para series temporales.
- **Gráfico de dispersión (scatter plot)**: Para ver relaciones entre variables.

Por ejemplo, para dibujar un rectángulo que represente el valor de una variable:

```
rect(50, 100, 20, -edad * 2);
```

Este código crea una barra cuya altura depende del valor de `edad`.

Animación básica

Processing tiene una estructura muy clara de ejecución:

- La función `setup()` se ejecuta una sola vez al inicio.
- La función `draw()` se repite constantemente, lo que permite crear animaciones.

Un ejemplo sencillo de animación es hacer que un círculo se mueva horizontalmente:

```
float x = 0;

void draw() {
  background(255);
  ellipse(x, 100, 30, 30);
  x += 1;
  if (x > width) x = 0;
}
```

Al combinar animación con datos, puedes crear gráficos que se actualizan en tiempo real o reaccionan a cambios en los datos.

---

4. Interacción con Visualizaciones

Una buena visualización no solo muestra datos, sino que también permite al usuario interactuar con ellos. En Processing, esto se logra fácilmente gracias a las funciones de manejo de eventos del ratón y el teclado.

Eventos del mouse

- `mousePressed()` – Detecta cuando se hace clic.
- `mouseX` / `mouseY` – Indican la posición del puntero.
- `mouseDragged()` – Detecta cuando se arrastra el ratón.

Por ejemplo, podrías permitir al usuario seleccionar una barra en un gráfico de barras haciendo clic sobre ella:

```
void mousePressed() {
  if (mouseX > 50 && mouseX < 70 && mouseY > 80 && mouseY < 100) {
    println("Barra seleccionada");
  }
}
```

Eventos del teclado

- `keyPressed()` – Detecta cuando se presiona una tecla.
- `key` – Devuelve qué tecla fue pulsada.

Esto puede servir para activar funciones como pausar una animación o cambiar entre modos de visualización.

La interacción permite construir visualizaciones más ricas y personalizables, donde el usuario puede explorar datos por sí mismo.

---

5. Visualización con Datos en Tiempo Real

Visualizar **datos en tiempo real** implica actualizar continuamente la pantalla con nuevos valores provenientes de fuentes como sensores, dispositivos móviles o APIs.

¿Cómo funciona?

El flujo típico es:

1. Recibir nuevos datos (por ejemplo, desde un sensor o un puerto serial).
2. Procesarlos (normalizar, filtrar, escalar).
3. Dibujarlos (actualizar el gráfico o animación).

En Processing, esto se logra usando bucles que leen datos periódicamente y redibujan la interfaz en la función `draw()`.

Un ejemplo común es visualizar datos recibidos por puerto serial, como la temperatura medida por un sensor conectado a Arduino.

---

6. Ejemplo Práctico con Arduino

Para ilustrar todo lo anterior, hagamos un ejemplo práctico: conectar un sensor analógico a Arduino, enviar los datos vía comunicación serial a Processing y graficarlos en tiempo real.

Paso 1: Configuración en Arduino

Conectamos un sensor analógico (por ejemplo, un termistor o LDR) al pin A0 de Arduino y enviamos sus valores por serial:

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  int valor = analogRead(A0);
  Serial.println(valor);
  delay(100);
}
```

Paso 2: Recepción en Processing

En Processing usamos la biblioteca `Serial` para recibir los datos del puerto serial:

```
import processing.serial.*;

Serial miPuerto;
float valorSensor;

void setup() {
  size(400, 300);
  String portName = Serial.list()[0];
  miPuerto = new Serial(this, portName, 9600);
}

void draw() {
  background(255);
  if (miPuerto.available() > 0) {
    valorSensor = float(miPuerto.readStringUntil('\n'));
  }
  fill(0);
  text("Valor: " + valorSensor, 10, 20);
  rect(50, height, 50, -valorSensor);
}
```

Este código lee los datos del sensor, los muestra como texto y crea una barra vertical cuya altura depende del valor recibido.

---

7. Conclusión

Processing es una plataforma versátil y accesible para la creación de visualizaciones de datos. Permite desde tareas simples como graficar datos estáticos hasta aplicaciones complejas con interacción y animación en tiempo real. Al integrarlo con dispositivos físicos como Arduino, se abre la puerta a proyectos interdisciplinarios que combinan hardware, software y diseño.

Este conjunto de habilidades es especialmente útil en campos como:

- Ciencia de datos
- Diseño interactivo
- Arte digital
- Educación tecnológica
- IoT (Internet de las Cosas)

Aprender a usar Processing para visualizar datos no solo mejora nuestra capacidad de análisis, sino también nuestra capacidad de contar historias con datos, algo cada vez más importante en el mundo actual.

🔗 Referencias:

- Sitio oficial de Processing
- Documentación de Arduino
- The Nature of Code – Daniel Shiffman
- Learning Processing – Daniel Shiffman
- Repositorios en GitHub con ejemplos de comunicación serial entre Arduino y Processing

---