INSTITUTO TECNOLÓGICO SUPERIOR PROGRESO

Carrera:

Ingeniería en sistemas computacionales

Materia:

Graficación

Actividad:

Reporte de practica 1-2

Docente:

Dr. Holzen Atocha Martínez García

Alumno:

David Ezequiel Caballero González

Fecha de entrega:

Lunes 8 de septiembre de 2025

Práctica 1-2

Introducción

En esta práctica se implementó un programa en Processing que genera un gradiente vertical entre dos colores definidos por el usuario, utilizando funciones como ***setup()***, ***gradient()*** y ***setGradient()***, se logra una transición suave de color desde la parte superior hasta la inferior de la ventana. Esta práctica permite explorar cómo se pueden manipular componentes RGB y aplicar interpolación de color para crear efectos visuales atractivos, además, se refuerza el uso de estructuras como bucles y funciones personalizadas dentro del entorno gráfico de Processing.

- El código puede verse en el repositorio en GitHub: Graficación Práctica 1 Práctica 1-2

Objetivo

Desarrollar un programa que genere un gradiente de color vertical en toda la ventana, interpolando entre dos colores definidos por sus componentes RGB.

Marco teórico

Processing es un lenguaje de programación basado de Java y está pensado para crear dibujos, animaciones y cosas interactivas. Su editor (PDE) es muy práctico porque te deja escribir código de manera rápida y sin tantas complicaciones, perfecto si apenas estás empezando con la programación visual.
En esta práctica se utiliza la función ***lerpColor()***, que permite interpolar entre dos colores, y ***map()***, que ajusta valores dentro de un rango específico. El gradiente se genera trazando líneas horizontales con colores que cambian progresivamente entre dos extremos definidos por componentes RGB y la constante ***Y_AXIS*** se usa para indicar que el gradiente debe aplicarse en dirección vertical.

Procedimiento

1. Se define la constante ***Y_AXIS = 1*** para indicar la dirección del gradiente.

```
// Constante para dirección vertical
int Y_AXIS = 1;
```

2. En ***setup()***, se establece el tamaño de la ventana ***(500x500)*** y se llama a ***gradient()*** con los colores rojo ***(255, 0, 0)*** y azul *(0, 0, 255)*.

```
void setup() {
  size(500, 500);

  // Llamamos a la función con colores definidos por componentes RGB
  gradient(255, 0, 0, 0, 0, 255); // Rojo a azul
}
```

3. La función ***gradient()*** convierte los valores RGB en objetos color y llama a ***setGradient()*** para aplicar el gradiente.

```
// Función que genera el gradiente vertical
void gradient(int r1, int g1, int b1, int r2, int g2, int b2) {
  color c1 = color(r1, g1, b1);
  color c2 = color(r2, g2, b2);
  setGradient(0, 0, width, height, c1, c2, Y_AXIS);
}
```

4. En ***setGradient()***, se recorre la altura de la ventana con un bucle ***for***, calculando la interpolación entre los colores con ***map()*** y ***lerpColor()***.

```
// Función base tomada del ejemplo de Processing
void setGradient(int x, int y, float w, float h, color c1, color c2, int axis) {

  if (axis == Y_AXIS) {
    for (int i = y; i <= y + h; i++) {
      float inter = map(i, y, y + h, 0, 1);
      color c = lerpColor(c1, c2, inter);
      stroke(c);
      line(x, i, x + w, i);
    }
  }
}
```

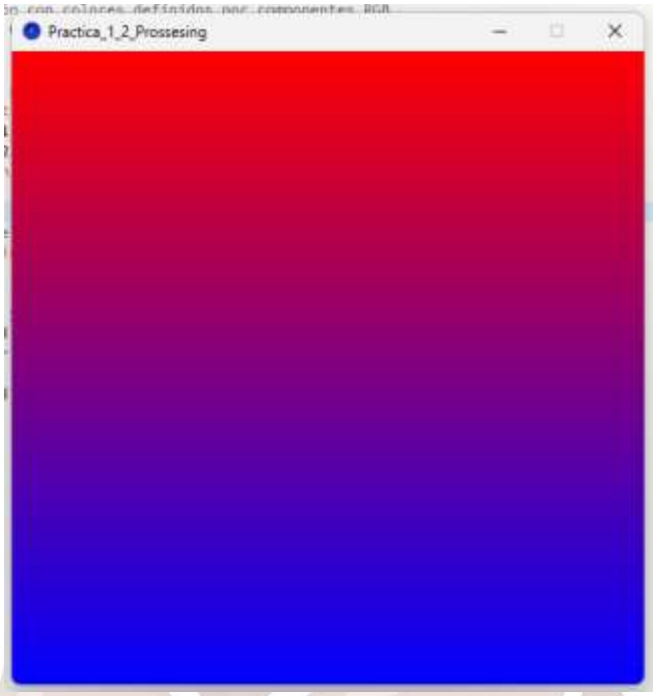
5. Se dibujan líneas horizontales con ***stroke(c)*** y ***line()*** para representar el gradiente.

Materiales y equipos utilizados

- Laptop
- Processing
- Página oficial de Processing

Resultados

Al ejecutar el programa, se genera una ventana de 500x500 píxeles con un gradiente vertical que va del rojo en la parte superior al azul en la parte inferior. La transición de color es suave y continua, mostrando cómo se puede manipular el color en Processing de forma precisa y el código es flexible y puede adaptarse fácilmente para crear gradientes horizontales o entre otros colores.



Análisis

Esta práctica demuestra cómo se puede aplicar interpolación de color para generar efectos visuales en Processing. La función ***lerpColor()*** permite crear transiciones suaves entre dos colores y el uso de ***map()*** facilita el control del rango de interpolación.
El gradiente vertical se logra trazando líneas horizontales con colores que cambian progresivamente, lo que refuerza el concepto de visualización basada en bucles y coordenadas, además el código permite reutilizar la función ***setGradient()*** para diferentes combinaciones de color y dirección.

Discusión

La práctica permitió entender cómo se puede controlar el color en Processing utilizando componentes RGB y funciones de interpolación.
El gradiente generado muestra claramente cómo se puede pasar de un color a otro de forma gradual y también se aprendió la importancia de organizar el código en funciones reutilizables, lo que facilita la modificación y expansión del programa.

Conclusión

Esta práctica fue útil para comprender cómo se pueden aplicar conceptos de interpolación y color en la programación gráfica, logrando generar un gradiente vertical entre dos colores utilizando Processing, reforzando el uso de funciones como ***lerpColor()*** y estructuras como bucles ***for***. Además, se destacó la importancia de modularizar el código y trabajar con componentes RGB para lograr efectos visuales precisos y el conocimiento adquirido será valioso para proyectos más complejos que involucren diseño gráfico y visualización interactiva.

Referencias

- Processing Foundation. (n.d.). Simple linear gradient. Processing. Recuperado de https://processing.org/examples/lineargradient.html
- del Valle Hernández, L. (s.f.). Processing, el lenguaje para gráficos. Programar Fácil Podcast #80. Recuperado de https://programarfacil.com/podcast/80-processing-lenguaje-para-graficos/