欢迎大家来到第五阶段课程

《分布式流媒体》实训项目

TNV DAY04

复习课

预习内容

媒体播放器（5）

媒体播放器（5）

主窗口类

知识讲解

- 响应媒体长度改变事件的静态处理函数
  - 调用普通处理函数
- 响应媒体长度改变事件的普通处理函数
  - 将媒体长度毫秒值设置为进度滑块控件的最大值
  - 将媒体长度毫秒值的十分之一设置为进度滑块控件的页步距
  - 将媒体长度毫秒值的百分之一设置为进度滑块控件的刻度间隔
  - 将媒体长度毫秒值保存到成员变量中以备后用

主窗口类

知识讲解

- 响应视频输出启动事件的静态处理函数
  - 调用普通处理函数
- 响应视频输出启动事件的普通处理函数
  - 枚举视频框控件的所有子窗口，即VLC视频输出窗口，每个子窗口的句柄将作为参数，传递给枚举回调函数
- 子窗口枚举回调函数
  - 禁用该子窗口，即VLC视频输出窗口，避免鼠标停留引发界面崩溃

主窗口类

知识讲解

- 响应时间改变事件的静态处理函数
  - 调用普通处理函数
- 响应时间改变事件的普通处理函数
  - 根据媒体时间毫秒值同步进度滑块控件
  - 根据媒体时间毫秒值同步进度标签控件
- 响应终点到达事件的静态处理函数
  - 调用普通处理函数
- 响应终点到达事件的普通处理函数
  - 向停止按钮发射点击信号

主窗口类

知识讲解

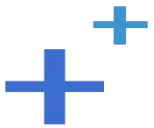
- 显示窗口时被调用的虚函数
  - 若非内嵌视频，则将窗口收缩至最小
- 视频框控件有事件时被调用的虚函数
  - 若视频框控件发生鼠标双击事件
    - 从窗口模式切换到全屏模式
      - 隐藏光标，进入全屏，调整界面
    - 从全屏模式恢复到窗口模式
      - 调整界面，退出全屏，恢复光标，激活窗口
  - 其它事件
    - 交由基类处理

附录：程序清单

QtPlayer/Src/MainWindow.cpp

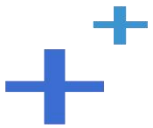
知识讲解

```
//响应媒体长度改变事件的静态处理函数
void MainWindow::onMediaPlayerLengthChanged(
    libvlc_event_t const* event, void* data)
{
    //调用普通处理函数
    reinterpret_cast<MainWindow*>(data)->onMediaPlayerLengthChanged(
        event->u.media_player_length_changed.new_length);
}

//响应媒体长度改变事件的普通处理函数
void MainWindow::onMediaPlayerLengthChanged(libvlc_time_t length)
{
    //将媒体长度毫秒值设置为进度滑块控件的最大值
    ui->sliderProgress->setMaximum(length);
    //将媒体长度毫秒值的十分之一设置为进度滑块控件的页步距
```



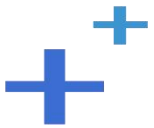
QtPlayer/Src/MainWindow.cpp


```
    ui->sliderProgress->setPageStep(length / 10);
    //将媒体长度毫秒值的百分之一设置为进度滑块控件的刻度间隔
    ui->sliderProgress->setTickInterval(length / 100);
    //将媒体长度毫秒值保存到成员变量中以备后用
    vlcMediaLength = length;
}

#ifdef Q_OS_WINDOWS

//子窗口枚举回调函数
BOOL CALLBACK MainWindow::onEnumVLCWindow(HWND hwnd, LPARAM)
{
    //禁用该子窗口，即VLC视频输出窗口，避免鼠标停留引发界面崩溃
    EnableWindow(hwnd, FALSE);
    return TRUE;
```



QtPlayer/Src/MainWindow.cpp


```
}

//响应视频输出启动事件的静态处理函数
void MainWindow::onMediaPlayerVout(
    libvlc_event_t const*, void* data)
{
    //调用普通处理函数
    reinterpret_cast<MainWindow*>(data)->onMediaPlayerVout();
}

//响应视频输出启动事件的普通处理函数
void MainWindow::onMediaPlayerVout(void)
{
    //枚举视频框控件的所有子窗口，即VLC视频输出窗口
    //每个子窗口的句柄将作为参数，传递给枚举回调函数
```



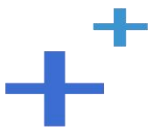
QtPlayer/Src/MainWindow.cpp


```
    EnumChildWindows((HWND)ui->frmVideo->winId(), onEnumVLCWindow, 0);
}

#endif // Q_OS_WINDOWS

//响应时间改变事件的静态处理函数
void MainWindow::onMediaPlayerTimeChanged(
    libvlc_event_t const* event, void* data)
{
    //调用普通处理函数
    reinterpret_cast<MainWindow*>(data)->onMediaPlayerTimeChanged(
        event->u.media_player_time_changed.new_time);
}

//响应时间改变事件的普通处理函数
```

QtPlayer/Src/MainWindow.cpp

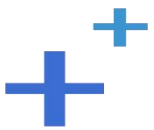
知识讲解

```
void MainWindow::onMediaPlayerTimeChanged(libvlc_time_t time)
{
    //根据媒体时间毫秒值同步进度滑块控件
    ui->sliderProgress->setValue(time);
    //根据媒体时间毫秒值同步进度标签控件
    ui->labProgress->setText(
        QTime::fromMSecsSinceStartOfDay(time).toString("hh:mm:ss.zzz") + "/" +
        QTime::fromMSecsSinceStartOfDay(vlcMediaLength).toString("hh:mm:ss.zzz"));
}

//响应终点到达事件的静态处理函数
void MainWindow::onMediaPlayerEndReached(
    libvlc_event_t const*, void* data)
{
    //调用普通处理函数
```



QtPlayer/Src/MainWindow.cpp

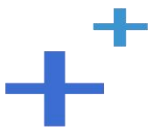

```
    reinterpret_cast<MainWindow*>(data)->onMediaPlayerEndReached();
}

//响应终点到达事件的普通处理函数
void MainWindow::onMediaPlayerEndReached(void)
{
    //向停止按钮发射点击信号
    emit ui->btnStop->click();
}

//显示窗口时被调用的虚函数
void MainWindow::showEvent(QShowEvent*)
{
    //若非内嵌视频...
    if (!embed)
```



QtPlayer/Src/MainWindow.cpp

知识讲解

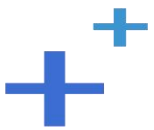
```
        //将窗口收缩至最小
        resize(minimumSize());
}

//视频框控件有事件时被调用的虚函数
bool MainWindow::eventFilter(QObject* obj, QEvent* event)
{
    //若视频框控件发生鼠标双击事件...
    if (obj == ui->frmVideo && event->type() == QEvent::MouseButtonDblClick)
    {
        static bool     fullscreen = false; //全屏模式
#ifdef Q_OS_WINDOWS
        static int      minHeight;          //最小高度
        static QSize    winSize;            //窗口大小
#else
```



QtPlayer/Src/MainWindow.cpp

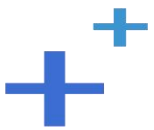
知识讲解

```
        static QMargins cntMargins;          //中心布局的边缘大小
        static int      frmStyle;            //屏幕框控件的边框风格
#endif // Q_OS_WINDOWS

        if (fullscreen) //全屏->窗口
        {
#ifdef Q_OS_WINDOWS
            //调整界面
            setMinimumHeight(minHeight);
            resize(winSize);
            ui->frmScreen->show();
            //退出全屏
            ui->frmVideo->setWindowFlags(Qt::SubWindow);
            ui->frmVideo->showNormal();
#else
```



QtPlayer/Src/MainWindow.cpp

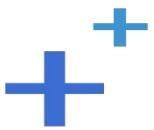

```
            //调整界面
            ui->labFullscreen->show();
            ui->frmControl->show();
            ui->layoutCentral->setContentsMargins(cntMargins);
            ui->frmScreen->setFrameStyle(frmStyle);
            //退出全屏
            showNormal();
#endif // Q_OS_WINDOWS
            //恢复光标
            ui->frmVideo->unsetCursor();
            //激活窗口
            qApp->setActiveWindow(this);
        }
        else //窗口->全屏
        {
```



QtPlayer/Src/MainWindow.cpp

知识讲解

```
        //隐藏光标
        ui->frmVideo->setCursor(Qt::BlankCursor);
#ifdef Q_OS_WINDOWS
        //进入全屏
        ui->frmVideo->setWindowFlags(Qt::Window);
        ui->frmVideo->showFullScreen();
        //调整界面
        ui->frmScreen->hide();
        minHeight = minimumHeight();
        winSize = size();
        setMinimumHeight(
            ui->layoutCentral->contentsMargins().top() +
            ui->labFullscreen->size().height() +
            ui->layoutCentral->spacing() +
            ui->frmControl->size().height() +
```



QtPlayer/Src/MainWindow.cpp

知识讲解

```
                ui->layoutCentral->contentsMargins().bottom());
            resize(size().width(), minimumHeight());
#else
            //进入全屏
            showFullScreen();
            //调整界面
            ui->labFullscreen->hide();
            ui->frmControl->hide();
            cntMargins = ui->layoutCentral->contentsMargins();
            ui->layoutCentral->setContentsMargins(QMargins());
            frmStyle = ui->frmScreen->frameStyle();
            ui->frmScreen->setFrameStyle(QFrame::NoFrame);
#endif // Q_OS_WINDOWS
        }

        fullscreen = !fullscreen;
        return true;
    }

    return QMainWindow::eventFilter(obj, event);
}
```

下节课见