C# 系统应用之鼠标模拟技术及自动操作鼠标

原创 Eastmount 最后发布于2014-10-13 20:04:54 阅读数 8959 ☆ 收藏 展开

游戏程序的操作不外乎两种——键盘输入控制和鼠标输入控制,几乎所有游戏中都使用鼠标来改变角色的位置和方向,本文主要是讲述如何使用C#调用Windows API函数实现鼠标模拟操作的功能.首先通过结合FindWindow和FindWindowEx寻找到窗体的按钮,在通过SetCursorPos或mouse_event函数操作鼠标,同时涉及到通过spy++工具获取窗体消息的信息.

一. Windows API函数介绍

.NET没有提供改变鼠标指针位置、模拟单机操作的函数,但是可以通过调用Windows API函数实现.

```
[DllImport("user32.dll")]
static extern bool SetCursorPos(int X,int Y);
```

该函数用于设置鼠标的位置,其中X和Y是相对于屏幕左上角的绝对位置.

```
[DllImport("user32.dll")]
static extern void mouse_event(MouseEventFlag flags,int dx,int dy,uint data,UIntPtr extraInfo);
```

该函数不仅可以设置鼠标指针绝对位置,而且可以以相对坐标来设置位置.

其中flags标志位集,指定点击按钮和鼠标动作的多种情况.dx指鼠标沿x轴绝对位置或上次鼠标事件位置产生以来移动的数量.dy指沿y轴的绝对位置或从上次鼠标事件以来移动的数量.data如果flags为MOUSE_WHEEL则该值指鼠标轮移动的数量(否则为0),正值向前转动.extraInfo指定与鼠标事件相关的附加32位值.

```
[DllImport("user32.dll")]
static extern IntPtr FindWindow(string strClass, string strWindow);
```

该函数根据类名和窗口名来得到窗口句柄,但是这个函数不能查找子窗口,也不区分大小写.如果要从一个窗口的子窗口查找需要使用FIndWindowEX函数.

```
[DllImport("user32.dll")]
static extern IntPtr FindWindowEx(IntPtr hwndParent, IntPtr hwndChildAfter,
        string strClass, string strWindow);
```

该函数获取一个窗口的句柄,该窗口的类名和窗口名与给定的字符串相匹配,该函数查找子窗口时从排在给定的子窗口后面的下一个子窗口开始.其中参数hwnParent为要查找子窗口的父窗口句柄,若该值为NULL则函数以桌面窗口为父窗口,查找桌面窗口的所有子窗口.

hwndChildAfter子窗口句柄,查找从在Z序中的下一个子窗口开始,子窗口必须为hwnParent直接子窗口而非后代窗口,若hwnChildAfter为NULL,查找从父窗口的第一个子窗口开始.

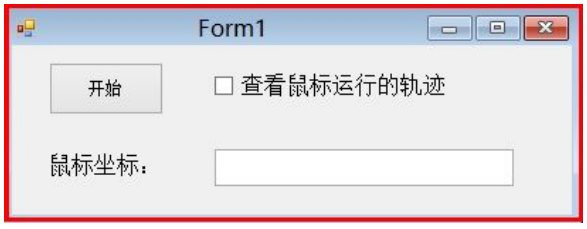
strClass指向一个指定类名的空结束字符串或一个标识类名字符串的成员的指针.

strWindow指向一个指定窗口名(窗口标题)的空结束字符串.若为NULL则所有窗体全匹配.

返回值:如果函数成功,返回值为具有指定类名和窗口名的窗口句柄,如果函数失败,返回值为NULL.

二. 鼠标自动点击按钮和查看鼠标运行轨迹

首先创建一个C#工程,设计的窗体如下图所示,同时添加Timer时间器控件:



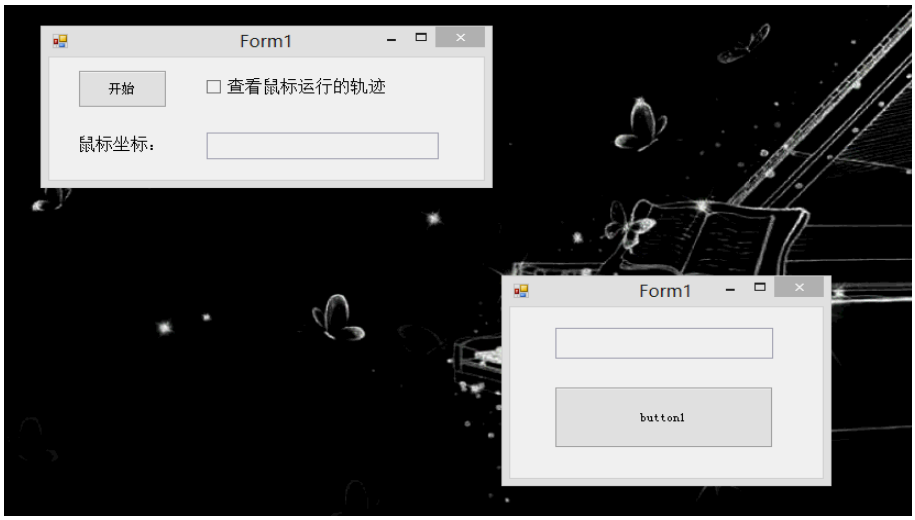
然后添加的如下代码,即可实现鼠标模拟技术及自动操作鼠标:

```
using System; using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
//引用新命名空间
using System.Runtime.InteropServices;  //StructLayout

namespace MouseAction
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        //结构体布局 本机位置
        [StructLayout(LayoutKind.Sequential)]
        struct NativeRECT
        {
            public int left;
            public int top;
            public int right;
            public int bottom;
        }

        //将枚举作为位域处理
        [Flags]
        enum MouseEventFlag : uint //设置鼠标动作的键值
        {
            Move = 0x0001,               //发生移动
            LeftDown = 0x0002,           //鼠标按下左键
            LeftUp = 0x0004,             //鼠标松开左键
            RightDown = 0x0008,          //鼠标按下右键
            RightUp = 0x0010,            //鼠标松开右键
            MiddleDown = 0x0020,         //鼠标按下中键
            MiddleUp = 0x0040,           //鼠标松开中键
            XDown = 0x0080,
            XUp = 0x0100,
            Wheel = 0x0800,              //鼠标轮被移动
            VirtualDesk = 0x4000,        //虚拟桌面
            Absolute = 0x8000
        }

        //设置鼠标位置
        [DllImport("user32.dll")]
        static extern bool SetCursorPos(int X, int Y);

        //设置鼠标按键和动作
        [DllImport("user32.dll")]
        static extern void mouse_event(MouseEventFlag flags, int dx, int dy,
            uint data, UIntPtr extraInfo); //UIntPtr指针多句柄类型

        [DllImport("user32.dll")]
        static extern IntPtr FindWindow(string strClass, string strWindow);

        //该函数获取一个窗口句柄,该窗口雷鸣和窗口名与给定字符串匹配 hwnParent=Null从桌面窗口查找
```

```
[DllImport("user32.dll")]
static extern IntPtr FindWindowEx(IntPtr hwndParent, IntPtr hwndChildAfter,
    string strClass, string strWindow);

[DllImport("user32.dll")]
static extern bool GetWindowRect(HandleRef hwnd, out NativeRECT rect);

//定义变量
const int AnimationCount = 80;
private Point endPosition;
private int count;

private void button1_Click(object sender, EventArgs e)
{
    NativeRECT rect;
    //获取主窗体句柄
    IntPtr ptrTaskbar = FindWindow("WindowsForms10.Window.8.app.0.2bf8098_r11_ad1", null);
    if (ptrTaskbar == IntPtr.Zero)
    {
        MessageBox.Show("No windows found!");
        return;
    }
    //获取窗体中"button1"按钮
    IntPtr ptrStartBtn = FindWindowEx(ptrTaskbar, IntPtr.Zero, null, "button1");
    if (ptrStartBtn == IntPtr.Zero)
    {
        MessageBox.Show("No button found!");
        return;
    }
    //获取窗体大小
    GetWindowRect(new HandleRef(this, ptrStartBtn), out rect);
    endPosition.X = (rect.left + rect.right) / 2;
    endPosition.Y = (rect.top + rect.bottom) / 2;
    //判断点击按钮
    if (checkBox1.Checked)
    {
        //选择"查看鼠标运行的轨迹"
        this.count = AnimationCount;
        movementTimer.Start();
    }
    else
    {
        SetCursorPos(endPosition.X, endPosition.Y);
        mouse_event(MouseEventFlag.LeftDown, 0, 0, 0, UIntPtr.Zero);
        mouse_event(MouseEventFlag.LeftUp, 0, 0, 0, UIntPtr.Zero);
        textBox1.Text = String.Format("{0},{1}", MousePosition.X, MousePosition.Y);
    }

}

//Tick:定时器,每当经过多少时间发生函数
private void movementTimer_Tick(object sender, EventArgs e)
{
    int stepx = (endPosition.X - MousePosition.X) / count;
    int stepy = (endPosition.Y - MousePosition.Y) / count;
    count--;
    if (count == 0)
    {
        movementTimer.Stop();
        mouse_event(MouseEventFlag.LeftDown, 0, 0, 0, UIntPtr.Zero);
        mouse_event(MouseEventFlag.LeftUp, 0, 0, 0, UIntPtr.Zero);
    }
```

```
            textBox1.Text = String.Format("{0},{1}", MousePosition.X, MousePosition.Y);
            mouse_event(MouseEventFlag.Move, stepx, stepy, 0, UIntPtr.Zero);
                                                                                    }
    }
}
```

同时自定义一个对话框,增加一个button按钮,其运行结果如下图所示:



可以看到当运行程序勾选"查看鼠标运行的轨迹"并点击"开始"按钮后,会通过FindWindow和FindWindowEx函数查找窗体"Form1"的"button1"按钮,并通过mouse_event移动鼠标和点击鼠标.其中函数原型为:

```
IntPtr FindWindowEx(
    IntPtr hwndParent,     // handle to parent window [父窗体句柄]
    IntPtr hwndChildAfter, // handle to child window [子窗体句柄]
    string strClass,       // class name [窗体类名]
    string strWindow       // window name [窗体名]
);
```

但是怎样找到窗体类名和按钮的类名呢?由于初学,很多窗体我都没有实现如QQ,它需要用到一个叫spy++的工具.

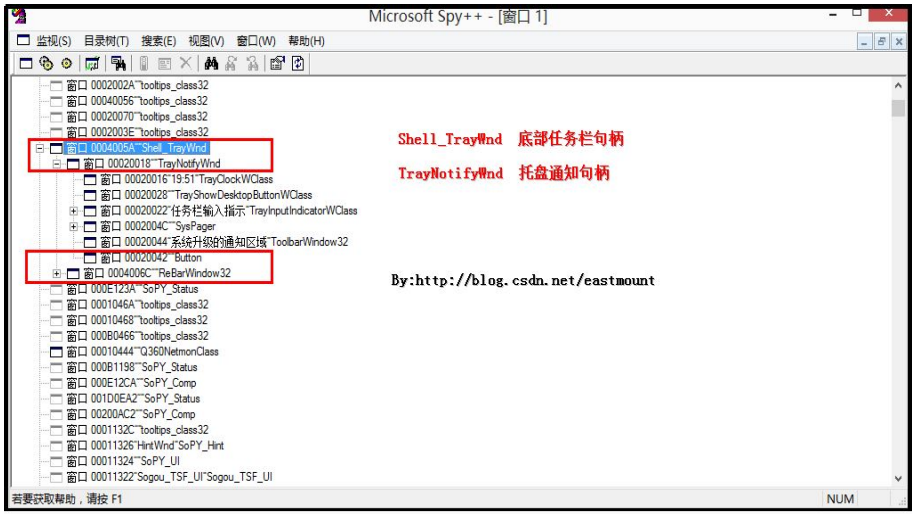
PS:第一次制作gif格式动态图片,参照博客 http://blog.csdn.net/tangcheng_ok/article/details/8246792

三. 使用SPY++工具获取窗体信息

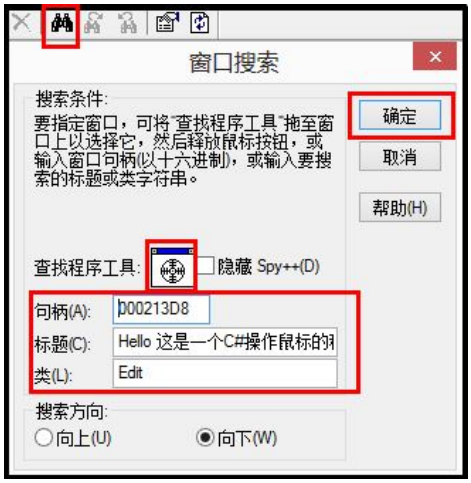
如果修改代码为:

```
//获取任务栏句柄
IntPtr ptrTaskbar = FindWindow("Shell_TrayWnd",null);
//托盘通知句柄
IntPtr ptrStartBtn = FindWindowEx(ptrTaskbar, IntPtr.Zero, "TrayNotifyWnd", null);
```

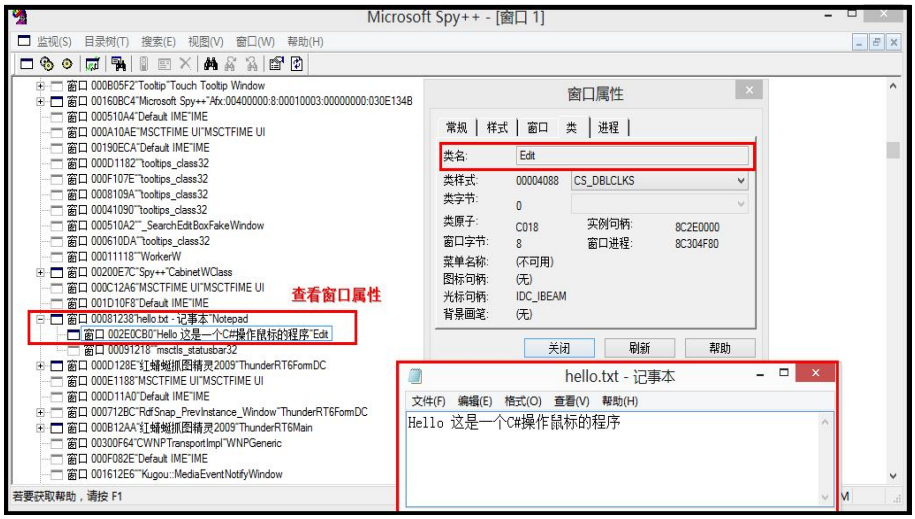
可以获取电脑底部任务栏的托盘通知句柄,其中通过Spy++工具(VS中"工具"中自带)查找如下图所示:



同样,我通过spy++工具获取txt句柄,首先打开spy++工具,同时点击"查找窗口"按钮(望远镜),再点击"查找程序工具"中按钮拖拽至要查看的窗体中,点击"确定"按钮.



这样就会显示这个txt的信息,同时可以右击"属性"显示窗体的类名、窗体题目、句柄等信息.



最后通过下面代码可以获取hello.txt的句柄:

```
//获取记事本句柄
IntPtr ptrTaskbar = FindWindow("Notepad", null);
IntPtr ptrStartBtn = FindWindowEx(ptrTaskbar, IntPtr.Zero, "Edit", null);
```

再通过mouse_event操作鼠标,同时可以通过SendMessage将指定的消息发送到一个或多个窗口,PostMessage将一个消息寄送到一个线程的消息队列后

就立即返回.实现消息传递等功能,学习ing~

四. 总结

该篇文章主要讲述C#如何操作鼠标的事件,在制作游戏外挂或自动运行程序时非常实用,但遗憾的是在上面通过窗体名称"Form1"获取窗体时总是失败,需要通过spy++获取它的类名来实现.Why?同时如果想学习键盘模拟技术的可以研究SetWindowsHookEx(安装钩子)、CallNextHookEx(下一个钩子)、UnhookWindowsHookEx(卸载钩子)和鼠标Hook实现很多技术.

希望文章对大家有所帮助,如果有错误或不足之处,请见谅~

(By:Eastmount 2014年10月13日 晚上8点 http://blog.csdn.net/eastmount/)

参考资料-在线笔记:

本文主要参考书籍《C#网络变成高级篇之网页游戏辅助程序设计》张慧斌 王小峰著

1.C#获取QQ聊天输入框中内容 http://www.csharpwin.com/csharpspace/9133r5654.shtml

2.C#查找窗口,FindWindow用法(By-LYBwwp)http://blog.csdn.net/lybwwp/article/details/8168553

3.FindWindowEx用法(By-coolszy) http://blog.csdn.net/coolszy/article/details/5523784

4.C# 隐藏任务栏开始按钮关闭shell(By-sshhbb)http://blog.csdn.net/sshhbb/article/details/6605976

5.任务栏句柄 http://blog.csdn.net/wangjieest/article/details/6943241

6.C#如何在外部程序的密码框内自动输入密码 http://biancheng.dnbcw.info/c/117849.html

7.C#实现对外部程序的调用操作 http://www.blue1000.com/bkhtml/c17/2012-11/70993.htm

8.百度知道 C# API函数FindWindowEx返回子窗体的值为零

9.百度知道 用C#操作API实现填写桌面窗体内的textbox并点击窗体按钮