C#键盘外挂

网上有很多外挂制作的教程，大多是讲针对大型网络游戏的，主要包含一些抓包、反汇编、C++的知识综合。事实也如此，常见的外挂都是使用 VC++写的，从来没有过 C#或者其他.NET 语言编写的外挂。

作为微软.NET 技术的忠实粉丝，这难免是一种遗憾。不过不要紧，下面流牛木马就教大家两招，包教包会，免收学费。 ：)

其实作为游戏外挂来说，主要就是三个功能：模拟键盘操作、模拟鼠标操作、修改内存数据。修改内存数据比较难，但模拟鼠标键盘的操作却很简单。很多流行游戏的外挂，都可以只通过模拟鼠标键盘来实现，例如：劲舞团、QQ 音速、连连看、各类网页游戏，以及各类大型网游中的自动打怪、自动吃药等等。

Warcraft Ⅲ，学名魔兽争霸之冰封王座，俗称魔兽，简称 war3，在最近六七年风靡全球。最近两年，war3 在中国又掀起了玩 DOTA 的新高潮。

本文制作 DOTA 游戏中的显血、改键外挂为例，简单地介绍如何使用 C#语言制作游戏外挂。

最终界面如下：

C#简单游戏外挂制作(以 Warcraft Ⅲ为例)

本示例包含两个功能：显血；将 Q 键改为小键盘的 7 键。玩 war3 的同学都知道，这两个功能对于 war3(尤其是 DOTA)相当重要。

首先简单介绍一下，外挂程序模拟键盘的原理。

外挂程序与游戏程序是两个不同的进程。外挂程序使用 Windows 提供的 API 找到游戏程序的进程，并设置键盘钩子(什么叫做钩子？你不知道，但百度知道。)设置完钩子后，我们再监控游戏进程中用户的按键，并根据用户需求进行处理，完成某些模拟键盘动作。

了解了这个过程之后，我们就可以开始整理思路了。完成外挂一共需要以下四个步骤：

一、声明 Windows API 中的函数和常量

```
//键盘 Hook 结构函数
    [StructLayout(LayoutKind.Sequential)]
    public  class  KeyBoardHookStruct
    {
        public  int  vkCode;
        public  int  scanCode;
```

```
        public int flags;
        public int time;
        public int dwExtraInfo;
    }
    #region DllImport
    //设置钩子
      [DllImport("user32.dll", CharSet = CharSet.Auto, CallingConvention =
CallingConvention.StdCall)]
     public static extern int SetWindowsHookEx(int idHook, HookProc lpfn,
IntPtr hInstance, int threadId);
      [DllImport("user32.dll", CharSet = CharSet.Auto, CallingConvention =
CallingConvention.StdCall)]
    //抽掉钩子
    public static extern bool UnhookWindowsHookEx(int idHook);
      [DllImport("user32.dll", CharSet = CharSet.Auto, CallingConvention =
CallingConvention.StdCall)]
    //调用下一个钩子
     public static extern int CallNextHookEx(int idHook, int nCode, IntPtr
wParam, IntPtr lParam);
    //取得模块句柄
      [DllImport("kernel32.dll", CharSet = CharSet.Auto, CallingConvention =
CallingConvention.StdCall)]
    private static extern IntPtr GetModuleHandle(string lpModuleName);

    //寻找目标进程窗口

    [DllImport("USER32.DLL")]
    public static extern IntPtr FindWindow(string lpClassName,
        string lpWindowName);
      //设置进程窗口到最前

    [DllImport("USER32.DLL")]
    public static extern bool SetForegroundWindow(IntPtr hWnd);
    //模拟键盘事件

    [DllImport("User32.dll")]
     public static extern void keybd_event(Byte bVk, Byte bScan, Int32
dwFlags, Int32 dwExtraInfo);

//释放按键的常量
  private const int KEYEVENTF_KEYUP =2;
```

本例所使用的函数比较少，它们都在系统的 USER32.dll 里，包括：设置和取消钩子、调用下一个钩子、导入进程、模拟键盘等等。我们依次导入它们。

这些函数的命名规范合理，几乎只根据函数名就能知道其功能。

如果读者对于其中的某些函数不熟悉，请自行搜索 MSDN。

二、使用 Windows API 设置钩子

有了以上 windows API 函数的声明，下一步就是设置钩子了。

寥寥两行代码，但包含了相当丰富的内容。

```
//委托
public delegate int HookProc(int nCode, IntPtr wParam, IntPtr lParam);



public void Hook_Start()
    {
        // 安装键盘钩子
        if (hHook == 0)
        {
            KeyBoardHookProcedure = new HookProc(KeyBoardHookProc);

            hHook = SetWindowsHookEx(WH_KEYBOARD_LL, KeyBoardHookProcedure,
GetModuleHandle(Process.GetCurrentProcess().MainModule.ModuleName), 0);

        }
    }
```

先介绍一下设置钩子的明星函数：SetWindowsHookEx 。它的参数说明如下。

```
SetWindowsHookEx(

idHook: Integer;        {钩子类型}

lpfn: TFNHookProc; {函数指针}

hmod: HINST;                {包含钩子函数的模块(EXE、DLL)句柄; 一般是 HInstance; 如果是当前线程这里可以是 0}

dwThreadId: DWORD    {关联的线程; 可用 GetCurrentThreadId 获取当前线程; 0 表示是系统级钩子}
```

): HHOOK;　　{返回钩子的句柄; 0 表示失败}

请注意 lpfn 这个参数。上面的解释是“函数指针”。在 C#中，是不能直接使用指针的，更不要说函数指针了。我们可以采用 C#中的委托（delegate）来实现函数指针的功能。

于是乎，在上面的代码中，我们定义了一个处理键盘消息函数的委托 KeyBoardHookProcedure = new HookProc(KeyBoardHookProc)，并将它作为参数传入 SetWindowsHookEx 内。KeyBoardHookProc 就是被委托的具体函数。

三、监控用户操作

设置好钩子后，我们可以在被委托的函数中写入监控用户操作与模拟键盘的代码。

```
public static int KeyBoardHookProc(int nCode, IntPtr wParam, IntPtr lParam)
        {
        //监控用户键盘输入

                                        KeyBoardHookStruct input =
(KeyBoardHookStruct)Marshal.PtrToStructure(lParam, typeof(KeyBoardHookStruct));

            //截获 Home 键

            if (input.vkCode == (int)Keys.Home)
            {
              //此处写入其他操作逻辑

            }

            //  继续执行下一个钩子程序
            return CallNextHookEx(hHook, nCode, wParam, lParam);

        }
```

四、根据用户需要模拟键盘操作

显血功能：玩 war3 的都知道，war3 自带的显血快捷键有 3 个。Alt 键是显示所有单位生命，[键显示友方单位生命，] 键显示地方单位生命。外挂需要做的事情仅仅是模拟一直按着某个键不松手而已。由于 Alt 键与其他很多键构成组合键，故我们不能模拟长按 Alt，否则会影响正常游戏。我们的解决方案应该是模拟长按 [键和] 键。代码如下：

```
            //获得魔兽程序的句柄
              IntPtr wcHandle = FindWindow(null, "Warcraft III");

              //如果钩子有效
```

```
if (wcHandle != IntPtr.Zero)
{
    //设置游戏窗口到最前
    SetForegroundWindow(wcHandle);

byte VK_NUM1 = 219;     //键盘上 [ 键的代码。按[可显示友方单位生命值。
  byte VK_NUM2 = 221;     // 键盘上] 键的代码。按]可显示敌方单位生命值。
  keybd_event(VK_NUM1, 0, 0, 0); //长按[
keybd_event(VK_NUM2, 0, 0, 0); //长按]

}
```

改键：小键盘（Numpad）上的快捷键很不方便按，所以很多玩家喜欢把小键盘上的键改到左边的字母键盘。玩 DOTA 的同学都知道，没有任何英雄的技能使用"Q”这个快捷键(召唤师有一种球是"Q"(不是技能))。于是我们把小键盘上的 7 键改到 Q 上，也不会造成任何冲突。方法也很简单：如果监控到用户按"Q”键，则像游戏进程发送小键盘上的"7"键。代码如下：

```
//如果用户按了 Q 键
  if (input.vkCode == (int)Keys.Q)
  {
      //获得魔兽程序的句柄
      IntPtr wcHandle = FindWindow(null, "Warcraft III");

      //如果钩子有效
      if (wcHandle != IntPtr.Zero)
      {
          //设置游戏窗口到最前
          SetForegroundWindow(wcHandle);
          byte VK_Q = (byte)Keys.NumPad7;
          keybd_event(VK_Q, 0, 0, 0);//按下小键盘 7
          keybd_event(VK_Q, 0, KEYEVENTF_KEYUP, 0); //松开小键盘 7
      }
      return 1;
  }
```

好了，到这里就把模拟键盘的外挂介绍完了。模拟鼠标与之非常类似，请用户自行揣摩。