Interpolated Colored Line Drawing Using WinAPI

Overview

This program demonstrates how to draw an interpolated colored line between two points using the Windows API (WinAPI). The implementation utilizes linear interpolation to smoothly transition colors along the drawn line.

Features

- **Mouse Interaction**: The user clicks to define the start and end points of the line.
- **Color Interpolation**: The line transitions smoothly from one color to another.
- **Efficient Rendering**: Utilizes WinAPI functions for rendering pixels directly.

Code Breakdown

1. Header Files and Namespace

```
#include <windows.h>
#include <cmath>
using namespace std;
```

- `windows.h`: Provides access to WinAPI functions.
- `cmath`: Used for mathematical calculations.

2. Rounding Function

```
int Round(double n) {
    return static_cast<int>(n + 0.5);
}
```

- Ensures accurate rounding of floating-point numbers to integers.

3. Extracting RGB Components

```
void GetCR(COLORREF c, int& r, int& g, int& b) {
    r = GetRValue(c);
    g = GetGValue(c);
    b = GetBValue(c);
}
```

- Extracts red, green, and blue components from a `COLORREF` color value.

4. Drawing an Interpolated Line

```
void InterpolatedColoredLine(HDC hdc, int x1, int y1, int x2,
int y2, COLORREF c1, COLORREF c2) {
    int diffX = x2 - x1, diffY = y2 - y1, x, y, r1, g1, b1, r2,
g2, b2;
    double step = 1.0 / max(abs(diffX), abs(diffY));

    GetCR(c1, r1, g1, b1);
    GetCR(c2, r2, g2, b2);

    for (double t = 0; t <= 1; t += step) {
        x = Round(diffX * t + x1);
        y = Round(diffY * t + y1);

        int r = Round(r1 + t * (r2 - r1));
        int g = Round(g1 + t * (g2 - g1));
        int b = Round(b1 + t * (b2 - b1));

        SetPixel(hdc, x, y, RGB(r, g, b));
    }
}
```

- Uses linear interpolation to gradually change the RGB values along the line.
- `SetPixel` sets each interpolated pixel color.

5. Window Procedure for Handling Mouse Events

```
LRESULT WindowProc(HWND hwnd, UINT m, WPARAM wp, LPARAM lp) {
    HDC hdc;
    static int x1, y1, x2, y2;
    static bool draw = false;

    if (m == WM_LBUTTONDOWN) {
        x1 = LOWORD(lp);
        y1 = HIWORD(lp);
        draw = true;
    }
    else if (m == WM_LBUTTONUP) {
        x2 = LOWORD(lp);
        y2 = HIWORD(lp);
        draw = false;
        hdc = GetDC(hwnd);
        InterpolatedColoredLine(hdc, x1, y1, x2, y2, RGB(0, 0,
0), RGB(255, 0, 255));
        ReleaseDC(hwnd, hdc);
    }
    else if (m == WM_DESTROY) {
        PostQuitMessage(0);
    }
    else {
        return DefWindowProc(hwnd, m, wp, lp);
    }
    return 0;
}
```

- Handles left mouse button events to determine the start and end points of the line.
- Calls `InterpolatedColoredLine` to draw the gradient-colored line.

6. Windows Application Entry Point

```
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
LPSTR lpCmdLine, int nCmdShow) {
    WNDCLASS wc = {};
    wc.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
    wc.hCursor = LoadCursor(nullptr, IDC_CROSS);
    wc.hIcon = LoadIcon(nullptr, IDI_APPLICATION);
    wc.lpszClassName = "LineWindowClass";
    wc.lpfnWndProc = WindowProc;
    wc.style = CS_HREDRAW | CS_VREDRAW;
    wc.hInstance = hInstance;

    RegisterClass(&wc);

    HWND hwnd = CreateWindow("LineWindowClass", "Interpolated
Line Drawing", WS_OVERLAPPEDWINDOW,
                             CW_USEDEFAULT, CW_USEDEFAULT, 700,
700, nullptr, nullptr, hInstance, nullptr);

    ShowWindow(hwnd, nCmdShow);
    UpdateWindow(hwnd);

    MSG msg = {};
    while (GetMessage(&msg, nullptr, 0, 0)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
    return 0;
}
```

- Registers and creates the window.
- Runs the main message loop to process user interactions.

How It Works

1. **User clicks the left mouse button**: Marks the start point.
2. **User releases the left mouse button**: Marks the endpoint and draws the interpolated colored line.
3. **Color smoothly transitions** from black (`RGB(0,0,0)`) to magenta (`RGB(255,0,255)`).

Possible Enhancements

- Allow users to select different colors.
- Implement anti-aliasing techniques for smoother lines.
- Support additional drawing algorithms.

Conclusion

This program showcases a fundamental application of color interpolation in computer graphics using WinAPI. It demonstrates efficient pixel manipulation and user interaction for simple rendering tasks.