http://www.cnblogs.com/findumars/p/6006070.html

Qt键盘事件属于[Qt事件系统](http://hi.baidu.com/cyclone/blog/item/fe6ab3de0e9f2155ccbf1aea.html),所以事件系统中所有规则对按键事件都有效。下面关注点在按键特有的部分：

##### focus

一个拥有焦点(focus)的QWidget才可以接受键盘事件。有输入焦点的窗口是活动窗口或活动窗口子窗口或子子窗口等。

焦点移动的方式有以下几种：

* 按下Tab或Shift+Tab
  + 注意：文本编译器（一般需要插入Tab），或者WebView(需要Tab来移动超链接焦点) 等
  + Qt中，需要输入Tab的地方可以用 Ctrl+Tab 或 Ctrl+Shift+Tab 替代。
* 点击一个QWidget
  + 建议：只对接受文本输入的Widget启用该功能
* 按下键盘的快捷键
  + QLabel::setBuddy(), QGroupBox,以及 QTabBar 支持
* 使用鼠标滚轮
* 用户移动焦点
  + 程序将决定被设置focus的Widget的哪一个子Widget获得焦点

注意：如果一个 Widget 已经 grabKeyboard，所有键盘事件将发送到该Widget而不是获得焦点的Widget

##### focusPolicy

一个QWidget获得焦点的方式受 focusPolicy 控制

Qt::TabFocus

通过Tab键获得焦点

Qt::ClickFocus

通过被单击获得焦点

Qt::StrongFocus

可通过上面两种方式获得焦点

Qt::NoFocus

不能通过上两种方式获得焦点(默认值),setFocus仍可使其获得焦点

##### keypress和keyrelease

首先，我们要是Widget获得焦点，一般设置focusPolicy。

然后要对按键进行响应，我们只需要直接重载：

* keyPressEvent
* keyReleaseEvent

注意：

* 对我们不处理的事件，要调用父类的相应事件处理函数。
* 如果widget当前没有焦点，考虑到事件转发：如果其子widget有焦点，那么该widget未处理的键盘事件将被转发过来。
* 有时输入焦点不在任何窗口中。这种情况发生在所有程序都是最小化的时候。这时，Windows将继续向活动窗口发送键盘消息，但是这些消息与发送给非最小化的活动窗口的键盘消息有不同的形式。

##### QKeyEvent

在windows下，与键盘事件有关的有8个消息：

* 对产生可显示字符的按键组合，Windows不仅给程序发送按键消息，而且还发送字符消息
* 有些键不产生字符，这些键包括shift键、功能键、光标移动键和特殊字符键如Insert和Delete。对于这些键，Windows只产生按键消息。

这些消息在Qt中只体现在QKeyEvent中。

* 对字符，可通过 QKeyEvent::text() 获得
* 其他键，QKeyEvent::key() 获得一个键值

实际程序：

public: void keyPressEvent(QKeyEvent  \*event);

在相应键盘事件之前需要 用一个widget 进行设置   LabelComment->setFocusPolicy(Qt::StrongFocus);

void CameraShow::keyPressEvent(QKeyEvent  \*event)   
{

     if(event->key()==Qt::Key\_Q)   
    {   
        Pause\_flag  ^= 1;   
    }

}

还有资料：（QTextBrowser的网址切换）

<http://hi.baidu.com/%D7%D4%D3%EF%B5%C4%C2%E6%CD%D5/blog/item/2758cf13d83f945df819b806.html>

在Qt中,可以使用 void QWidget::keyPressEvent ( [QKeyEvent](http://www.qiliang.net/qt/qkeyevent.html) \* k )来进行键盘响应,例如:

void Form1::keyPressEvent( QKeyEvent \*k )   
{   
    if(k->key() == Key\_A)   
    {   
      this->focusNextPrevChild(FALSE);//按A时焦点切换至上一部件   
    }   
    else if(k->key() == Key\_D)   
    {   
      this->focusNextPrevChild(TRUE);//按D时焦点切换至下一部件   
    }   
    else if(k->key() == Key\_W)   
    {   
      if(k->state() == Qt::ShiftButton)   
      {   
         this->resize(100,100);//当按下Shift+W时改变窗口大小   
      }   
    }   
}

但是,有一些特殊的按键比如说Tab键,如果在keyPressEvent中实现则是不能成功的,因为默认Tab事件(切换焦点)被先捕获了,默认Tab和Shift+Tab事件定义在qwidget.h中,代码为:

case QEvent::KeyPress: {   
        QKeyEvent \*k = (QKeyEvent \*)e;   
        bool res = FALSE;   
        if ( k->key() == Key\_Backtab ||   
        (k->key() == Key\_Tab &&   
        (k->state() & ShiftButton)) ) {   
        QFocusEvent::setReason( QFocusEvent::Tab );   
        res = focusNextPrevChild( FALSE );   
        QFocusEvent::resetReason();   
        } else if ( k->key() == Key\_Tab ) {   
        QFocusEvent::setReason( QFocusEvent::Tab );   
        res = focusNextPrevChild( TRUE );   
        QFocusEvent::resetReason();   
        }   
}

所以我们要在之前就实现我们自己的Tab事件.实现代码如下:

bool MyWidget::event(QEvent \*event)

{

if (event->type() == QEvent::KeyPress) {

QKeyEvent \*ke = static\_cast(event);

if (ke->key() == Qt::Key\_Tab) {

// special tab handling here

return true;

}

} else if (event->type() == MyCustomEventType) {

MyCustomEvent \*myEvent = static\_cast(event);

// custom event handling here

return true;

}

return QWidget::event(event);

}

QTextBrowser中的网址切换   
void ALMTextView::keyPressEvent(QKeyEvent\* e)   
{   
   AMDEBUG("ALMTextView special key event/n");   
   QScrollBar \*sbv = verticalScrollBar();   
    switch( e->key() ) {   
        //case Key\_Right:   
        case Key\_Down:   
            if ( !selectNextPrevHref( TRUE ) )   
            {   
                // scroll the screen down by one page   
            if (sbv->value() == sbv->maxValue())   
            {   
                printf("asdasdasdasdasdasda/n");   
                this->focusNextPrevChild(TRUE);   
            }   
                sbv->setValue( sbv->value() + (sbv->pageStep() >> 1) );   
                selectNextPrevHref( TRUE );   
            }   
            e->accept();   
            return;   
        //case Key\_Left:   
        case Key\_Up:   
            if ( !selectNextPrevHref( FALSE ) ){   
                // scroll the screen up by one page   
                if(sbv->value() == 0)   
                {   
                        printf("1234567890/n");   
                        this->focusNextPrevChild(FALSE);   
                }   
                sbv->setValue( sbv->value() - (sbv->pageStep() >> 1) );   
                selectNextPrevHref( FALSE );   
            }   
            e->accept();   
            return;   
    }   
    QTextView::keyPressEvent(e);   
}

QKeyEvent类参考  
QKeyEvent类用于描述键盘按键所产生的QT事件  
#include<QKeyEvent>  
继承于QInpueEvent  
公有函数：

QKeyEvent ( Type type, int key, Qt::KeyboardModifiers modifiers, const QString & text = QString(), bool autorep = false, ushort count = 1 )

int count () const  
bool isAutoRepeat () const  
int key () const  
bool matches ( QKeySequence::StandardKey key ) const  
Qt::KeyboardModifiers modifiers () const  
quint32 nativeModifiers () const  
quint32 nativeScanCode () const  
quint32 nativeVirtualKey () const  
QString text () const  
相关的非成员函数：  
bool operator== ( QKeyEvent \* e, QKeySequence::StandardKey key )  
bool operator== ( QKeySequence::StandardKey key, QKeyEvent \* e )  
细节描述：  
QKeyEvent类用于描述键盘按键所产生的QT事件，当一个窗口具有输入焦点时，通过键盘的按键的（敲击/释放）所产生的QT键盘事件  
可以传送给具有输入焦点的窗口进行处理。  
键盘QT事件包含了一些指定的接收标志用于指出QT事件接受者是否对该事件进行处理，如果窗口不处理键盘的（敲击/释放）  
可以通过调用ignore()函数予以忽略。键盘QT事件具有传递性，会传送给父窗口及顶层窗口所构成的链表，一层一层的传递，  
直到某窗口调用accept()予以接受，或者过滤直至该事件被销毁。键盘QT事件对于多媒体在默认情况下总是被忽略的，  
如果你的窗口要处理该事件，可以通过调用accept()函数予以接收。  
  
  
可以通过调用QWidget::setEnable()函数设置某个窗口开启/禁止接收鼠标或键盘事件。  
QT事件句柄函数 QWidget::keyPressEvent(), QWidget::keyReleaseEvent(), QGraphicsItem::keyPressEvent()   
, QGraphicsItem::keyReleaseEvent()可以用于接收键盘事件。  
  
  
成员函数描述：

QKeyEvent::QKeyEvent ( Type type, int key, Qt::KeyboardModifiers modifiers, const QString & text = QString(), bool autorep = false, ushort count = 1 )

初始化一个keyEvent对象。  
参数type必须是QEvent::KeyPress, QEvent::KeyRelease, or QEvent::ShortcutOverride其中之一。   
参数key用于监听QT::Key循环事件，如果key为0则导致键盘事件的是一个未定义按键，例如，导致该键盘事件的可能是  
连续不断的按键下压操作或者是预定义的快捷键。    
参数modifiers用于指出键盘按键键值的改变，同时给出该按键对应的Unicode编码信息。  
如果参数autorep为true，函数isAutoRepeat()函数就返回true  
参数count用于计算该键盘事件所牵扯到得按键的个数。  
  
  
int QKeyEvent::count () const  
返回键盘QT事件所牵扯到的按键的数量，如果函数text()返回不为空，则本函数只是简单的计算该文本字符串的长度。  
  
  
bool QKeyEvent::isAutoRepeat () const  
如果该键盘QT事件时由于一些列的自动重复的按键导致的则返回true，否则返回false  
注意，如果导致多次叠加的按键组合其中一部分是自动循环形式产生的，则该函数不确定是返回true还是false。  
  
  
int QKeyEvent::key () const  
返回按键被下压或释放时对应的编码值。查看Qt::Key键盘编码清单，这些编码信息都是依赖于底层操作系统的。  
该函数不区分大小写，要想知道返回的按键编码所对应的按键是大写还是小写（比如A和a）可以通过调用text()函数得到。    
数字0和Qt::Key\_unknown代表导致QT键盘事件的不是一个well\_known按键所产生的，例如，导致该键盘事件的可能是一系列复杂的输入，或者是快捷键，或者是被压缩的按键所导致的。  
  
bool QKeyEvent::matches ( QKeySequence::StandardKey key ) const  
如果导致键盘QT事件的按键编码值和参数key所给出的标准按键值匹配则返回true，否则返回false.  
该函数在QT4.2中有介绍。该函数可以通过迭代方式得知当前按下的是哪一个按键。  
  
  
Qt::KeyboardModifiers QKeyEvent::modifiers () const  
键盘QT事件生成后直接返回导致该事件的热键。提示：该函数并不总是返回很确切的结果，当用户同时按下shift+组合键或者  
释放其中某一个键时，该函数就可能返回一个让用户迷惑的结果。  
  
  
quint32 QKeyEvent::nativeModifiers () const  
返回导致键盘QT事件的热键编码值，如果该事件不包含所对应的热键则返回数字0.  
附注：即便键盘事件包含扩展信息，该热键也可能是为0的。  
该函数在QT4.2中有介绍。  
  
  
quint32 QKeyEvent::nativeScanCode () const  
返回导致该键盘QT事件的扫描码，如果敢事件不包含所对应的扫描码则返回数字0.  
附注：即便键盘事件包含扩展信息，该扫描码也可能是为0的。  
附注：在Mac OS/X 系统上该函数式无效的，因为没有办法从Cocoa或者Carbon中获取扫描码，该函数通常返回1，或者当包含扩展信息时返回0.  
该函数在QT4.2中有介绍  
  
  
quint32 QKeyEvent::nativeVirtualKey () const  
返回导致键盘QT事件的虚拟键或者字符码，如果该键盘事件没有包含该按键的编码则返回0.  
附注：即便键盘事件包含扩展信息，该虚拟键也可能是为0的。   
该函数在QT4.2中有介绍  
  
  
QString QKeyEvent::text () const  
返回导致的键盘事件的Unicode字符编码信息，当导致键盘事件的原因是由热键比如Shift, Control, Alt等的下压或者释放，  
则该函数返回的字符可以为空值。在这种情况下函数key()可以返回一个有效值。  
  
  
相关的非成员函数：  
bool operator== ( QKeyEvent \* e, QKeySequence::StandardKey key )  
如果参数e和参数key类型匹配且相等则返回true  
等同于函数 e->matches(key).  
  
  
bool operator== ( QKeySequence::StandardKey key, QKeyEvent \* e )  
如果参数e和参数key类型匹配且相等则返回true  
等同于函数 e->matches(key).

http://blog.csdn[.NET](http://lib.csdn.net/base/dotnet)/zzk197/article/details/7383715

http://blog.csdn[.Net](http://lib.csdn.net/base/dotnet)/free\_program\_1314/article/details/7681456

http://blog.csdn.net/huyisu/article/details/30455563

分类: [Qt-事件](http://www.cnblogs.com/findumars/category/621018.html)