Building on our program that draws content at a fixed screen position regardless of window position

devblogs.microsoft.com/oldnewthing/20130701-00

July 1, 2013

Raymond Chen

Today's Little Program uses the technique we saw last week of drawing content at a fixed screen position, regardless of window position, but adds a little physics to it.

Start with our scratch program and make these changes:

```
#include <math.h> // physics requires math (go figure)
#define Omega 2.0f
class Motion
{
public:
 Motion() : x0(0.0f), v0(0.0f) { RecalcCurve(); }
 void ShiftOrigin(double dx)
 {
  Tick();
  v0 = v;
  x0 = x + dx;
  RecalcCurve();
 }
 double Pos() { return x; }
 bool Moving() { return fabs(x) >= 0.5f || fabs(v) >= 1.0f; }
 void Tick() {
  t = (GetTickCount() - tm0) / 1000.0f;
  double ewt = exp(-Omega * t);
  double abt = A + B * t;
  x = abt * ewt;
  v = (-Omega * abt + B) * ewt;
 }
private:
 void RecalcCurve() {
  A = x0;
  B = v0 + Omega * x0;
  tm0 = GetTickCount();
 }
public:
 DWORD tm0;
 double x0, v0, A, B, t, x, v;
};
```

The `Motion` class simulates damped motion. Ask a physicist how it works.

```
Motion g_mX;  // motion in x-direction
Motion g_mY;  // motion in y-direction
POINT g_ptRest; // desired rest point
POINT CalcRestPoint(HWND hwnd)
{
    RECT rc;
    GetClientRect(hwnd, &rc);
    MapWindowRect(hwnd, HWND_DESKTOP, &rc);
    POINT pt = { rc.left + (rc.right - rc.left) / 2,
                 rc.top + (rc.bottom - rc.top) / 2 };
    return pt;
}
```

The rest point is the center of the window.

```
void ScheduleFrame(HWND hwnd)
{
    InvalidateRect(hwnd, 0, TRUE);
    KillTimer(hwnd, 1);
}
VOID CALLBACK InvalidateMe(HWND hwnd, UINT, UINT_PTR, DWORD)
{
    ScheduleFrame(hwnd);
}
```

To schedule the painting of a new frame, we invalidate our window and cancel any outstanding animation timer (because the timer is no longer needed now that a paint has been scheduled).

```
void OnWindowPosChanged(HWND hwnd, LPWINDOWPOS lpwpos)
{
    if (IsWindowVisible(hwnd)) {
        POINT ptRest = CalcRestPoint(hwnd);
        if (ptRest.x != g_ptRest.x ||
            ptRest.y != g_ptRest.y) {
          g_mX.ShiftOrigin(g_ptRest.x - ptRest.x);
          g_mY.ShiftOrigin(g_ptRest.y - ptRest.y);
          ScheduleFrame(hwnd);
        }
        g_ptRest = ptRest;
    }
}
    HANDLE_MSG(hwnd, WM_WINDOWPOSCHANGED, OnWindowPosChanged);
```

If the window changes its rest point while it is vislble, then move the origin of the motion variables and schedule a new frame of animation.

Okay, here’s the fun part: Drawing the moving circle.

```
void
PaintContent(HWND hwnd, PAINTSTRUCT *pps)
{
 RECT rc;
 g_mX.Tick();
 g_mY.Tick();
 POINT ptOrigin = { 0, 0 };
 ClientToScreen(hwnd, &ptOrigin);
 POINT ptOrg;
 SetWindowOrgEx(pps->hdc, ptOrigin.x, ptOrigin.y, &ptOrg);
 int x = g_ptRest.x + static_cast<int>(floor(g_mX.Pos() + 0.5f));
 int y = g_ptRest.y + static_cast<int>(floor(g_mY.Pos() + 0.5f));
 Ellipse(pps->hdc, x - 20, y - 20, x + 20, y + 20);
 SetWindowOrgEx(pps->hdc, ptOrg.x, ptOrg.y, nullptr);
 if (g_mX.Moving() || g_mY.Moving()) {
  SetTimer(hwnd, 1, 30, InvalidateMe);
 }
}
```

We tick the motion variables to get their current locations, then tinker with our window origin because we're going to be drawing based on screen coordinates. We then draw a circle at the current animated position, and if the circle is still moving, we schedule a timer to draw the next frame.

Finally, we initialize our rest point before we show the window, so that the circle starts out at rest.

```
        g_ptRest = CalcRestPoint(hwnd);
        ShowWindow(hwnd, nShowCmd);
```

And that's it. Run the program and move it around. The circle will seek the center of the window, wherever it is.

(For extra credit, you can also add

```
UINT OnNCHitTest(HWND hwnd, int x, int y)
{
    UINT ht = FORWARD_WM_NCHITTEST(hwnd, x, y, DefWindowProc);
    if (ht == HTCLIENT) ht = HTCAPTION;
    return ht;
}
    HANDLE_MSG(hwnd, WM_NCHITTEST, OnNCHitTest);
```

so that the window can be dragged by its client area.)

Raymond Chen

Follow