Autoscrolling on drag, part 4: Dynamic autoscroll based on escape velocity

devblogs.microsoft.com/oldnewthing/20210128-00

January 28, 2021

Raymond Chen

Last time, we noted some problems with autoscroll based on the distance from the mouse to the window. An alternate design is to use the speed with which the mouse exited the window to establish the autoscroll speed. This no longer penalizes people with small screens or windows near the edge of the screen. Anybody can just slam the mouse into the edge of the screen, and it will scroll quickly.

It so happens that this is the algorithm recommended by the Windows 95 User Interface Guidelines, but in practice nobody ever implemented it.

Let's try to implement it.

The trick here is to use `GetMouseMovePointsEx` to read the mouse position history prior to the mouse motion that caused the mouse to exit the window.[1]

```
int g_dyAutoScroll = 0; // nonzero when autoscrolling

void CancelAutoScroll(HWND hwnd)
{
    KillTimer(hwnd, IDT_AUTOSCROLL);
    g_dyAutoScroll = 0;
}
```

When we cancel autoscrolling, we set the autoscroll amount to zero so we know that autoscrolling isn't happening any more.

```
int DetectAutoScroll(POINT pt)
{
    if (pt.y <= g_cyLine) return -1;
    if (pt.y >= g_cyWindow - g_cyLine) return +1;
    return 0;
}
```

We replace `TryAutoScroll` with this function that detects whether the mouse is in a place that can trigger autoscroll, and if so, whether the user is trying to scroll backward or forward. Note that we tweak the formula so that there is a band inside the client area at the top and bottom which are considered trigger locations. That allows us to autoscroll maximized windows.

```
void HandleDragMouseMove(HWND hwnd, POINT pt)
{
    if (DetectAutoScroll(pt)) {
        ScrollDelta(hwnd, g_dyAutoScroll);
    } else {
        CancelAutoScroll(hwnd);
    }
}
```

The new version of `HandleDragMouseMove` checks if the mouse is still in a place that activates autoscroll. If so, then we continue the autoscroll. Otherwise, we cancel it.

The fancy stuff happens when we detect that we should start autoscrolling:

```
void OnMouseMove(HWND hwnd, int x, int y, UINT keyFlags)
{
    if (g_fDragging) {
        if (g_dyAutoScroll == 0) {
            int direction = DetectAutoScroll({ x, y });
            if (direction) {
                DWORD tmTimer = GetDoubleClickTime() / 5;
                auto [distance, time] = GetMouseVelocity(hwnd, { x, y });
                if (time != 0) {
                    g_dyAutoScroll = MulDiv(distance, tmTimer, time);
                } else {
                    g_dyAutoScroll = 0;
                }
                if (g_dyAutoScroll > -g_cyLine && g_dyAutoScroll < g_cyLine) {
                    g_dyAutoScroll = direction * g_cyLine;
                }
                SetTimer(hwnd, IDT_AUTOSCROLL, tmTimer, OnAutoScroll);
            }
        } else {
            if (!DetectAutoScroll({ x, y })) {
                CancelAutoScroll(hwnd);
            }
        }
    }
}
```

When we are dragging and autoscroll is not yet active (`g_dyAutoScroll` is zero), then we see if autoscroll should start. If so, we calculate the mouse velocity and convert it to the same velocity relative to the autoscroll timer rate. If something goes wrong while calculating the mouse velocity, or if the resulting velocity is too tiny, then we use some minimum velocity.

On the other hand, if autoscroll is active, but the mouse is no longer in a place that can trigger autoscroll, then we cancel autoscroll.

The last missing piece is calculating the mouse velocity.

```
std::pair<int, DWORDgt;
GetMouseVelocity(HWND hwnd, POINT pt)
{
    ClientToScreen(hwnd, &pt);
    MOUSEMOVEPOINT lastPoint{ LOWORD(pt.x), LOWORD(pt.y),
                              (DWORD)GetMessageTime(), 0 };
    MOUSEMOVEPOINT recentPoints[3];
    int count = GetMouseMovePointsEx(sizeof(lastPoint), &lastPoint,
                    recentPoints, ARRAYSIZE(recentPoints),
                    GMMP_USE_DISPLAY_POINTS);
    if (count >= 3) {
        return { (short)recentPoints[0].y - (short)recentPoints[2].y,
                  recentPoints[0].time - recentPoints[2].time };
    }
    return { 0, 0 };
}
```

We take the mouse position and ask `GetMouseMovePoints` to look up the two points that led to the current mouse position (a total of three points). If successful, then we calculate the average mouse velocity over those last three points. Otherwise, we return `{ 0, 0 }` to say that we don't know what the mouse velocity is, and the caller should just use some default values.

And that's it. This version of autoscroll uses the exit velocity of the mouse to decide how fast to scroll, and scrolling continues at that speed until the mouse moves back into the client area. This lets you control the speed of scrolling even if the window is right against the edge of the screen: To scroll faster, just slam the mouse into the edge of the screen harder.

But you know what's missing?

Scrolling faster by wiggling the mouse.

Recall that scrolling faster by wiggling the mouse is a bug that turned into a feature because people discovered it and began to rely on it. We'll bring that back next time.

[1] The `GetMouseMovePointsEx` function didn't exist in Windows 95. To get this algorithm to work on Windows 95, you would have to emulate the function yourself by recording mouse position history manually.

Raymond Chen

Follow