EREN ÇAĞLAROĞLU CMPE360 HOMEWORK1 SEC03
11500168978

CMPE 360 HOMEWORK1

Introduction

This report offers a thorough description of the composite JavaScript function's operation. The composite function uses alpha (transparent) blending to combine background and foreground images. Four input parameters are required by the function: BackGround (the background image), ForeGround (the foreground image), ForeGroundOpacity (the foreground's opacity), and ForeGroundPosition (the foreground's location). Let's examine this function's operational logic.

Functions

```
function composite(BackGround, ForeGround, ForeGroundOpacity, ForeGroundPosition) {
    var bgData = BackGround.data;
    var fgData = ForeGround.data;
    var width = BackGround.width;
    var height = BackGround.height;
```

To access the pixel information of the BackGround and ForeGround images, the function initializes the arrays bgData and fgData.The background image's width and height are stored in the width and height variables.

```
    // Calculate the bounds for the foreground image
    var x = ForeGroundPosition.x;
    var y = ForeGroundPosition.y;
    var minX = Math.max(x, 0);
    var minY = Math.max(y, 0);
    var maxX = Math.min(x + ForeGround.width, width);
    var maxY = Math.min(y + ForeGround.height, height);
```

The foreground's position in relation to the backdrop is determined by the coordinates x and y.The foreground zones that will be blended are determined by the minX, minY, maxX, and maxY variables. By doing this, it is made sure that the foreground does not go over the limits of the background. Each pixel is processed through two nested loops (py and px) to blend the foreground and background together.

```
for (var py = minY; py < maxY; py++) {
    for (var px = minX; px < maxX; px++) {
        var bgIdx = (py * width + px) * 4; // Background pixel index
        var fgIdx = ((py - y) * ForeGround.width + (px - x)) * 4; // Foreground pixel index

        // Extract the color components
        var bgA = bgData[bgIdx + 3]; // Background alpha
        var fgA = fgData[fgIdx + 3]; // Foreground alpha

        // Calculate the blended color
        var alpha = ForeGroundOpacity * fgA / 255;
        var blendedR = Math.round(bgData[bgIdx] * (1 - alpha) + fgData[fgIdx] * alpha);
        var blendedG = Math.round(bgData[bgIdx + 1] * (1 - alpha) + fgData[fgIdx + 1] * alpha);
        var blendedB = Math.round(bgData[bgIdx + 2] * (1 - alpha) + fgData[fgIdx + 2] * alpha);

        // Update the background pixel data with the blended color
        bgData[bgIdx] = blendedR;
        bgData[bgIdx + 1] = blendedG;
        bgData[bgIdx + 2] = blendedB;
    }
}
```

To access the pixel data of the background and foreground, bgIdx and fgIdx are computed.The foreground's (fgA) and background's (bgA) alpha values are then determined. The pixel's transparency is determined by alpha.Alpha mixing is taken into consideration as the color components of the pixels (bgR, bgG, bgB, fgR, fgG, fgB) are mixed.The background's pixel data is lastly updated with the blended color values.

Conclusion

In this manner, the composite function utilizes alpha blending between the background and foreground photos to produce the desired outcome. The resultant image is created by suitably merging transparent foreground pixels (i.e., pixels with a non-black background) with the background. An essential idea in image processing is implemented in this piece of code.