```
#include "videosink.h"
#include "ui_videosink.h"


VideoSink::VideoSink(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::VideoSink)
{
    ui->setupUi(this);
}

VideoSink::VideoSink(QWidget *parent,QtGStreamerDriver * vGstDriver):QMainWindow(parent),
    ui(new Ui::VideoSink)
{
    ui->setupUi(this);

    ui->verticalLayoutVideoLayout->addWidget(vGstDriver);

    mGstDriver=vGstDriver;

    //    ui->toolButtonNext->setIcon(ui->centralwidget->style()-
>standardIcon(QStyle::SP_MediaSkipForward));
    //    ui->toolButtonPrevious->setIcon(ui->centralwidget->style()-
>standardIcon(QStyle::SP_MediaSkipBackward));
    ui->toolButtonFullScreen->setIcon(ui->centralwidget->style()-
>standardIcon(QStyle::SP_TitleBarMaxButton));
    //    ui->toolButtonVolume->setIcon(ui->centralwidget->style()-
>standardIcon(QStyle::SP_MediaVolume));
    ui->toolButtonToMain->setIcon(ui->centralwidget->style()-
>standardIcon(QStyle::SP_TitleBarMinButton));


    connect(mGstDriver, SIGNAL(positionChanged()), this, SLOT(onPositionChanged()));
    connect(mGstDriver, SIGNAL(stateChanged()), this, SLOT(onStateChanged()));

    onStateChanged();

    //    ui->toolButtonPlayPause->setIcon(ui->centralwidget->style()-
>standardIcon(QStyle::SP_MediaPlay));
    //    ui->toolButtonPlayPause->setIconSize(QSize(32,32));

    mPlay.addFile(":/ui/toolButton/icon/Resources/ui/toolButton/icon/play.png");
    mPause.addFile(":/ui/toolButton/icon/Resources/ui/toolButton/icon/pause.png");
    mVolume.addFile(":/ui/toolButton/icon/Resources/ui/toolButton/icon/volume.png");
    mVolumeMuted.addFile(":/ui/toolButton/icon/Resources/ui/toolButton/icon/
volumeMuted.png");
    ui->toolButtonPlayPause->setIcon(mPlay);

    connect(ui->toolButtonPlayPause,SIGNAL(clicked()),this,SLOT(setPlayPause_clicked()));
    connect(this,SIGNAL(setPlayState()),mGstDriver,SLOT(play()));
    connect(this,SIGNAL(setPauseState()),mGstDriver,SLOT(pause()));


    connect(ui->toolButtonNext,SIGNAL(clicked()),this,SLOT(setNext_clicked()));
    connect(ui->toolButtonPrevious,SIGNAL(clicked()),this,SLOT(setPrevious_clicked()));


    ui->horizontalSliderMediaPosition->setTracking(false);
    connect(ui-
>horizontalSliderMediaPosition,SIGNAL(valueChanged(int)),this,SLOT(positionSliderMoved(in
t)));

    ui->horizontalSliderVolume->setMaximum(10);
    connect(ui-
>horizontalSliderVolume,SIGNAL(valueChanged(int)),this,SLOT(setVolume(int)));
    ui->horizontalSliderVolume->setValue(5);
    ui->horizontalSliderVolume->setSliderPosition(5);

    connect(ui->toolButtonFullScreen,SIGNAL(clicked()),this,SLOT(toFullScreen()));
```

```
    connect(ui->toolButtonVolume,SIGNAL(clicked()),this,SLOT(toggleMute()));
    connect(ui->toolButtonToMain,SIGNAL(clicked()),this,SLOT(toLibraryMode()));


    //    setMouseTracking(true);
    //    mGstDriver->setMouseTracking(true);
    //    parent->setMouseTracking(true);
    mShowControlsTimer.setSingleShot(true);
    connect(&mShowControlsTimer, SIGNAL(timeout()), this, SLOT(hideControls()));
    this->installEventFilter(this);
    //    mShowControlsTimer.start(3000);
}

VideoSink::~VideoSink()
{
    delete ui;
}

void VideoSink::closeEvent(QCloseEvent * vEvent)
{
    emit(closeMain());
    vEvent->accept();
}


//slots


void VideoSink::onStateChanged()
{

    QGst::State vNewState = mGstDriver->getState();
    //    ui->toolButtonPlayPause->setEnabled(vNewState != QGst::StatePlaying);
    if(vNewState == QGst::StateNull || vNewState == QGst::StatePaused)
    {
        ui->toolButtonPlayPause->setIcon(mPlay);
        //        ui->toolButtonPlayPause->setIconSize(QSize(32,32));
        isPlaying=false;
    }
    else if(vNewState == QGst::StatePlaying)
    {
        mGstDriver->setAutoFillBackground(true);
        ui->toolButtonPlayPause->setIcon(mPause);
        //        ui->toolButtonPlayPause->setIconSize(QSize(32,32));
        mTempTime = 0;
        mTempTime = mTempTime + ( mGstDriver->getDuration().hour() * 60);
        mTempTime = ( mTempTime +  mGstDriver->getDuration().minute() ) * 60;
        mTempTime = mTempTime + (mGstDriver->getDuration().second());
        ui->horizontalSliderMediaPosition->setMaximum(mTempTime);
        isPlaying=true;
    }

    if(vNewState == QGst::StateNull)
    {
        ui->horizontalSliderMediaPosition->setValue(0);
        ui->horizontalSliderMediaPosition->setSliderPosition(0);
        //        QMessageBox::information(this,"State Changed","Playing");
    }

    ui->toolButtonNext->setEnabled(vNewState != QGst::StateNull);
    ui->toolButtonPrevious->setEnabled(vNewState != QGst::StateNull);
    ui->horizontalSliderMediaPosition->setEnabled(vNewState != QGst::StateNull);
    //    ui->labelVolume->setEnabled(vNewState != QGst::StateNull);

    if(vNewState == QGst::StatePaused || vNewState == QGst::StatePlaying)
    {
        ui->labelTime->show();
        ui->labelLength->show();
        ui->labelTime->setEnabled(true);
```

```
        ui->horizontalSliderMediaPosition->setEnabled(true);
    }
    else
    {
        ui->labelTime->hide();
        ui->labelLength->hide();
    }
    //if we are in Null state, call onPositionChanged() to restore
    //the position of the slider and the text on the label
    if (vNewState == QGst::StateNull) {
        onPositionChanged();
    }
}

void VideoSink::onPositionChanged()
{

    if (mGstDriver->getState() != QGst::StateReady && mGstDriver->getState() !=
QGst::StateNull)
    {
        mPlayBacklength = mGstDriver->getDuration();
        if(mPlayBacklength.hour()==0)
        {
            ui->labelLength->setText(mPlayBacklength.toString("mm:ss"));
        }
        else
        {
            ui->labelLength->setText(mPlayBacklength.toString("HH:mm:ss"));
        }
        mPlayBackcurpos = mGstDriver->getPosition();
    }

    if(mPlayBackcurpos.hour() ==0)
    {
        ui->labelTime->setText(mPlayBackcurpos.toString("mm:ss"));
    }
    else
    {
        ui->labelTime->setText(mPlayBackcurpos.toString("HH:mm:ss"));
    }
    if(mGstDriver->getState()!= QGst::StateNull)
    {
        mTempTime = 0;
        mTempTime = mTempTime + ( mPlayBackcurpos.hour() * 60);
        mTempTime = ( mTempTime +  mPlayBackcurpos.minute() ) * 60;
        mTempTime = mTempTime + (mPlayBackcurpos.second());
        if(ui->horizontalSliderMediaPosition->isSliderDown()==false)
        {
            ui->horizontalSliderMediaPosition->setSliderPosition(mTempTime);
        }
    }
    else
    {
        ui->horizontalSliderMediaPosition->setValue(0);
        ui->horizontalSliderMediaPosition->setSliderPosition(0);
        //        QMessageBox::information(this,"State Changed","Playing");


    }

}


void VideoSink::setPlayPause_clicked()
{
    if(isPlaying==true)
    {
        isPlaying=false;
        emit(setPauseState());
```

```
    }
    else
    {
        isPlaying=true;
        emit(setPlayState());
    }

}

void VideoSink::setNext_clicked()
{
    emit(nextClicked());
}

void VideoSink::setPrevious_clicked()
{
    emit(prevClicked());
}

void VideoSink::positionSliderMoved(int vSliderValue)
{

    vSliderValue=setSliderOnClick(ui->horizontalSliderMediaPosition,vSliderValue);
    mTempTime = vSliderValue;
    int vSeconds = mTempTime % 60;
    mTempTime = mTempTime/60 ;
    int vMinutes = mTempTime % 60;
    int vHours = mTempTime/60;
    QTime vPosition(vHours,vMinutes,vSeconds);
    mGstDriver->setPosition(vPosition);
}

void VideoSink::setVolume(int vSliderValue)
{
    vSliderValue = setSliderOnClick(ui->horizontalSliderVolume,vSliderValue);

    emit(setVolumeAtMain(vSliderValue));
    ui->horizontalSliderVolume->setSliderPosition(vSliderValue);
}

int VideoSink::setSliderOnClick(QSlider * vQSlider , int vClickedPosition)
{
    Qt::MouseButtons vMouseButton = QApplication::mouseButtons();
    QPoint vMousePos = vQSlider->mapFromGlobal(QCursor::pos());
    bool isClicked = (vMouseButton &Qt::LeftButton) &&
            (vMousePos.x() >=0 && vMousePos.y() >=0 &&
             vMousePos.x() < vQSlider->size().width() &&
             vMousePos.y() < vQSlider->size().height());

    if(isClicked == true)
    {
        float vPosRatio = vMousePos.x() / (float)vQSlider->size().width();
        int vSliderRange = vQSlider->maximum()-vQSlider->minimum();
        int vSliderPositionUnderMouse = vQSlider->minimum() + vSliderRange *vPosRatio;

        if(vSliderPositionUnderMouse != vClickedPosition)
        {
            vQSlider->setValue(vSliderPositionUnderMouse);
            return vSliderPositionUnderMouse;
        }
    }
    return vClickedPosition;
}

void VideoSink::toggleMute()
{
    if(ui->toolButtonVolume->isChecked()==true)
    {
        mCurrentVolume=ui->horizontalSliderVolume->value();
```

```
        setVolume(0);
        ui->toolButtonVolume->setIcon(mVolumeMuted);
    }
    else
    {
        setVolume(mCurrentVolume);
        ui->toolButtonVolume->setIcon(mVolume);
    }
}


void VideoSink::toFullScreen()
{
    if(VideoSink::isFullScreen())
    {
        VideoSink::showNormal();
    }
    else
    {
        VideoSink::showFullScreen();
    }
}

void VideoSink::toLibraryMode()
{
    emit(setVideoMode(false));
    VideoSink::hide();
}

void VideoSink::setVolumeSlider(int vPosition)
{
    ui->horizontalSliderVolume->setSliderPosition(vPosition);
}

void VideoSink::setNowPlaying(QString vNowPlaying)
{
    ui->labelNowPlaying->setText(vNowPlaying);
    VideoSink::setWindowTitle("MediaTake :: " + vNowPlaying);
}

void VideoSink::showControls()
{
    ui->horizontalSliderMediaPosition->show();
    ui->horizontalSliderVolume->show();
    ui->labelLength->show();
    ui->labelNowPlaying->show();
    ui->labelTime->show();
    ui->toolButtonFullScreen->show();
    ui->toolButtonNext->show();
    ui->toolButtonPlayPause->show();
    ui->toolButtonPrevious->show();
    ui->toolButtonToMain->show();
    ui->toolButtonVolume->show();
    ui->widget->show();
}

bool VideoSink::eventFilter(QObject *obj, QEvent *event)
{
    if (event->type() == QEvent::HoverMove)
    {
//        QMouseEvent *mouseEvent = static_cast<QMouseEvent*>(event);
        showControls();

        mShowControlsTimer.start(3000); //re-hide controls after 3s
    }
    return false;
}

void VideoSink::hideControls()
```

```
{

    ui->horizontalSliderMediaPosition->hide();
    ui->horizontalSliderVolume->hide();
    ui->labelLength->setVisible(false);
    ui->labelNowPlaying->setVisible(false);
    ui->labelTime->setVisible(false);
    ui->toolButtonFullScreen->hide();
    ui->toolButtonNext->hide();
    ui->toolButtonPlayPause->hide();
    ui->toolButtonPrevious->hide();
    ui->toolButtonToMain->hide();
    ui->toolButtonVolume->hide();
    ui->widget->hide();
}
```