Playing with the Windows Animation Manager: Moving lots of stuff around

devblogs.microsoft.com/oldnewthing/20130315-00

March 15, 2013

Raymond Chen

We saw last time a sample program that moved a circle around. Today I'll try to build the classic demo of animating a lot of objects in a list.

This isn't the prettiest code, but I wanted to make as few changes as possible. Start with the Timer-Driven Animation, and make these changes to the `MainWindow.h` header file.

```
struct Item
{
    IUIAnimationVariable *m_pAnimationVariableX;
    IUIAnimationVariable *m_pAnimationVariableY;
    Gdiplus::Color m_color;
};


class MainWindow
{
    …


    // ~~HRESULT ChangeColor(~~
    //     ~~DOUBLE red,~~
    //     ~~DOUBLE green,~~
    //     ~~DOUBLE blue~~
    //     ~~);~~
    HRESULT ChangePos();


    …
private:


    static const int ItemCount = 100;
    static const int ItemWidth = 40;
    static const int ItemHeight = 40;
    static int XFromIndex(int index)
    {
        return (index % 10) * 50;
    }
    static int YFromIndex(int index)
    {
        return (index / 10) * 50;
    }


    …


    // ~~IUIAnimationVariable *m_pAnimationVariableRed;~~
    // ~~IUIAnimationVariable *m_pAnimationVariableGreen;~~
    // ~~IUIAnimationVariable *m_pAnimationVariableBlue;~~
    Item m_Items[ItemCount];
};
```

From the changes in the header file, I think you see where this is going. Instead of just having one item on the screen, I'm going to put a hundred.

Here are the changes to `MainWindow.cpp`. First, we need to null out our pointers at construction and clean them up at destruction. (This sample program does not use smart pointers, so I won't either.)

```
CMainWindow::CMainWindow() :
    m_hwnd(NULL),
    m_pAnimationManager(NULL),
    m_pAnimationTimer(NULL),
    m_pTransitionLibrary(NULL)// ~~,~~
    // ~~m_pAnimationVariableRed(NULL),~~
    // ~~m_pAnimationVariableGreen(NULL),~~
    // ~~m_pAnimationVariableBlue(NULL)~~
{
    for (int i = 0; i < ItemCount; i++)
    {
        m_Items[i].m_pAnimationVariableX = NULL;
        m_Items[i].m_pAnimationVariableY = NULL;
    }
}


CMainWindow::~CMainWindow()
{
    // Animated Variables
    // ~~SafeRelease(&m_pAnimationVariableRed);~~
    // ~~SafeRelease(&m_pAnimationVariableGreen);~~
    // ~~SafeRelease(&m_pAnimationVariableBlue);~~
    for (int i = 0; i < ItemCount; i++)
    {
        SafeRelease(&m_Items[i].m_pAnimationVariableX);
        SafeRelease(&m_Items[i].m_pAnimationVariableY);
    }


    …
}
```

Next we get rid of the initial animation.

```
HRESULT CMainWindow::Initialize(
    HINSTANCE hInstance
    )
{
    …
                // ~~Fade in with Red~~
                // ~~hr = ChangeColor(COLOR_MAX, COLOR_MIN, COLOR_MIN);~~
    …
}
```

As you might expect, the `CreateAnimationVariables` method changed completely, since it now has to create the variables for each item. But the basic idea is the same: Create each variable with the appropriate initial value. (It's also a lot shorter!)

```
HRESULT CMainWindow::CreateAnimationVariables()
{
    HRESULT hr = S_OK;


    for (int i = 0; SUCCEEDED(hr) && i < ItemCount; i++)
    {
        m_Items[i].m_color = Color(
            static_cast<BYTE>(RandomFromRange(COLOR_MIN, COLOR_MAX)),
            static_cast<BYTE>(RandomFromRange(COLOR_MIN, COLOR_MAX)),
            static_cast<BYTE>(RandomFromRange(COLOR_MIN, COLOR_MAX))
        );
        hr = m_pAnimationManager->CreateAnimationVariable(
            XFromIndex(i),
            &m_Items[i].m_pAnimationVariableX
        );
        if (SUCCEEDED(hr))
        {
            hr = m_pAnimationManager->CreateAnimationVariable(
            YFromIndex(i),
                &m_Items[i].m_pAnimationVariableY
                );
        }
    }


    return hr;
}
```

The `DrawBackground` method is becoming increasingly inaccurately-named, since in addition to drawing the background, we also draw the foreground!

```
HRESULT CMainWindow::DrawBackground(
    Graphics &graphics,
    const RectF &rectPaint
    )
{
    SolidBrush brushBackground(Color(255, 255, 255));
    HRESULT hr = HrFromStatus(graphics.FillRectangle(
            &brushBackground,
            rectPaint
            ));


    for (int i = 0; SUCCEEDED(hr) && i < ItemCount; i++)
    {
        INT32 x;
        hr = m_Items[i].m_pAnimationVariableX->GetIntegerValue(
            &x
            );
        if (SUCCEEDED(hr))
        {
            INT32 y;
            hr = m_Items[i].m_pAnimationVariableY->GetIntegerValue(
                &y
                );
            if (SUCCEEDED(hr))
            {
                SolidBrush brush(m_Items[i].m_color);
                RectF rectItem(
                    static_cast<REAL>(x),
                    static_cast<REAL>(y),
                    static_cast<REAL>(ItemWidth),
                    static_cast<REAL>(ItemHeight));
                hr = HrFromStatus(graphics.FillRectangle(
                    &brush,
                    rectItem
                    ));
            }
        }
    }


    return hr;
}
```

We change what happens when you click the left mouse button. Instead of changing the color, we shuffle the items randomly.

```
HRESULT CMainWindow::OnLButtonDown()
{
    HRESULT hr = ChangePos();


    return hr;
}
```

And now the money function: Shuffling the items and animating them to their new locations.

```
HRESULT CMainWindow::ChangePos()
{
    const UI_ANIMATION_SECONDS DURATION = 0.5;
    const DOUBLE ACCELERATION_RATIO = 0.5;
    const DOUBLE DECELERATION_RATIO = 0.5;


    // Assign final locations randomly
    int Destination[ItemCount];
    Destination[0] = 0;
    for (int i = 1; i < ItemCount; i++)
    {
        int j = rand() % (i + 1);
        Destination[i] = Destination[j];
        Destination[j] = i;
    }


    // Create a storyboard


    IUIAnimationStoryboard *pStoryboard = NULL;
    HRESULT hr = m_pAnimationManager->CreateStoryboard(
        &pStoryboard
        );
    if (SUCCEEDED(hr))
    {
        for (int i = 0; SUCCEEDED(hr) && i < ItemCount; i++)
        {
            // Create transitions for the position animation variables


            IUIAnimationTransition *pTransitionX;
            hr = m_pTransitionLibrary->CreateAccelerateDecelerateTransition(
                DURATION,
                XFromIndex(Destination[i]),
                ACCELERATION_RATIO,
                DECELERATION_RATIO,
                &pTransitionX
                );


            if (SUCCEEDED(hr))
            {
                IUIAnimationTransition *pTransitionY;
                hr = m_pTransitionLibrary->CreateAccelerateDecelerateTransition(
                    DURATION,
                    YFromIndex(Destination[i]),
                    ACCELERATION_RATIO,
                    DECELERATION_RATIO,
                    &pTransitionY
                );
```

```
        // Delete “blue” transition


        if (SUCCEEDED(hr))
        {
            // Add transitions to the storyboard


            hr = pStoryboard->AddTransition(
                m_Items[i].m_pAnimationVariableX,
                pTransitionX
                );
            if (SUCCEEDED(hr))
            {
                hr = pStoryboard->AddTransition(
                    m_Items[i].m_pAnimationVariableY,
                    pTransitionY
                    );
                // Delete “blue” transition
                // Move “Schedule” out of the loop
            }


            pTransitionY->Release();
        }


        pTransitionX->Release();
    }
}


// Scheduling code moved outside the loop
if (SUCCEEDED(hr))
{
    // Get the current time and schedule the storyboard for play


    UI_ANIMATION_SECONDS secondsNow;
    hr = m_pAnimationTimer->GetTime(
        &secondsNow
        );
    if (SUCCEEDED(hr))
    {
        hr = pStoryboard->Schedule(
            secondsNow
            );
    }
}
```

```
        pStoryboard->Release();
    }


    return hr;
}
```

It looked like a lot of code, but really wasn't. The only real change was to add the shuffling code and to put a loop around the code that generates the transitions and adds them to the storyboard.

And there you have it, a program that smoothly animates 100 items each time you click on the window. For me, the fun thing to do is to just click repeatedly on the window and watch the items swirl around like a swarm of insects.

Raymond Chen

Follow