文章编号:1672 - 058X(2004)06 - 0585 - 03

回调函数的 C++ 封装

刘书良, 韩　力, 罗辞勇
(重庆大学 电气工程学院,重庆 400044)

摘　要:回调函数是应用程序参与操作系统运行的一个非常重要的接口,在 DirectX Play 开发过程中,经常需要使用到回调函数,直接使用回调函数显得复杂麻烦。介绍了用 C++ 完成回调函数的封装的方法,使回调函数的处理变得容易。

关键词:回调函数;C++ ;封装
中图分类号: TP 31　　**文献标识码**:A

在进行软件开发的过程中,常会用到一些声明为 CALLBACK 的函数,这些函数就是回调函数。使用回调函数可以改善软件的结构、提高软件的复用性。比如,在一个规模较大的软件项目中,可以将一些资源或相对独立的处理模块封装到动态连接库(DLL) 中,然后通过回调函数在不同的场合来使用这些资源和模块。利用回调函数还可以进行程序间复杂的通信,实现一些通知的功能,在某些场合它是比消息更合适的一种方式;在一些特殊的情况下,回调函数更有不可替代的作用。Win32 API 中有许多回调函数的应用,在进行软件设计时也会经常用到这种函数,而有些时候则需要编写自己的回调函数。因此,理解回调函数的原理并掌握它的基本用法是非常必要的。

C++ 是当代使用最广泛的语言,从嵌入式系统到大型机系统、从 LINUX 到 WINDOWS,在大型系统的编制中,到处都是它的身影。它以高效和易编程性获得了许多资深程序员的信赖。在 DirectX Play 开发过程中,经常需要使用到回调函数,直接使用回调函数显得复杂麻烦,采用用 C++ 实现对回调函数的封装,使回调函数变得方便实用,对于 DirectX Play 等编程就显得是非常有意义的。

回调函数简单讲就是一个函数指针。写一个函数,然后把函数地址传递给这个函数指针就可以了。回调函数的原形对 C++ 的成员函数用做回调函数的影响是什么?

编写回调函数简单地说就是函数原形一致。函数的输入参数,输出参数一致是很容易保证的。要注意调用类型一致性。函数传参数有好几种类型,搞错了传参数的方式,系统必然运行错误。一般来说都是 WINAPI 传参数方式。要注意 C++ 的类的成员函数和一般的 C 函数的区别。C++ 类采用 this 规则传递函数。在使用类的成员函数作为回调函数,要求该成员函数被声名为静态成员函数,并且注意函数声名的时候要同时声明好参数传递规则。

1　静态成员函数调用非静态成员函数

由于静态成员函数的存在是在类的实例产生之前,在每本 C++ 的书中都提到,静态成员函数只可以调用静态成员函数,并使用类的静态成员。但如果这样,写一个全部都是静态成员的类,似乎失去了C++ 类的大部分优点。它在完美的封装的同时,使得类不可以实例化。比如,对窗口消息处理的回调函数编程,系统中有无数个窗口,如果写一个全部都是静态成员的类,是无法描述这些窗口的。怎么在静态成员函数中使用类的非静态成员呢?

可以给静态成员函数传入参数,而且一般设计的比较好的回调函数都提供一个数据块指针作为传入

收稿日期:2004 - 03 - 26;**修回日期**:2004 - 05 - 25。
作者简介:刘书良(1980 -),男,江苏泰兴人,硕士研究生,主要从事电机与电器方面的研究。

© 1995-2005 Tsinghua Tongfang Optical Disc Co., Ltd.　All rights reserved.

参数,这时候就可以采用这种结构。

回调函数类型 void Fun(void *pData,UINT uMsg)

```
class CFun
{
  static void StaticFun(void *pData,UINT uMsg)
  {
    CFun *pThisObject = (CFun *)pData;
    pThisObject -> Fun(uMsg); //(1)
}
  void Fun(UINT nMsg)
  {
    }
  };
```

如此,回调函数就和 C++类形成了一个完美的体系。由于(1)处使用的是 CFun 的成员函数,该类生成的实例可以访问类的其他任何函数成员和数据成员。这样处理回调函数只是把 C++类和回调函数连接起来了。但还没有发挥出 C++的全部优势。现在把 FUN 的函数声明改一下。

2 函数的分发再处理

```
class CFun
  {
  static void StaticFun(void *pData,UINT uMsg);
  protected:
        vitual void Fun(UINT nMsg);
  };
```

同时写一个派生类,

```
class CSubFun: public CFun
{
        void Fun(UINT nMsg);
        void SubFun1();
        void SubFun2();
        ..
}
```

同时实现该函数如下

```
void CSubFun::Fun(UINT nMsg)
{
        switch(nMsg)
        {
        case 1:SubFun1(); break;
        case 2:SubFun2(); break;
        ...
        }
}
```

现在可以看到由于虚函数的使用,C++的派生类也具有了函数处理的能力。也就是说,实现了一个对回调函数封装的类,可以在基类中定义一些实现处理,但如果基类对回调函数的传入参数不符合要求,可以随时按照我们的要求修改它。

按照要求实现对基类处理的修改,上面的处理是有缺陷的。没有调用到基类的处理函数,这样丢弃了基类的处理例程是非常不可取的。可以重新做一下类的设计。修改派生类的函数 Fun 如下:

```
BOOL CSubFun::Fun(UINT nMsg)
  {
  bHandle = FALSE;
  switch(nMsg)
  {
  case 1: bHandle = SubFun1(); break;
  case 2: bHandle = SubFun2(); break;
  ...
  }
  if(bHandle)
  return TRUE;
  else
  return CFun::Fun(nMsg);
  }
```

至此,C++的类终于和回调函数完美的结合了。所有 C++的组合继承等等的优点全部重现。派生类可以决定是否处理该消息,甚至在该消息处理后继续要求基类去处理。

© 1995-2005 Tsinghua Tongfang Optical Disc Co., Ltd. All rights reserved.

3　回调函数分发的简化

其次,如果 nMsg 的数目是非常庞大的,那么函数码函数处理看起来就不那么愉快了。可以定义几个宏来处理:

```
# define DECLARE-FUN-TABLE() \
        virtual void Fun(UINT nMsg) ;
# define BEGIN-FUN-TABLE(theClass ,baseClass)
      BOOL theClass::Fun(UINT nMsg) { \
          bHandle = FALSE; \
          switch(nMsg) {
# define ON-FUN(nMsg ,Function) \
        case nMsg: bHandle = Funcion() ;break;
# define END-FUN-TABLE(theClass ,baseClass) } \
        if(bHandle) return TRUE; \
        else return baseClass::Fun(nMsg) ;}
```

如此一来,基类不变,派生类就改变如下:

```
class CSubFun : public CFun
{
BOOL SubFun1() ;
BOOL SunFun2() ;
...
DECLARE-FUN-TABLE()
}
BEGIN-FUN-TABLE(CSubFun ,CFun)
ON-FUN(1 ,SubFun1)
ON-FUN(2 ,SubFun2)
END-FUN-TABLE(CSubFun ,CFun)
```

这种类似的宏结构在 MFC 的消息映射机制中可以看到,但它没有采用虚函数实现。MFC 认为虚拟机制这种效率太低了,WINDOW 有数千个消息,窗口消息回调函数的调用是非常频繁的。但由于上述回调函数的调用频率比消息映射机制少的多,消息的数目也少的多,采用这种机制就可以了。如果想继续提升这种封装机制,可以参考以下 MFC 的消息映射机制和 ATL 的函数分发机制。

4　结束语

回调函数是 Windows 编程经常需要面对和处理的,在一个大型程序中,回调函数的处理的,变得非常重要。在讨论回调函数的编写规则的基础上,将类的成员函数作为回调函数来进行处理。在最后采用类似 MFC 消息映射机制方法,实现了在一个类中回调函数分发的简化。

参考文献:

[1] EUGEN O. MFC Virsual C++6 技术内幕[M]. 王建华,陈一飞,陈焕生,等译. 北京: 机械工业出版社,2000
[2] VICTOR S. C++精髓 - 软件工程方法[M]. 李师贤,译. 北京: 机械工业出版社,2002
[3] 王铬. 回调函数在软件设计中的应用[J]. 河南教育学院学报,2003,12(3): 44 - 46

C++ Encapsulation of callback function

LIU Shu - liang, HAN Li, LUO Ci - yong
(Electric Engineering College of Chongqing University, Chongqing 400044, China)

Abstract: Callback Function is very important interface for application in operating system. In the development of DirectX Play, callback Function is used frequently. It is inconvenient to make use of callback function directly. C++ is popular programming technology at present, In this paper, C++ encapsulation of callback function is finished, which makes treatment very easy to use recall function.

Key words: callback function; C++; encapsulation

责任编辑:杨祖彬

© 1995-2005 Tsinghua Tongfang Optical Disc Co., Ltd. All rights reserved.