Exemple: inversió de colors o BN

Aquest exemple mostra la **manipulació de píxels** per aplicar un filtre (en aquest cas, la inversió de colors) a una imatge utilitzant l'objecte `ImageData` del Canvas.

L'operació de **inversió de colors** transforma cada color al seu complementari (negatiu). En el model **RGBA**, on el valor màxim per a cada component és **255**, la fórmula per a la inversió és simple:

Nou Valor = 255 - Valor Original

Un píxel amb un valor de Vermell de **200** passarà a ser 255 - 200 = 55. El blanc pur (R=255, G=255, B=255) es converteix en negre pur (R=0, G=0, B=0).

Passos a seguir

1. **Carregar i Dibuixar la Imatge:**
   - S'utilitza `window.onload` per assegurar que tant el Canvas com la imatge (`<img id="enfeinada">`) s'han carregat abans d'executar el codi.
   - `ctx.drawImage(srcImg, 0, 0, ctx.canvas.width, ctx.canvas.height)`: Dibuixa la imatge sobre el Canvas, escalant-la a les dimensions del Canvas.
2. **Llegir les Dades (`getImageData`):**
   - `var imgData = ctx.getImageData(0, 0, ctx.canvas.width, ctx.canvas.height)`: Aquesta funció llegeix el contingut del Canvas a l'àrea especificada i retorna l'objecte `ImageData`.
   - `var pixels = imgData.data`: Accedeix a l'array `Uint8ClampedArray` que conté els valors $\text{R}, \text{G}, \text{B}, \text{A}$ consecutius.
3. **Bucle de Transformació:**
   - `for (var i = 0; i < pixels.length; i += 4)`: El bucle itera sobre l'array de píxels de 4 en 4, processant un píxel complet a cada pas.
   - `pixels[i] = 255 - pixels[i];`: Aplica la fórmula d'inversió individualment a les components **Vermella** (`i`), **Verda** (`i + 1`) i **Blava** (`i + 2`). L'opacitat (`i + 3`) es manté sense canvis.
4. **Dibuixar les Dades Modificades (`putImageData`):**
   - `ctx.putImageData(imgData, 0, 0, 25, 25, 200, 182)`: Aquesta és la part més complexa i que conté una subtilesa. Els últims quatre paràmetres (`dirtyX`, `dirtyY`, `dirtyWidth`, `dirtyHeight`) s'utilitzen per indicar al navegador que només ha de redibuixar una sub-regió de les dades de píxels, optimitzant el rendiment.
     - En aquest cas: Dibuixa l'objecte `imgData` a `(0, 0)`, però només utilitza la sub-regió de l'`imgData` que comença a `(25, 25)` i té una mida de `200x182`. **Això fa que només una part de la imatge modificada sigui visible, cosa que pot ser un error si es vol mostrar el filtre complet.**

Exemple

Millorem la llegibilitat, fem l'extracció de dades més robusta i simplifiquem l'ús de `putImageData` per mostrar el filtre complet.

HTML

```
<img id="imatgeOriginal" src="https://siensis.com/img/patterns-canvas.jpg" alt="imatge
a processar">
<canvas width="300" height="300" id="llencolFiltre" style="border: 1px solid #ddd;"></
canvas>

<script>
    window.onload = function() {
        const llencol = document.getElementById("llencolFiltre");
        const ctx = llencol.getContext("2d");

        if (ctx) {
            const recursImatge = document.getElementById("imatgeOriginal");

            // 1. Dibuixar la imatge base i assegurar que el canvas té la mida
correcta
            const ample = llencol.width;
            const alçada = llencol.height;

            // Dibuixa la imatge escalada al Canvas
            ctx.drawImage(recursImatge, 0, 0, ample, alçada);

            // 2. Extreure les dades de píxels
            const dadesImatge = ctx.getImageData(0, 0, ample, alçada);
            const arrayPixels = dadesImatge.data;

            // 3. Aplicar el Filtre d'Inversió
            // La matriu de píxels és (ample * alçada * 4)
            for (let i = 0; i < arrayPixels.length; i += 4) {
                // Invertim el Vermell (R)
                arrayPixels[i + 0] = 255 - arrayPixels[i + 0];

                // Invertim el Verd (G)
                arrayPixels[i + 1] = 255 - arrayPixels[i + 1];

                // Invertim el Blau (B)
                arrayPixels[i + 2] = 255 - arrayPixels[i + 2];

                // L'Alpha (A) es manté intacte (arrayPixels[i + 3])
            }

            // 4. Col·locar la imatge modificada al Canvas (Filtre Complet)
            // Utilitzem la versió de 3 paràmetres per mostrar tota la imatge a (0, 0)
```

```
            ctx.putImageData(dadesImatge, 0, 0);

            // *NOTEM: Si es volia mostrar la imatge original i la invertida una al
costat de l'altra:
            // ctx.drawImage(recursImatge, 0, 0); // Original a l'esquerra
            // ctx.putImageData(dadesImatge, ample, 0); // Invertida a la dreta
        }
    }
</script>
```

Extra: Filtre Threshold

Un altre exemple interessant de manipulació de píxels és el **Filtre Threshold**, que converteix la imatge a blanc i negre pur, basant-se en un llindar de brillantor. Si la mitjana dels colors d'un píxel supera el llindar (p. ex., 128), el píxel es converteix en blanc (255, 255, 255); si no, es converteix en negre (0, 0, 0).

JavaScript

```
    // --- Funció de Filtre de Passada (Threshold) ---
    function aplicarFiltrePassada(dadesImatge, llindar = 128) {
        const arrayPixels = dadesImatge.data;

        for (let i = 0; i < arrayPixels.length; i += 4) {
            const R = arrayPixels[i + 0];
            const G = arrayPixels[i + 1];
            const B = arrayPixels[i + 2];

            // Calcula la brillantor (escala de grisos)
            const brillantor = (R + G + B) / 3;

            let nouColor;
            if (brillantor > llindar) {
                nouColor = 255; // Blanc
            } else {
                nouColor = 0;   // Negre
            }

            // Assigna el nou valor (blanc o negre) a R, G, B
            arrayPixels[i + 0] = nouColor;
            arrayPixels[i + 1] = nouColor;
            arrayPixels[i + 2] = nouColor;
            // A es manté
        }
    }
```

```
    // Ús (dins de window.onload):
    // const dadesImatge = ctx.getImageData(0, 0, ample, alçada);
    // aplicarFiltrePassada(dadesImatge, 100); // Prova amb un llindar diferent
    // ctx.putImageData(dadesImatge, 0, 0);
```

Revisió núm. 2
★Admin l'ha creat 2025-11-19 11:28:01 UTC
Admin l'ha actualitzat 2025-11-19 11:35:08 UTC