实验一、窗体应用的消息传递

一、实验内容

分别利用WinForm和WPF实现两个窗体应用程序的消息传递。

二、实验原理

2.1设置消息的结构体

结构体设置如下：

```
    public struct COPYDATASTRUCT
    {
        public IntPtr dwData;
        public int cbData;
        [MarshalAs(UnmanagedType.LPStr)]
        public string lpData;
    }
```

并生成dll，在发送方和接收方的应用程序中引用。

2.2 发送方的消息传递

主要利用windows系统库User32.dll中的SendMessage（）函数来实现发送方的发送消息；同时设置消息常量为0x004A。首先通过系统库USer32.dll中的FindWindow（）函数找取接收方的文件句柄，若找到，则设置消息结构体，并使用SendMessage（）函数发送消息。

2.3 接收方的消息传递

2.3.1 对于WinForm

重载DefWndProc()函数，使用switch（）条件分支，来匹配接受的消息是否为上述发送的消息，若是，则类型转换得到发送的消息。

2.3.2 对于WPF

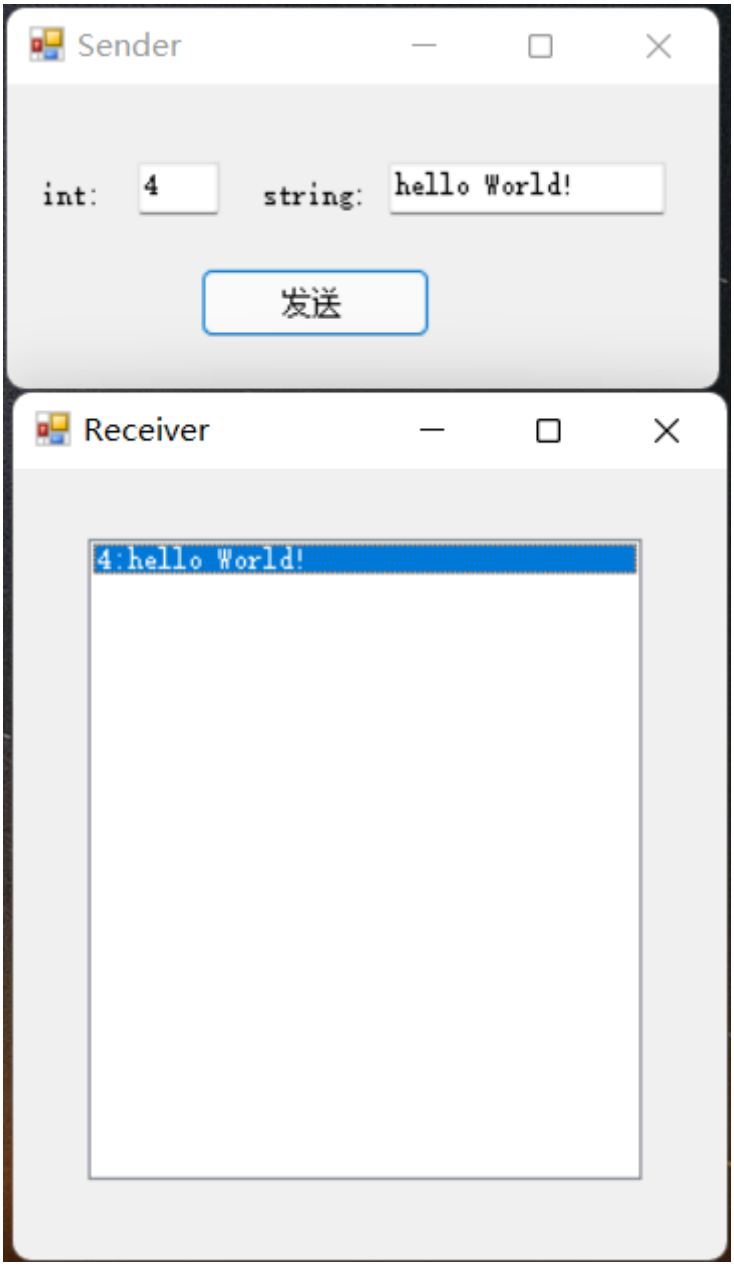
重载OnSourceInitialized（）函数，在其中绑定钩子函数。

设计钩子函数，代码如下：

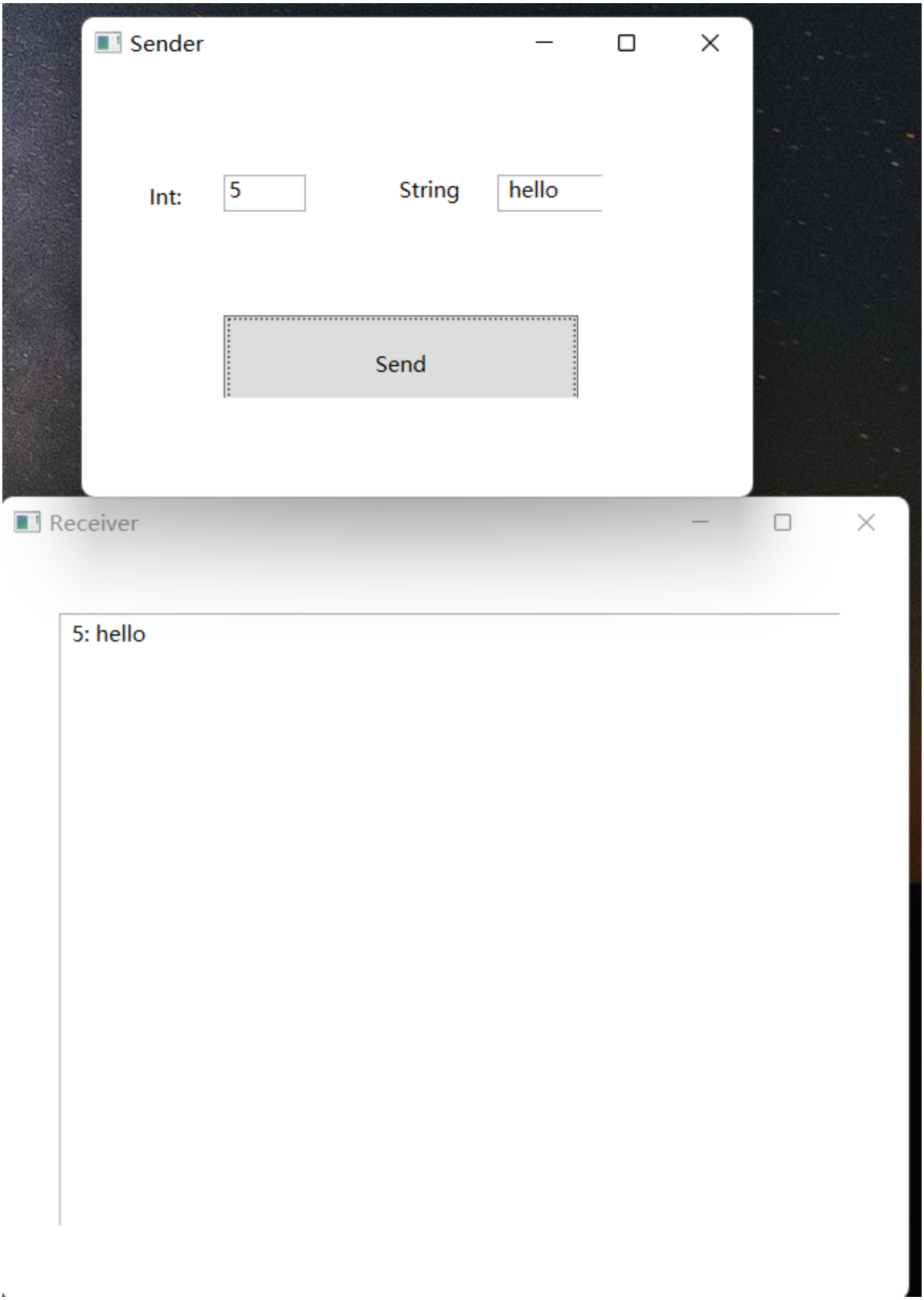
```
        IntPtr WndProc(IntPtr hwnd, int msg, IntPtr wParam, IntPtr lParam, ref
bool handled)
        {
            if (msg == WM_COPYDATA)
            {
                COPYDATASTRUCT cds =
(COPYDATASTRUCT)Marshal.PtrToStructure(lParam, typeof(COPYDATASTRUCT)); // 接收封
装的消息
                string strResult = cds.dwData.ToString() + ":" + cds.lpData;
                lsvMsgList.Items.Add(strResult);
            }
            return hwnd;
        }
```

三、实验结果

WinForm结果如下：



WPF如下：



实验二、事件的应用

一、实验内容

分别实验WinForm窗体和WPF窗体实现事件的定义、触发与处理。以火灾事件为例。

二、实验步骤

2.1 事件参数定义

重载EventArgs类，增加room和ferocity参数，代码如下：

```
        public class FireEventArgs : EventArgs
        {
            public FireEventArgs(string room, int ferocity)
            {
                this.room = room;
                this.ferocity = ferocity;
            }
            public string room;
            public int ferocity;
        }
```

2.2 事件的定义

在其中定义事件处理委托和事件，并且设计了一个激活事件的函数

```
        public class FireAlarm
        {
            public delegate void FireEventHandler(object sender, FireEventArgs
fe);
            public event FireEventHandler FireEvent;
            public void ActivateFireAlarm(string room, int ferocity)
            {
                FireEventArgs fireArgs = new FireEventArgs(room, ferocity);
                FireEvent(this, fireArgs);
            }
        }
```

2.3 事件处理类

主要核心是注册事件处理函数

```
fireAlarm.FireEvent += new FireAlarm.FireEventHandler(ExtinguishFire);//注册

            void ExtinguishFire(object sender, FireEventArgs fe)
            {
                OutStr.sw.WriteLine(" {0} 对象调用，灭火事件ExtinguishFire 函数.",
sender.ToString());
                parent.showComment(sender.ToString() + " 对象调用，灭火事件
ExtinguishFire 函数.");
                //根据火情状况，输出不同的信息.
                if (fe.ferocity < 2)
                {
                    OutStr.sw.WriteLine(" 火情发生在{0}，主人浇水后火情被扑灭了",
fe.room);
                    parent.showComment(" 火情发生在 " + fe.room + "，主人浇水后火情被
扑灭了");
                }
                else if (fe.ferocity < 5)
                {
                    OutStr.sw.WriteLine(" 主人正在使用灭火器处理{0} 火势.", fe.room);
```

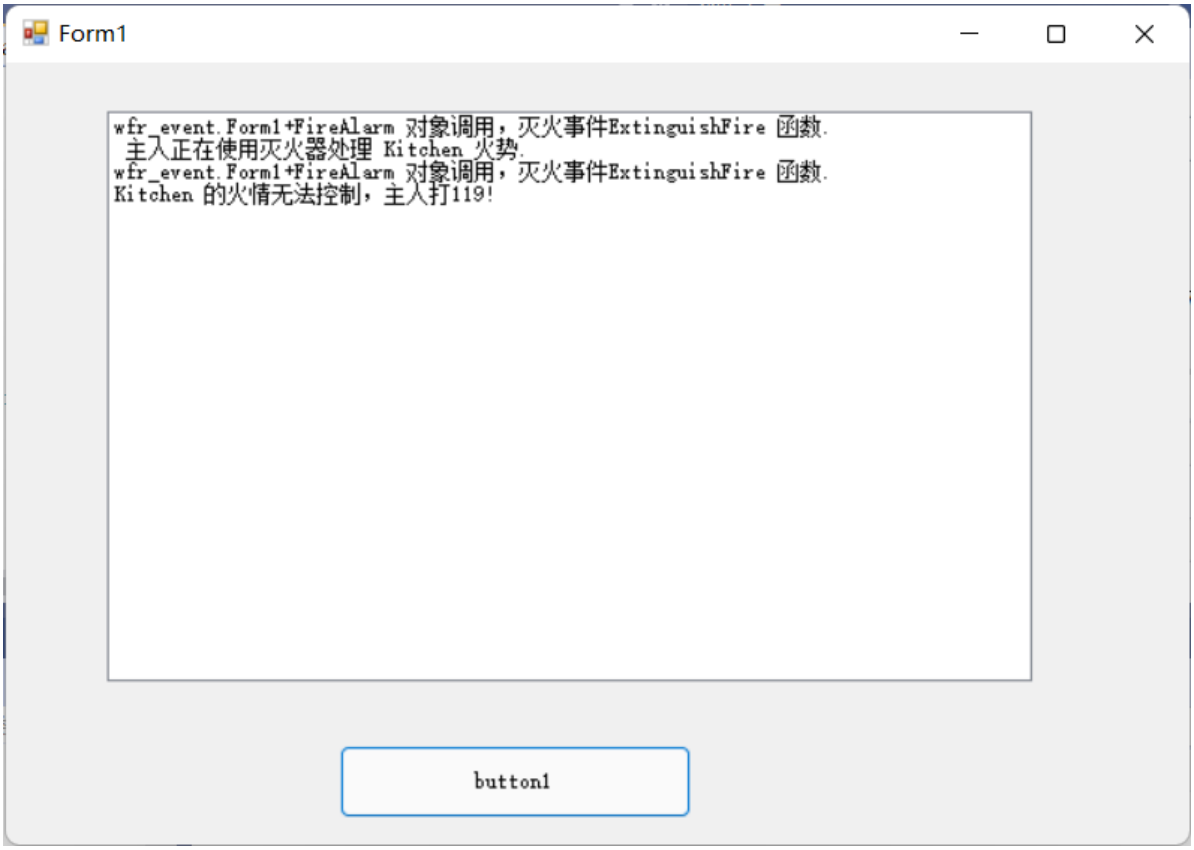
```
                    parent.showComment(" 主人正在使用灭火器处理 " + fe.room + " 火
势.");
                }
                else
                {
                    OutStr.sw.WriteLine("{0} 的火情无法控制，主人打119!", fe.room);
                    parent.showComment(fe.room + " 的火情无法控制，主人打119!");
                }
            }
```

2.4 激活事件

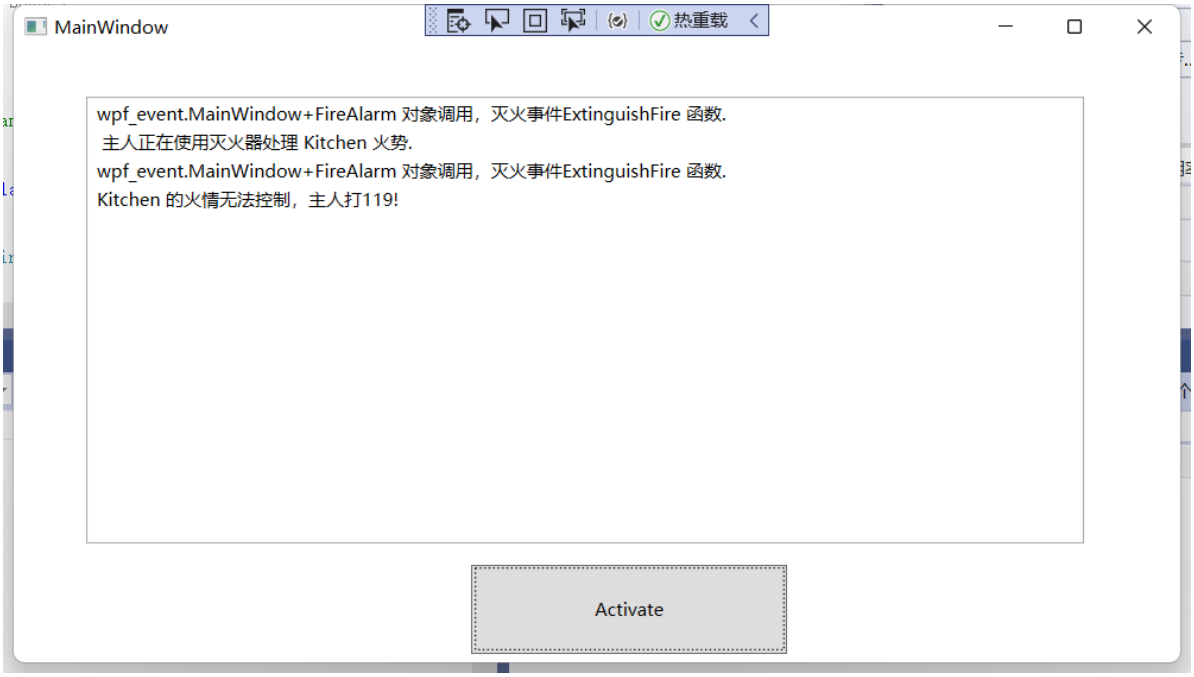
使用ActivateFireAlarm（）函数激活即可。

三、实验结果

对于WinForm



对于WPF：


MainWindow
热重载
wpf_event.MainWindow+FireAlarm 对象调用，灭火事件ExtinguishFire 函数.
主人正在使用灭火器处理 Kitchen 火势.
wpf_event.MainWindow+FireAlarm 对象调用，灭火事件ExtinguishFire 函数.
Kitchen 的火情无法控制，主人打119!
Activate