## 附

南京工程学院

**实 训 报 告**

课程名称 **多媒体编程基础**

实训项目名称 实训4： 动画及媒体播放

实训学生班级 数嵌 172

实训学生姓名 朱广锋

学　　　　号 202170638

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. **实训目的**   1.掌握 Qt Creator 的基本使用方法  2.理解定时器应用  3.理解媒体播放及显示的相关类  4.锻炼综合应用能力   1. **实训环境及开发工具：**   PC机、Qt5.14（或其它版本）   1. **实训要求及内容：**   1. 使用定时器(两种方法均可)设计一个**简单动画**，动画形式如：卡通、位移动画、数字时钟、表盘时钟等，功能不限。  2. 制作一个**简易媒体播放器**(音频、视频均可)，功能至少包含：打开文件、播放、停止、暂停、全屏及退出全屏(视频) 以及**一个**滑动调节（如：音量、进度、对比度、色饱和度等）。  3. 可以参考教材或提供的源代码，自己选择界面开发方法。   1. **程序设计思路（30分）** 2. 数字时钟，使用QImage保存数字和背景图像，并设置定时器来刷新绘图。 3. 视频播放器，使用来QMediaPlayer和QVideoWidget播放视频，为了支持更多视频格式，需要安装解码器。   1）、设置视频播放进度与进度条同步  2）、设置音量滑块与音量按钮同步  3）、设置亮度、对比度、色相、饱和度的信号  4）、设置播放、暂停、停止、快进、快退的播放控制  5）、设置全屏的进入和退出控制   1. **设计方法及代码（30分）** 2. 数字时钟   加载资源中的数码图像和背景图像，并对数码做拆分，设置定时任务。   |  | | --- | | QImage image;  image.load(":/DigitalClockX/res/digital.png");  background.load(":/DigitalClockX/res/bk.jpg");  digitalWidth = image.width() / 11;  digitalHeight = image.height();  for (size\_t i = 0; i < 11; i++)  digitalImages[i] = image.copy(digitalWidth \* i, 0, digitalWidth, digitalHeight);  setFixedSize(background.width(), background.height());  freshTimer = startTimer(100); |   覆写绘图事件，获取时间，绘制背景和数码   |  | | --- | | QPainter painter(this);  painter.setBackgroundMode(Qt::BGMode::TransparentMode);  QTime current\_time = QTime::currentTime();  painter.drawImage(0, 0, background);  paintClock(painter,current\_time.hour(),current\_time.minute(), current\_time.second()); |   数码绘制   |  | | --- | | void DigitalClockX::paintClock(QPainter& painter, int h, int m, int s)  {  painter.drawImage(digitalWidth \* 0, 0, digitalImages[h / 10]);  painter.drawImage(digitalWidth \* 1, 0, digitalImages[h % 10]);  painter.drawImage(digitalWidth \* 2, 0, digitalImages[10]);  painter.drawImage(digitalWidth \* 3, 0, digitalImages[m / 10]);  painter.drawImage(digitalWidth \* 4, 0, digitalImages[m % 10]);  painter.drawImage(digitalWidth \* 5, 0, digitalImages[10]);  painter.drawImage(digitalWidth \* 6, 0, digitalImages[s / 10]);  painter.drawImage(digitalWidth \* 7, 0, digitalImages[s % 10]);  } |  1. 视频播放器    * 1. 在Qt设计师中完成控件位置摆放和属性设置  * + 1. 在QVideoWidget派生类VideoControlWidget中覆写keyPressEvent方法，响应全屏时的Esc退出，覆写mouseDoubleClickEvent方法，在双击时实现暂停或者播放。使用双击的目的是防止误点击。  |  | | --- | | void VideoControlWidget::keyPressEvent(QKeyEvent\* event)  {  if (event->key() == Qt::Key\_Escape && isFullScreen())  {  setFullScreen(false);  event->accept();  QVideoWidget::keyPressEvent(event);  }  }  void VideoControlWidget::mouseDoubleClickEvent(QMouseEvent\* event)  {  if (event->button() == Qt::LeftButton)  {  if (mediaPlayer->state() == QMediaPlayer::PlayingState)  mediaPlayer->pause();  else  mediaPlayer->play();  }  QVideoWidget::mouseDoubleClickEvent(event);  } |  * + 1. 初始化并设置视频播放器的输出对象  |  | | --- | | player = new QMediaPlayer(this);  player->setNotifyInterval(10);  player->setVideoOutput(ui.videoWidget);  ui.videoWidget->setMediaPlayer(player); |  * + 1. 隐藏暂停按钮，修改 视频设置 窗口标题  |  | | --- | | ui.pauseBtn->hide();  optionDialog.setWindowTitle("视频设置"); |  * + 1. 设置进度条滑块与视频播放的同步  |  | | --- | | connect(player, &QMediaPlayer::stateChanged, [&](QMediaPlayer::State state)  {  ui.pauseBtn->setHidden(state == QMediaPlayer::PausedState || state == QMediaPlayer::StoppedState);  ui.playBtn->setHidden(state == QMediaPlayer::PlayingState);  ui.stopBtn->setEnabled(state != QMediaPlayer::StoppedState);  });  connect(player, &QMediaPlayer::durationChanged, [&](qint64 duration)  {  ui.positionSlider->setMaximum(duration);  int secs = duration / 1000;  int hour = secs / 3600;  int mins = (secs / 60) % 60;  secs %= 60;  durationTime = QString::asprintf("%02d:%02d:%02d", hour, mins, secs);  ui.positionLab->setText(positionTime + "/" + durationTime);  });  connect(player, &QMediaPlayer::positionChanged, [&](qint64 position)  {  if (ui.positionSlider->isSliderDown())  return;  ui.positionSlider->setSliderPosition(position);  int secs = position / 1000;  int hour = secs / 3600;  int mins = (secs / 60) % 60;  secs %= 60;  positionTime = QString::asprintf("%02d:%02d:%02d", hour, mins, secs);  ui.positionLab->setText(positionTime + "/" + durationTime);  });  connect(ui.positionSlider, &QSlider::valueChanged, [&](int value)  {  if (ui.positionSlider->isSliderDown())  player->setPosition(value);  }); |  * + 1. 设置音量滑块信号以及与音量按键的同步，在拖动音量为0时，按钮改变图像，按钮切换静音，则改变音量滑块位置。  |  | | --- | | connect(ui.volumeSlider, &QSlider::valueChanged, [&](int value)  {  if (!ui.volumeSlider->isSliderDown())  return;  lastVolume = value;  player->setVolume(lastVolume);  player->setMuted(lastVolume <= 0);  ui.volumeBtn->setIcon(QIcon(player->isMuted() ? ":/VideoPlayerX/res/静音.png" : ":/VideoPlayerX/res/喇叭.png"));  });  connect(ui.volumeBtn, &QPushButton::clicked, [&]  {  player->setMuted(!player->isMuted());  ui.volumeSlider->setValue(player->isMuted() ? 0 : lastVolume);  ui.volumeBtn->setIcon(QIcon(player->isMuted() ? ":/VideoPlayerX/res/静音.png" : ":/VideoPlayerX/res/喇叭.png"));  }); |  * + 1. 设置播放、暂停等按钮的逻辑，设置按钮隐藏切换等  |  | | --- | | connect(ui.openBtn, &QPushButton::clicked, [&]  {  QString aFile = QFileDialog::getOpenFileName(this, "打开视频", "", "视频文件(\*.wmv;\*.avi;\*.mp4;\*.mpeg;\*.mkv;\*.rmvb;\*.flv);;所有文件(\*.\*)");  if (aFile.isEmpty())  return;  QFileInfo fileInfo(aFile);  setWindowTitle("VideoPlayerX - " + fileInfo.fileName());  player->setMedia(QUrl::fromLocalFile(aFile));  player->play();  });  connect(ui.playBtn, &QPushButton::clicked, player, &QMediaPlayer::play);  connect(ui.pauseBtn, &QPushButton::clicked, player, &QMediaPlayer::pause);  connect(ui.stopBtn, &QPushButton::clicked, player, &QMediaPlayer::stop);  connect(ui.leftBtn, &QPushButton::clicked, [&]  {  auto new\_postion = player->position() - 2000;  if (new\_postion < 0)  new\_postion = 0;  player->setPosition(new\_postion);  });  connect(ui.rightBtn, &QPushButton::clicked, [&]  {  auto new\_postion = player->position() + 2000;  if (new\_postion > player->duration())  new\_postion = player->duration();  player->setPosition(new\_postion);  });  connect(ui.fullscreenBtn, &QPushButton::clicked, [&] {ui.videoWidget->setFullScreen(true); }); |  * + 1. 设置亮度、对比度、色相、对比度  |  | | --- | | connect(ui.optionButton, &QPushButton::clicked, &optionDialog, &OptionDialog::show);  connect(optionDialog.ui.brightnessSlider, &QSlider::valueChanged, ui.videoWidget, &QVideoWidget::setBrightness);  connect(optionDialog.ui.contrastSlider, &QSlider::valueChanged, ui.videoWidget, &QVideoWidget::setContrast);  connect(optionDialog.ui.hueSlider, &QSlider::valueChanged, ui.videoWidget, &QVideoWidget::setHue);  connect(optionDialog.ui.saturationSlider, &QSlider::valueChanged, ui.videoWidget, &QVideoWidget::setSaturation);  connect(optionDialog.ui.brightnessDefaultBtn, &QPushButton::clicked, [&] {optionDialog.ui.brightnessSlider->setValue(0); });  connect(optionDialog.ui.contrastDefaultBtn, &QPushButton::clicked, [&] {optionDialog.ui.contrastSlider->setValue(0); });  connect(optionDialog.ui.hueDefaultBtn, &QPushButton::clicked, [&] {optionDialog.ui.hueSlider->setValue(0); });  connect(optionDialog.ui.saturationDefaultBtn, &QPushButton::clicked, [&] {optionDialog.ui.saturationSlider->setValue(0); }); |   **六、实训结果及说明（30分）**  1. 数字时钟，显示结果如下图  2. 视频播放器  视频播放示例  视频属性设置示例  全屏播放  **七、实训思考（10分）**  曾遇到问题：   1. 设置进度条同步时遇到播放会出现声音卡顿的现象，卡顿频率刚好与setNotifyInterval的设置相同。   原因：响应positionSlider的valueChanged时 ，直接执行了player的setPosition方法，而这个方法又引发了positionSlider的valueChanged信号，导致无限调用（可能最终被Qt特殊处理，关闭了过长时间的调用），出现声音卡顿现象。  解决：positionSlider的valueChanged仅在鼠标按下时设置进度。  if (ui.positionSlider->isSliderDown())  player->setPosition(value);   1. 打开mp4、avi等格式的文件无法正常播放或无法播放   原因：缺少解码器  解决：安装了K-Lite解码器，多数视频可以正常播放。  未来拓展：   1. 视频播放器全屏界面增加一个可以隐藏的进度条，方便在全屏时调整进度； 2. 视频播放器除了按钮的快进快退，应该增加方向键左右来控制进度； 3. 视频播放器增加播放列表，使用类似PotPlayer的操作界面。 |