Qt Overlay on Video

NuMicro® 64/32位系列微控制器范例代码介绍

文件信息

应用简述	本范例程序说明如何编写 Qt 应用程序，此应用程序同时于屏幕底层播放影片并于上层覆盖互动组件，通过互动组件控制影片播放。
BSP 版本	Linux-5.10.x
开发平台	NuMaker-HMI-MA35D1-S1

1. 概述

混合显示是计算机图学中的一种技术，用于将多个图像或图层组合输出成单一图片，如图 1-1 所示，通常应用于影片与图像处理的应用程序，例如：影片合成。

本范例程序示范如何同时播放影片和运行 Qt 应用程序并显示于屏幕上，影片和 Qt 应用程序分别输出至不同的 Frame Buffer，如 Frame Buffer 0、Frame Buffer 1，并透过显示器的驱动程序提供的混合渲染功能进行混合，再将混合后的图显示于屏幕上。MA35D1 显示器驱动提供两种混合渲染功能：Color Key 与 Alpha Blending，此范例程序则是使用 Alpha Blending。

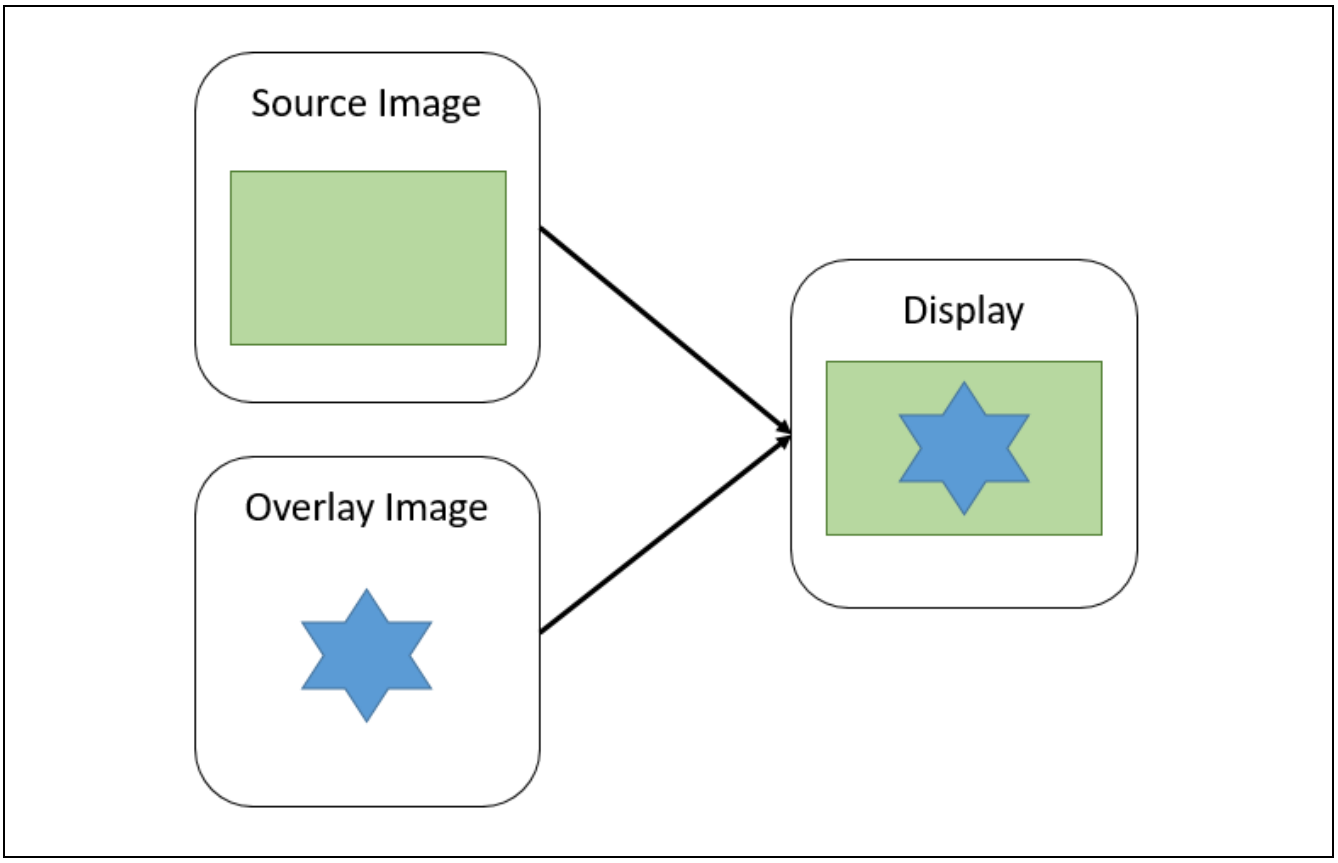

图 1-1 图像混合示意图

1.1 软件架构

软件实现架构如图 1-2 所示，Qt 应用程序使用 Gstreamer 函数库播放影片，Gstreamer 则会使用 VC8000 硬件译码器译码影片并直接输出至 Frame Buffer 0，Qt 使用 Qt 绘图函数库绘制需要显示在影片上的组件输出至 Frame Buffer 1，后续则交由显示控制器混合影片与组件。

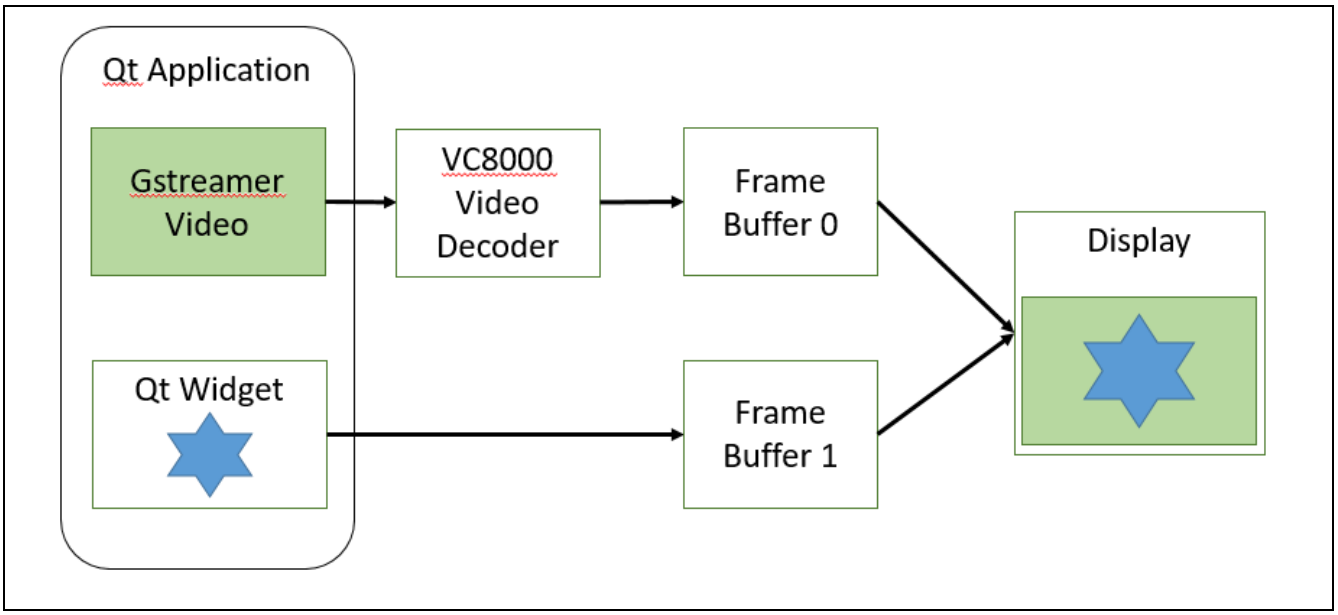

图 1-2 软件架构图

1.2 驱动设定

1.2.1 显示器设备树设定

确认显示器的分辨率与开启相关 overlay 选项，用户可根据自身屏幕分辨率调整显示驱动分辨率，以下使用分辨率 1024*600 为例。

```
/* Ensure overlay related setting */
&display {
    ....
    overlay-en = <1>; /* Enable overlay function */
    overlay-pixel-fmt = <6>; /* Set RGB888 format */
    overlay-width = <1024>; /* Set overlay resolution width */
    overlay-height = <600>; /* Set overlay resolution height */
    overlay-rect-tlx = <0>; /* Set the starting coordinate of the x-axis */
    overlay-rect-tly = <0>; /* Set the starting coordinate of the y-axis */
    ....
};
```

2. 用户程序介绍

此范例程序需使用 Qt Creator 来编辑与编译 Qt 应用程序，根据以下步骤操作：

1. Gstreamer 与 Qt 是一个跨平台的多媒体框架开发平台，其在 MA35D1 上的安装设定请参考 MA35D1 Buildroot Quick Start 的 Environment Setup 章节，Yocto 预设已经包含 Qt 与 Gstreamer，因此不用额外设定

2. 根据 MA35D1 HMI GUI Development Environment 里的 Qt Creator Cross-Compile Environment Setup 章节来建立 Qt Creator 环境

3. 打开 Qt Creator，选择 Open Project

4. 复制 EC_MA35D1_Qt_Overlay_on_Video_V1.00\Src\MA35D1_Qt_Overlay_on_Video 专案至 Linux 环境并开启项目，如图 2-1

5. 双击 MA35D1_Qt_Overlay_on_Video.pro 打开

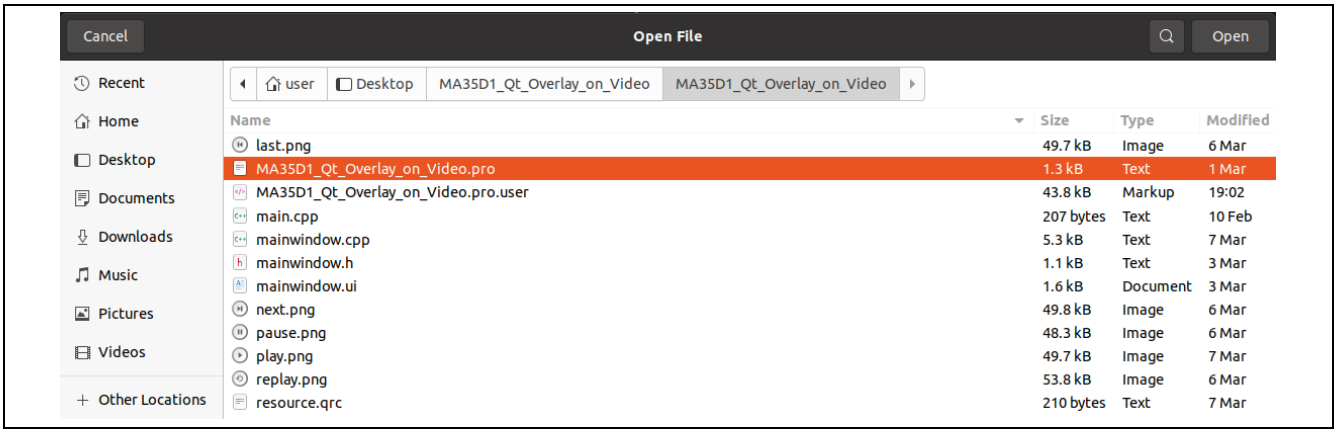

图 2-1 Qt 专案示意图

主要的 UI 控制行为都由 mainwindow.cpp 操控，此程序代码会将路径/opt/videos 里面的影片使用 Gstreamer 轮播，绘制 Qt 组件并输出。主要控制行为分为：暂停、回复、重新播放、上一个与下一个，按下对应的按钮即会触发相对应的动作。

```
#include <QApplication>
#include <QDebug>
#include <QDir>
#include <stdio.h>
#include <stdlib.h>
#include "mainwindow.h"
#include "ui_mainwindow.h"
```

```
#include <QTouchEvent>
#include <QThread>
#include <QTimer>
#include <gst/gst.h>

/* Define Videos location */
#define VIDEOS_FD_PATH "/opt/videos/"

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

    /* Gstreamer Setting */
    cur_video_idx = 0;
    QDir dir(QString::fromStdString(VIDEOS_FD_PATH));
    dir.setFilter(QDir::Files | QDir::Hidden | QDir::NoSymLinks);
    dir.setSorting(QDir::Size | QDir::Reversed);
    v_list = dir.entryInfoList();
    QString v_path = "gst-launch-1.0 filesrc location=" +
v_list.at(cur_video_idx).filePath() + " ! qtdemux name=demux demux.audio_0 ! queue !
decodebin ! audioconvert ! audioresample ! autoaudiosink demux.video_0 ! queue !
decodebin ! nufbdevsink ! fakesink sync=true ";
    gst_init (NULL, NULL);
    pipeline = gst_parse_launch (v_path.toStdString().c_str(), NULL);

    /* Set Icon */
    ui->play_button->setStyleSheet("QPushButton{ border-style: none; border-image:
url(:/pause.png); background-color: transparent;}");
    ui->replay_button->setStyleSheet("QPushButton{ border-style: none; border-image:
url(:/replay.png); background-color: transparent;background-position: center;}");
    ui->next_button->setStyleSheet("QPushButton{ border-style: none; border-image:
url(:/next.png); background-color: transparent;}");
    ui->last_button->setStyleSheet("QPushButton{ border-style: none; border-image:
url(:/last.png); background-color: transparent;}");

    /* This timer is used to auto replay the video */
    thread = new QThread(this);
    timer = new QTimer(0);
    timer->setInterval(1000);
    timer->moveToThread(thread);
    connect(timer, SIGNAL(timeout()), this, SLOT(video_update()),Qt::DirectConnection);
    connect(thread, SIGNAL(started()), timer,SLOT(start()));
    thread->start();

    /* Transparent Background */
    this->setAttribute(Qt::WA_TranslucentBackground, true);

    /* Start to Play Video */
    gst_element_set_state (pipeline, GST_STATE_PLAYING);
    playing = true;
}

MainWindow::~MainWindow()
{
    delete ui;
```

```
}

void MainWindow::video_update()
{
    bus = gst_element_get_bus(pipeline);
    msg = gst_bus_timed_pop_filtered (bus,
GST_CLOCK_TIME_NONE,(GstMessageType)(GST_MESSAGE_ERROR | GST_MESSAGE_EOS |
GST_MESSAGE_UNKNOWN ));
    switch (GST_MESSAGE_TYPE (msg)) {
    case GST_MESSAGE_ERROR:
        qDebug() << "Stream Error";
        next_video();
        break;

    case GST_MESSAGE_EOS:
        qDebug() << "End-of-Stream";
        gst_element_set_state (pipeline, GST_STATE_NULL);
        gst_object_unref (pipeline);
        gst_message_unref (msg);
        gst_object_unref (bus);
        next_video();
        break;

    default:
        break;
    }

}

/* Play Video Action */

void MainWindow::resume_video()
{
    gst_element_set_state (pipeline, GST_STATE_PLAYING);
}

void MainWindow::pause_video()
{
    gst_element_set_state (pipeline, GST_STATE_PAUSED);
}

void MainWindow::replay_video()
{
    if(v_list.size() == 0)
        return;
    cur_video_idx--;
    cur_video_idx = cur_video_idx < 0 ? v_list.size()-1 : cur_video_idx;
    gst_element_post_message (pipeline, gst_message_new_eos(GST_OBJECT(pipeline)));
}

void MainWindow::next_video()
{
    if(v_list.size() == 0)
        return;
    cur_video_idx++;
    cur_video_idx = (cur_video_idx)%v_list.size();
```

```
    play_video();
}

void MainWindow::last_video()
{
    if(v_list.size() == 0)
        return;
    cur_video_idx-=2;
    if(cur_video_idx <= -2)
        cur_video_idx = v_list.size()-2;
    gst_element_post_message (pipeline, gst_message_new_eos(GST_OBJECT(pipeline)));
}

void MainWindow::play_video()
{
    gst_init (NULL, NULL);
    QString v_path = "gst-launch-1.0 filesrc location=" +
v_list.at(cur_video_idx).filePath() + " ! qtdemux name=demux demux.audio_0 ! queue !
decodebin ! audioconvert ! audioresample ! autoaudiosink demux.video_0 ! queue !
decodebin ! nufbdevsink ! fakesink sync=true ";
    pipeline = gst_parse_launch (v_path.toStdString().c_str(), NULL);
    gst_element_set_state (pipeline, GST_STATE_PLAYING);
}


void MainWindow::on_play_button_clicked()
{

    if(playing == true)
    {
        qDebug() << "Pause";
        pause_video();
        ui->play_button->setStyleSheet("QPushButton{ border-style: none; border-image:
url(:/play.png); background-color: transparent;}");
    }
    else
    {
        qDebug() << "Resume";
        resume_video();
        ui->play_button->setStyleSheet("QPushButton{ border-style: none; border-image:
url(:/pause.png); background-color: transparent;}");
    }
    playing = !playing;
}

void MainWindow::on_replay_button_clicked()
{
    qDebug() << "Replay";
    replay_video();
}

void MainWindow::on_next_button_clicked()
{
    qDebug() << "Next";
    gst_element_post_message (pipeline, gst_message_new_eos(GST_OBJECT(pipeline)));
}
```

```
void MainWindow::on_last_button_clicked()
{
    qDebug() << "Last";
    last_video();
}
```

图 2-2 为 Qt 项目中的 UI 设计图，统一将组件放置于屏幕下方，全部大小为 1024*600，每个按钮大小为 100*100。

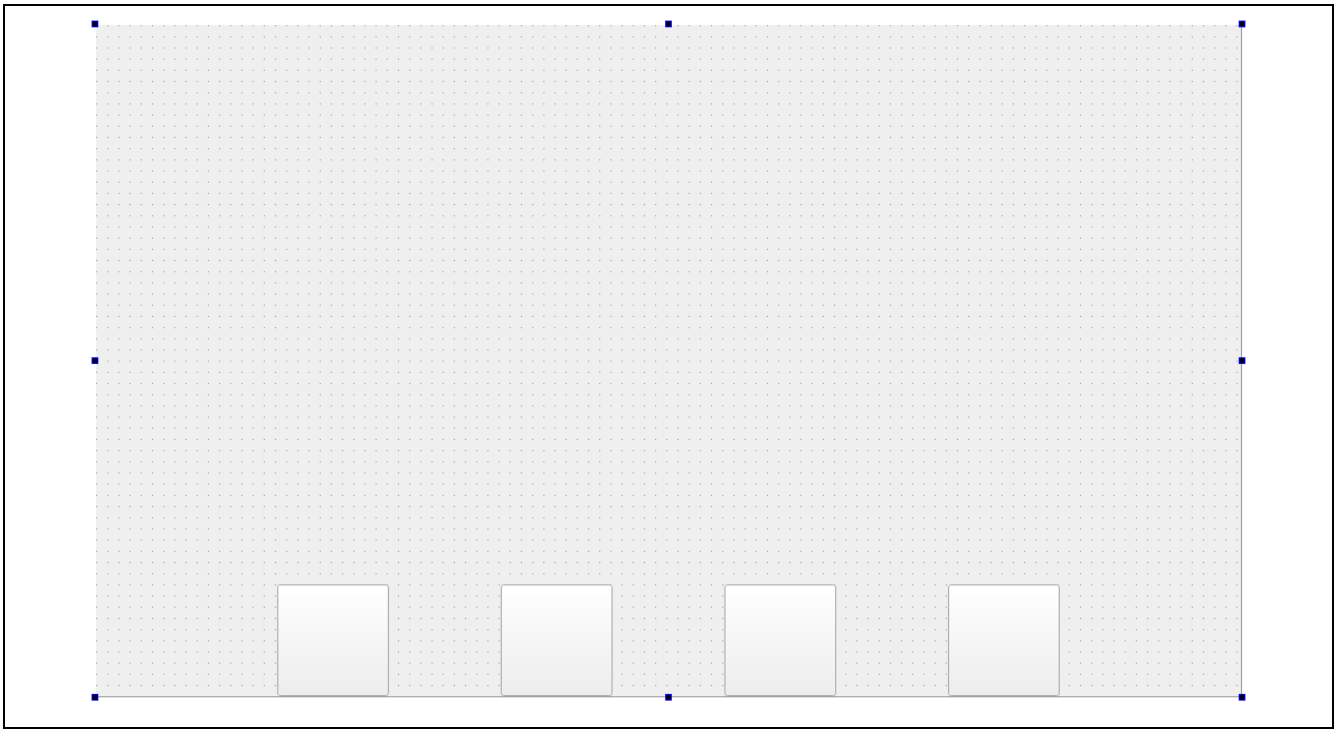

图 2-2 mainwindow.ui 规划图

3. 软件与硬件需求

3.1 软件需求

- BSP 版本
  - Linux-5.10.x

3.2 硬件需求

- 电路组件
  - NuMaker-HMI-MA35D1-S1

4. 目录信息

EC_MA35D1_Qt_Overlay_on_Video_V1.00	
Src	Source files of Qt project
DT-Binding	Device tree requirements on the contents of driver

图 4-1 目录信息

5. 范例程序执行

1. 设定显示器设备树(请参照 1.2.1 显示器设备树设定)
2. 重新编译 MA35D1 Image 并烧录
3. MA35D1 开机后，参考 MA35D1 HMI GUI Development Environment 来设置 Qt 与 tslib 环境变量
4. 复制 MA35D1_Qt_Overlay_on_Video 应用程序到开发板
5. 复制影片至 /opt/videos/
6. 执行以下命令

```
./MA35D1_Qt_Overlay_on_Video -platform linuxfb:fb=/dev/fb1
```

7. 执行后画面如图 5-1

图 5-1 MA35D1_Qt_Overlay_on_Video 执行画面

6. 修订纪录

Date	Revision	Description
2024.02.16	1.00	初始版本。

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.