**需求**

有三个软件A、B、C。A是主要软件，B、C是辅助软件。

⑴ 在使用A的过程中，按N键可以呼出软件B，按M键可以呼出软件C。再次按下可以隐藏它们。

⑵ 在软件B、C都存在的条件下，按J键可以实现鼠标焦点以A->B->C的顺序在软件之间切换。

**分析**

**需求（1）**

在按键事件中对M或N键做处理，当对应键按下时，首先判断B.exe或C.exe是否存在，如果不存在则执行对应exe，否则显示或隐藏它们。这里不再赘述。

**需求（2）**

以从A切到B为例，**由A通过UDP发消息给B，B收到消息后将焦点切到自身**。其他类似。

**定义通用变量**

主要定义软件端口、消息类型、发送者类型。后面将建立三个Qt工程，MainDlg代表软件A，compass代表软件B，chatlist代表软件C。定义如下：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Author: 张志浩

Mail: 791745123@qq.com

Time: 2019-1-5

Function:

通用变量、宏定义头文件

DESTPORT : 目标主机端口

SENDERTYPE : 发送者类型

MSGTYPE ： 消息类型

Version: v 1.0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#pragma once

typedef enum

{

PORT\_MAINDLG = 8000,

PORT\_COMPASS,

PORT\_CHATLIST

}DESTPORT;

typedef enum

{

PROCESS\_NONE = 100,

PROCESS\_MAINDLG,

PROCESS\_COMPASS,

PROCESS\_CHATLIST

}SENDERTYPE;

typedef enum

{

MSG\_NONE = 10,

MSG\_CHANGEMOUSEPOS

}MSGTYPE;

**设计通信报文**

写一个报文基类，包含消息类型、发送者、附加消息三类信息。后期可以继承它来丰富信息种类。实现如下：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Author: 张志浩

Mail: 791745123@qq.com

Time: 2019-1-5

Function:

报文基类

Version: v 1.0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#pragma once

#include "commonType.h"

#include <memory>

#define BUFF\_LENGTH 128

class CInfoBase

{

public:

CInfoBase():infoType(0), senderType(0)

{

memset(addMsg, 0, BUFF\_LENGTH);

}

bool InputAddMsg(const char\* buff, int length)

{

if (length < BUFF\_LENGTH && length > 0)

{

memcpy(addMsg, buff, length);

return true;

}

return false;

}

public:

//消息类型

int infoType;

//发送者

int senderType;

//附加消息

char addMsg[BUFF\_LENGTH];

};

**焦点切换**

焦点切换按以下几步进行：

//获取自身窗口句柄并置前

HWND hwnd = ::FindWindow(NULL, L"compass");

::SetForegroundWindow(hwnd);

//获取置前窗口句柄(该步骤可省略，直接用上一步获得的句柄)

HWND hForeWnd = ::GetForegroundWindow();

//获取当前工作线程ID

DWORD dcurid = ::GetCurrentThreadId();

//获取置前窗口的线程ID

DWORD dfoid = ::GetWindowThreadProcessId(hForeWnd, NULL);

//依附

::AttachThreadInput(dcurid, dfoid, TRUE);

//设置鼠标位置

QRect rect = this->geometry();

SetCursorPos(rect.left() + 200, rect.top() + 200);

**依附的步骤是必要的，因为如果不依附，就算鼠标位置从A移到B了，此时的键盘输入焦点还在A。按J键只会进入A的键盘事件，除非手动点击一下B再按J。**

**整体流程梳理**

以从MainDlg切往compass为例，此时三个软件都已经打开并显示在桌面。

1 按J键进入MainDlg键盘事件，发送消息到compass

void MainDlg::keyPressEvent(QKeyEvent \* event)

{

switch(event->key())

{

case Qt::Key\_J:

{

CInfoBase m\_sendMsg;

m\_sendMsg.infoType = MSG\_CHANGEMOUSEPOS;

m\_sendMsg.senderType = PROCESS\_MAINDLG;

m\_myudp.SendData((char\*)&m\_sendMsg, sizeof(m\_sendMsg), PORT\_COMPASS, "127.0.0.1");

break;

}

default:

{

}

}

}

2 compass收到UDP消息，切换焦点

void CUdpMsgBase::DataHanding(const char\* data)

{

int msgType = MSG\_NONE;

::memcpy(&msgType, data, sizeof(int));

if (msgType == MSG\_NONE)

{

}

if (msgType == MSG\_CHANGEMOUSEPOS)

{

emit changepos();

}

return;

}

进入槽函数

void MainDlg::changePos()

{

HWND hwnd = ::FindWindow(NULL, L"MainDlg");

::SetForegroundWindow(hwnd);

HWND hForeWnd = ::GetForegroundWindow();

DWORD dcurid = ::GetCurrentThreadId();

DWORD dfoid = ::GetWindowThreadProcessId(hForeWnd, NULL);

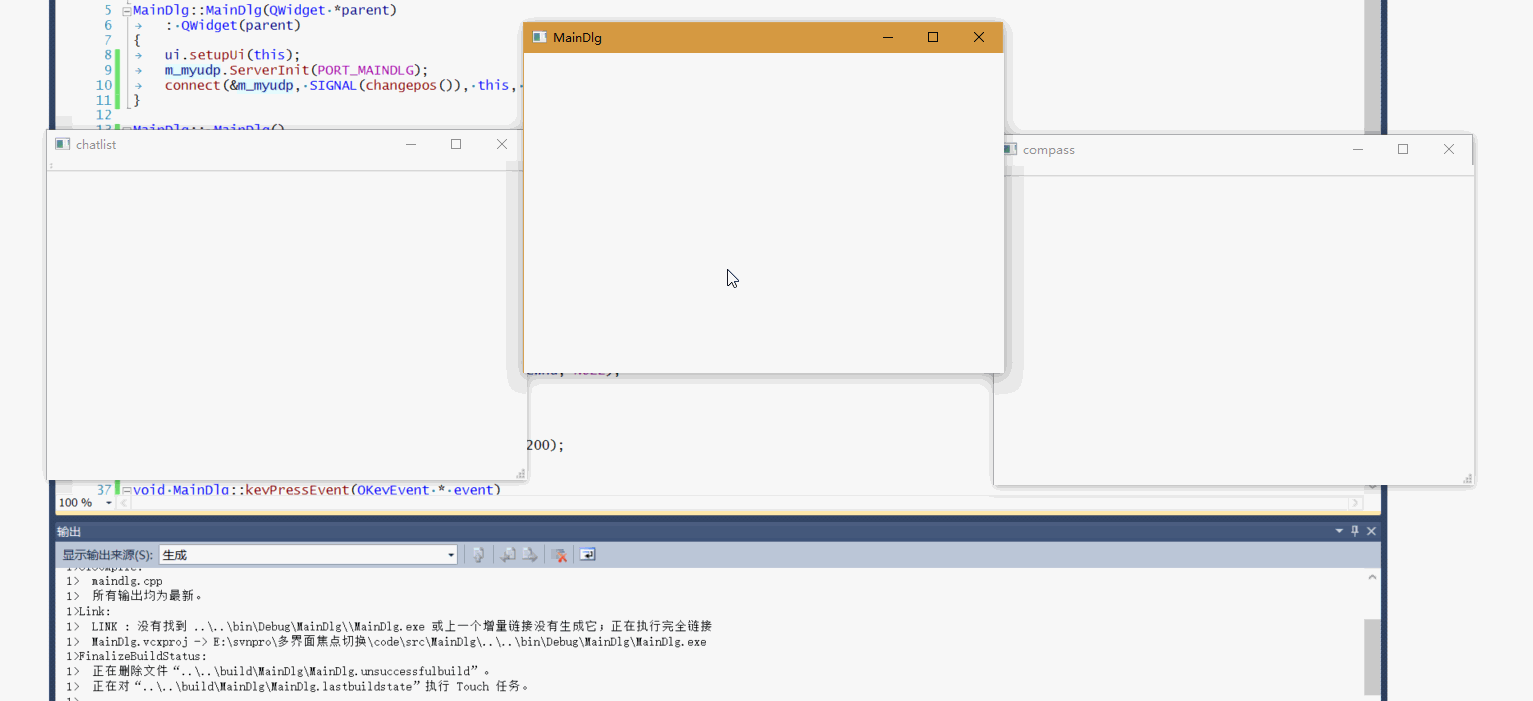
::AttachThreadInput(dcurid, dfoid, TRUE);

QRect rect = this->geometry();

SetCursorPos(rect.left() + 200, rect.top() + 200);

}

**效果**

​

**总结**

主要考察到对AttachThreadInput的运用，还有就是设计好三者之间切换的流程。**即将焦点切换的动作交给目标进程来做，自身进程只负责发消息**，避免逻辑混乱。