基于 Cairo 的组合式仪表 HMI 系统设计

郭健忠[1],耿　屹[1],谢　斌[2],闵　锐[2]
(1. 武汉科技大学 汽车与交通工程学院,湖北 武汉 430065;
2. 武汉保华显示科技有限公司,湖北 武汉 430081)

摘　要:应用于中低端汽车市场的组合式汽车仪表一般采用低成本的系统设计方案,因此在人机交互界面(HMI)显示效果、信息呈现形式等方面与全液晶仪表存在一定差距。为使组合式仪表 HMI 系统能在保持低成本的同时具备丰富的信息呈现形式和更佳的显示效果,本设计在硬件选型上采用高性价比的全志 F1c100s 作为嵌入式 HMI 平台的微处理单元,在界面设计上优化了汽车 HMI 图层、场景结构与资源文件的封装方式,加快了软件对资源的加载速度从而间接提升画面帧率;在开发方式上,搭建了基于 Cairo 图形库的跨平台开发环境,简化了开发流程。HMI 系统软件部分包含数据解析、事件队列、XML 文件解析和图形抽象四个模块,以上模块在虚拟平台上完成开发、调试与优化后,移植到嵌入式平台运行验证。测试结果表明,该方案能以较低的软硬件成本实现可靠的功能性及更佳的显示效果。

关键词:Cairo;跨平台;人机交互界面;组合式汽车仪表;F1c100s
中图分类号:TP311　　**文献标识码**:A　　**文章编号**:1673-629X(2021)10-0196-07
doi:10.3969/j.issn.1673-629X.2021.10.033

Design of Composite Instrument HMI System Based on Cairo

GUO Jian-zhong[1], GENG Yi[1], XIE Bin[2], MIN Rui[2]
(1. School of Automobile and Traffic Engineering, Wuhan University of Science and Technology, Wuhan 430065, China;
2. Wuhan Baohua Display Technology Co., Ltd., Wuhan 430081, China)

Abstract: The low-cost system design scheme is generally adopted for the composite vehicle instrument applied in the middle and low end automobile market. Therefore, there is a certain gap in HMI display effect, information presentation form and other aspects with the full liquid crystal instrument. In order to make the instrument HMI system achieve a better display effect at a lower cost, this design adopts the low-cost Allwinner F1c100s as the micro-processing unit of the embedded HMI platform. In terms of interface design, the encapsulation mode of automobile HMI layer, scene structure and resource file is optimized, and the loading speed of software to resources is accelerated, thus indirectly improving the frame rate of pictures. In terms of development mode, a cross-platform development environment based on Cairo graphics library is built, which simplifies the development process. HMI program been designed for four modules: data parsing, event queue, XML file parsing and graphic abstraction. After development, debugging and optimization on the virtual platform, the modules are transplanted to the embedded platform to run verification. The test shows that the scheme can achieve reliable function and better display effect with lower hardware and software cost.

Key words: Cairo; cross platform; human machine interface; composite vehicle instrument; F1c100s

0　引　言

随着智能网联汽车、高级辅助驾驶等技术的发展,数字仪表作为车况信息显示终端被广泛应用于汽车电子系统中[1-2]。作为数字仪表的一种,组合式仪表由低成本微处理单元(MPU)、5 至 8 英寸液晶显示屏、机械指针或断码屏组合而成,相较于全液晶仪表,其显示效果与信息呈现形式存在一定差距[3]。人机交互界面(HMI)是汽车数字仪表的核心功能之一,承担驾驶员与汽车信息交互的功能[4-5]。传统 HMI 软件通常借助 MiniGUI[6-7]、Qt/Embedded[8]等嵌入式图形用户界面(GUI)框架开发,应用于窗口化而非全屏模式的应用场景,UI 界面响应鼠标、键盘等输入信号,未针对汽车 HMI 以总线信号为触发源的信号环境进行优化。在移植方式上,传统框架需要将自身框架源码跨平台编

收稿日期:2020-11-16　　**修回日期**:2021-03-16
基金项目:湖北省自然科学基金项目(2019CFB694)
作者简介:郭健忠(1969-),男,副教授,研究方向为汽车电子与信息技术;耿　屹(1995-),男,硕士,研究方向为汽车液晶仪表技术及开发。

译至目标平台,移植方式较为繁琐[9]。Cairo 图形库具备跨平台、轻量化、可定制性强等特性[10],本设计基于 Cairo 图形库构建虚拟 HMI 平台和嵌入式平台,创新跨平台开发方式,简化开发与移植流程,针对低成本嵌入式平台,从多维度设计并优化 HMI 软件,提高软硬件运行效率,以更低成本实现可靠的功能性及更佳的显示效果。

1　汽车仪表 HMI 设计与优化

1.1　HMI 图层与场景

汽车仪表 HMI 一般由图 1 左侧所示的背景图层、仪表盘指针图层、状态指示灯图层和报警/菜单显示图层组成,由于图层结构相对固定,图形库渲染时按照从下至上的顺序对图层进行堆叠渲染。图层由如图 1 右侧所示的一个或多个场景组成,场景中包含图片、文字以及动画效果等元素,场景中包含的元素内容、显示内容及触发方式均与总线数据逐一对应,例如仪表盘指针图层与背景图层根据总线信号中关于当前驾驶模式的数据,分别显示 ECO、SPORT 和标准模式场景以提供不同的主题色彩和显示效果;报警/菜单图层则根据

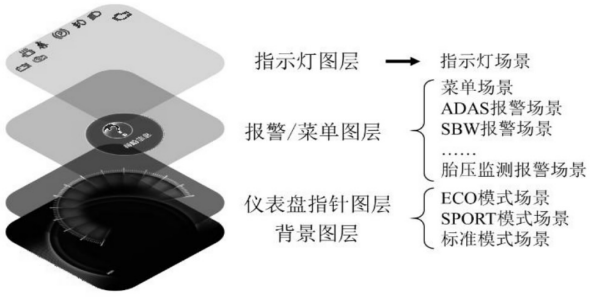

图 1　汽车仪表 HMI 图层及场景示意图

总线信号中当前菜单的索引数据和报警数据触发相应的菜单或报警场景。

1.2　场景设计与导出

HMI 图层与场景采用 Fairy GUI 软件进行设计和导出,Fairy GUI 是一款用于软件或游戏的 UI 设计软件,可提供跨平台 UI 设计解决方案[11]。该软件支持在工程设计树中以包和组件的方式管理 HMI 图层和场景,可在场景中导入界面元素,为每个场景内元素设定显示坐标、旋转角度、旋转锚点、缩放比例、透明度等信息。设计完成后可通过软件中的发布功能生成包含当前图层和场景布局信息的 XML 文件,用于后期图形渲染。为简化设计流程,HMI 中静态文字、按钮等控件元素均使用支持透明背景(支持 Alpha 通道)的 PNG 图片资源,动态文字通过预留空位,在程序中调用 Cairo 图形库中的矢量字库接口绘制文字。

1.3　场景元素封装

在嵌入式 GUI 框架中,图片资源通过资源文件路径加载,而资源文件通常储存在存储器不同扇区中,此方式会增加文件随机读写次数,不适用于图片资源较多且读写速度有限的 Flash 存储器上。汽车 HMI 单一场景内的元素内容数量较少、重复性小且元素搭配相对固定,因此以场景为单位,封装场景内的元素为单个二进制(bin)文件,将多次随机读写转换为单次顺序读写,可有效提高场景资源文件加载效率,封装后的二进制文件格式及相关描述如表 1 所示。为进一步提升 Flash 存储器的读写效率,文件中数据以四字节对齐的方式排列,元素数据域内图片数据按照设计的堆叠顺序依次收尾相连,数据加载时通过文件头中元素地址索引定位。

表 1　场景元素封装二进制文件格式

名称	起始地址	大小/Byte	说明
文件头(0xAA)	0x00	1	指示 bin 文件类型
文件索引号	0x01	2	指示当前文件对应的场景
场景元素数量	0x03	2	指示文件中包含元素数量
默认元素序号	0x05	2	指示默认情况下绘制元素序号
元素地址	0x08	4	用于在元素数据域中定位元素数据
CRC 校验值	0x08+4 * n	2	校验文件头数据
文件头截止位	4 * n+0x0A	2	指示文件头结束语数据域起始
元素数据域	4 * n+0x0C	--	封装元素的数据域内容

2　双平台图形库环境搭建

Cairo 是基于 C 语言编写的跨平台 2D 图形库,支持 Windows、Linux 等主流操作系统和部分嵌入式平台,支持将渲染好的像素数据导出为多种格式的文件或与各平台图形底层结合显示图像,其图形应用程序接口(API)和画面输出效果在任意平台下均保持一致,拥有优秀的跨平台开发特性[12]。基于 Cairo 图形库搭建虚拟 HMI 开发平台和嵌入式平台,可确保双平台下 HMI 软件代码的一致性,便于后期程序跨平台

移植。

2.1 虚拟 HMI 平台环境搭建

虚拟 HMI 平台本质是在 Windows 操作系统环境下运行的窗口应用程序,其编译与运行分别依赖静态库 cairo. lib 和动态库 cairo. dll,上述库文件可从 Gtk+ 框架中提取或通过编译图形库源码获得[13-14]。由于 Gtk+中的 Cairo 库文件版本较低,对 Win32 平台支持性较差,而通过编译最新版本源码生成的库文件具有良好的兼容性并支持完整的新特性。Cairo 源码编译依赖 PNG 图片解码库 libpng 和像素处理函数库 pixman,而 libpng 库依赖 zlib 函数库提供对 PNG 图片数据压缩与解压的支持[15],因此编译顺序如图 2 所示。编译完成后,将生成的静态库文件添加到工程链接器配置属性中的附加依赖项中,将动态库文件移动至生成的程序执行文件根目录下,即可完成虚拟 HMI 开发平台运行环境搭建。

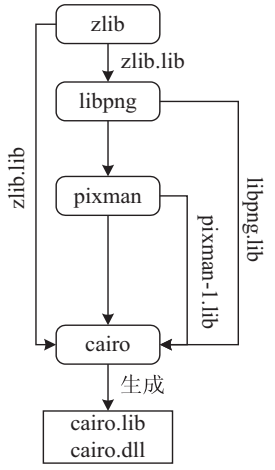

图 2 Cairo 图形库编译顺序

2.2 嵌入式平台

目前市场主流全液晶仪表采用恩智浦 i. MX 系列[16-17]、瑞芯微 RK 系列等高性能处理器,此类处理器硬件采购成本、电路设计及电路板印制成本较高,无法满足组合式仪表对成本的限制需求。普通 MCU 由于性能较弱无法兼顾数据处理与画面显示,采用双芯片设计能有效均衡算力从而降低成本[18]。文中采用如图 3 所示的 S32K144 MCU 处理总线数据及 IO 信号,F1c100s MPU 用于画面显示,二者通过高速串口进行数据交互。

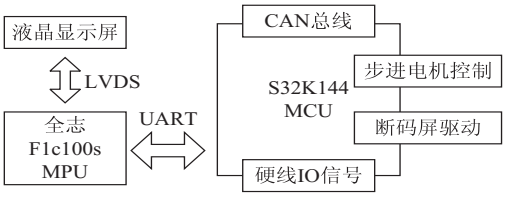

图 3 双芯片架构示意图

其中全志 F1c100s 是基于 ARM9 架构设计默认主频为 400 MHz 的微处理器,集成 DDR 内存,可运行 XBOOT、精简版 Linux 等操作系统。

XBOOT 是一款功能强大、可移植性强、代码复用率高的嵌入式系统 Bootloader,同时也是片上系统应用程序执行引擎,可为应用程序提供启动及运行环境。由于 XBOOT 系统具备体积小,功能完善的特点,且内置 Cairo 图形库与文件系统,无需用户额外配置,适合跨平台轻量化 HMI 开发。XBOOT 源码可从开源项目网站上获取,解压后需编辑源码目录中的 Makefile 文件,添加编译选项,指定目标编译平台和交叉编译器:

```
CROSS_COMPILE? =arm-eabi-
PLATFORM ? =arm32-f1c100s
```

再将源码工程添加至集成 CDT 插件的 Ecllipse IDE 中编译,将生成的二进制 bin 文件通过 sunxi_tools 烧写工具烧写至板载闪存中运行。

3 HMI 程序设计

由于跨平台软件开发存在编译速度慢、资源烧写过程繁琐甚至无法调试等问题,极大影响软件开发速度。虚拟 HMI 平台依托 Visual Studio 集成开发环境(IDE),具备完善的编译与调试环境,无需烧写资源,使 HMI 软件开发流程更为简化,从而提高开发效率。

3.1 虚拟 HMI 平台架构

虚拟 HMI 平台由图 4 所示的五个部分组成:(1)窗口程序主框架;(2)虚拟总线数据模块;(3)虚拟按键模块;(4)数据缓冲区;(5)HMI 模块。窗口程序主框架负责初始化程序各模块,创建可视化窗口,监听响应系统消息和用户操作和任务调度;虚拟总线数据模块与虚拟按键模块为非模态对话框子程序,负责转换用户鼠标操作为虚拟总线数据并发送至数据缓冲区,HMI 模块扫描缓冲区中的数据进行实时画面显示。

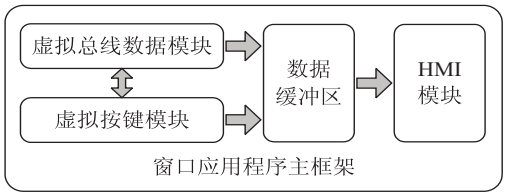

图 4 虚拟 HMI 平台架构

3.2 HMI 模块

HMI 模块包含数据解析、事件队列、XML 文件解析和图形抽象四个单元,模块的数据接口与平台类型无关,图形抽象接口可根据当前运行环境适配,因此在虚拟 HMI 平台中构建的 HMI 模块可整体移植至目标嵌入式平台,使移植步骤更简化。

3.2.1 HMI 模块工作流程

数据解析单元负责扫描缓冲区中数据的异动情况,根据当前数据异动情况向事件队列单元发送相应

事件,事件是一个包含当前事件信息的结构体 DisplayEvent:

```
typedef struct __DisplayEvent
{
scr_id_t ScrId;  //场景 ID,与二进制文件头中 ID 对应
scr_time_t ScrDisplayTime;  //场景显示时长
priority_t ePriority;  //事件优先级
bool isInterrupt;  //是否能被打断
bool isRedisplay;  //打断后是否重新显示
payload_t ePayload;  //当前场景数据负载
}DisplayEvent;
```

事件队列单元中的事件按照优先级从高到底以链表形式排列,事件插入函数依据传入结构体中的 ePriority 值向链表中插入事件,事件响应函数优先响应链表头部事件,再调用 XML 解析单元和图形抽象单元绘制场景元素。XML 文件解析依赖 libxml2 库,场景布局数据以键值对的形式存储在 XML 文件中,解析时将键值对数据逐一拷贝至 PngInfo 结构体数组中相应的位置,解析流程如图 5 所示。

```
typedef struct __PngInfo
{
uint32_t pIndex;  //场景元素编号
float x;  //x 轴坐标
float y;  //y 轴坐标
float scale_x;  //x 轴缩放比
float scale_y;  //y 轴缩放比
int rotation;  //旋转角度
float rot_x;  //旋转锚点 x 轴坐标
float rot_y;  //旋转锚点 y 轴坐标
uint8_t * pPngFileData;  //文件数据指针
uint8_t * pPngRawData;  //元素数据指针
cairo_surface_t PNGSurface;  //图形库 Surface 指针
} PngInfo;
```

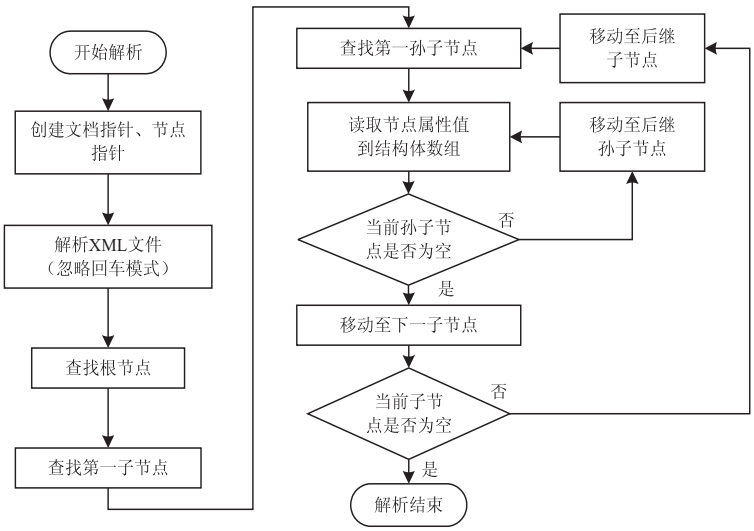

图 5　XML 解析单元流程

3.2.2　图形抽象单元

图形抽象单元负责场景元素加载、解析和渲染,虚拟 HMI 平台中的图形抽象单元基于 Windows 系统下的图形设备接口(GDI)与 Cairo 图形库搭建,由于 Cairo 图形库无法直接在应用程序窗口上绘制图形,依赖 GDI 为 Cairo 图形库提供设备上下文环境(device context,DC)。使用 GetDC()函数可基于当前窗口创建 DC,若直接在窗口 DC 上渲染图形会造成画面撕裂和闪烁,使用双重缓冲区可解决上述问题。调用 CreateCompatibleDC()和 CreateCompatibleBitmap()函数在内存中分别创建基于窗口 DC 的兼容 DC 和兼容位图数据缓冲区,待 Cairo 图形库绘制完成后,使用 BitBlt()函数将兼容 DC 中的像素数据逐字节拷贝至窗口 DC,可在窗口中实现稳定的画面显示。

Cairo 图形库为 GDI 提供了 cairo_win32_surface_create()接口,该接口可将兼容 DC 转换成类型为 cairo_surface_t 的抽象绘制平面(Surface),后续图形均在此上下文绘制。图形绘制流程由资源加载、图形变换、图形绘制、资源销毁四个阶段组成:

阶段一,资源加载:图片资源从对应场景的 bin 文件中加载,一种方式是通过 libpng 库将 bin 文件数据域中的图片数据解析为像素数据,传入 Cairo 图形库中的 cairo_image_surface_create_for_data()接口创建类型为 cairo_t 的上下文目标。第二种方式无需显式地让 libpng 库参与图片加载过程,而是向 Cairo 图形库中的 cairo_image_surface_create_from_png_stream()接口传入事先读取好的图片数据创建上下文目标。以上两种方案均需重载用于读取 PNG 文件的回调函数,并将函数指针传递至 png_set_read_fn()接口调用。

阶段二,图形变换:汽车仪表 HMI 中涉及到的基本变换操作包含平移、缩放和旋转,变换参数已由 XML 解析单元解析至 PngInfo 结构体中,为简化变换

流程,将所需的变换步骤封装为如下所示单一函数中:

```
void Paint_PNG_Element(cairo_t * cr, PngInfo * info)
{
cairo_save(cr);
if (!(info->rotation == 0))  //旋转
{
cairo_translate(cr, info->rot_x, info->rot_y);
cairo_rotate(cr, info->rotation);
cairo_translate(cr, -info->rot_x, -info->rot_y);
}
if (!(info->scale_x == 1.0&&info->scale_y == 1.0))
//缩放并平移
{
cairo_scale(cr, info->scale_x, info->scale_y);
cairo_set_source_surface(cr, info->PNGSurface,
(int)info->x / info->scale_x,
(int)info->y / info->scale_y);
}
else  //仅平移
{
cairo_set_source_surface(cr, info->PNGSurface,info->x,
info->y);
}
}
```

阶段三,图形绘制:图形的加载和变换将图片位图数据置于内存中处理,并未与最初创建的 Cairo 抽象绘制平面绑定,因此可使用 cairo_paint()或 cairo_paint_with_alph()函数将内存中的数据渲染在上下文上,后者支持透明背景渲染。cairo_paint()不仅可以绘制图片,还可以绘制实时矢量图形和矢量文字,增加了 HMI 软件设计的灵活性。

阶段四,资源销毁:受嵌入式平台内存或显存因素的制约,当场景内的某一元素不再显示或发生场景切换等情况,需及时销毁当前资源或场景在内存中的数据,释放内存。Cairo 图形库提供了 cairo_surface_destroy()和 cairo_destroy()接口,分别用于销毁内存中的图片数据和抽象绘制平面。

4 跨平台移植

由于 XBOOT 系统为嵌入式平台提供了包含 Cairo 图形库在内的完整的运行环境,虚拟 HMI 平台与嵌入式平台图形环境基本一致,不同之处在于数据解析单元的总线数据来源、图形抽象层与设备底层之间的接口以及部分设备驱动接口。因此移植过程更为简化,仅需将虚拟 HMI 平台中的代码进行部分修改后拷贝至嵌入式平台工程中编译即可,移植详情如表 2 所示。

表 2 HMI 模块各单元移植方式

虚拟 HMI 模块单元名称	移植方式
数据解析单元	修改数据来源至 UART 缓冲区
XML 解析单元	无需修改
事件队列单元	无需修改
图形抽象单元	注入资源文件并修改设备上下文环境及图形库相关接口
其他	增加定时器接口统计 FPS 信息

首先要向工程中注入资源文件,将各场景的 bin 文件及 xml 文件分别拷贝至./xboot/src/romdisk/framework/asset/scr 和./xboot/src/romdisk/framework/asset/cfg 目录下,此目录在编译阶段会自动生成虚拟文件系统,便于程序通过文件路径调用文件。由于 XBOOT 内存结构及空间申请方式与 Linux 系统类似,采用双向链表的形式管理,设备上下文环境需要调用 list_for_each_entry_safe 函数在链表中寻找足够的内存空间,并在此内存空间上创建 Cairo 图形库的抽象绘制平面作为上下文,具体代码如下,后续绘图阶段代码则与虚拟 HMI 平台保持一致。

```
cairo_surface_t * cs;
cairo_t * cr;
struct framebuffer_t * fb;
list_for_each_entry_safe(pos, n, &__device_head[DEVICE_
TYPE_FRAMEBUFFER], head)
fb = (struct framebuffer_t *)(pos->priv))
cs = cairo_xboot_surface_create(fb);
cr = cairo_create(cs);
```

Cairo 图形库渲染后的像素数据关联在 cs 指针中,还需通过 cairo_xboot_surface_present()函数将数据传递至硬件底层,交由 LVDS 驱动传输至显示屏显示。

5 测试验证与软件优化

5.1 测试与试验验证

本节采用虚拟 HMI 平台与嵌入式平台联合实验的方式验证 HMI 软件的可行性与可靠性。首先测试 HMI 软件对总线信号响应的准确度,由于双平台中总线数据解析、事件队列及 XML 解析单元中代码一致,因此可在虚拟 HMI 平台中测试验证。根据如表 3 所

示的汽车仪表 HMI 软件测试用例对 HMI 软件中指示灯显示状态、优先级打断逻辑、报警场景触发准确性等方面进行全功能测试,结果表明,仪表能准确响应总线信号并显示正确的提示信息,具备可靠的功能性。

表 3　汽车仪表 HMI 软件测试用例部分测试项目

测试编号	类型	用例名称	操作流程/期望结果	实际结果	测试结果
001-001	指示灯	驻车符号指示	勾选“驻车”,指示灯亮,反之,熄灭	与期望相同	PASS
001-002	指示灯	总电源指示	勾选“总电源”,指示灯亮,反之,熄灭	与期望相同	PASS
001-003	指示灯	气压报警指示	勾选“气压”,指示灯亮,反之,熄灭	与期望相同	PASS
001-035	仪表盘	实时车速	发送车速 85km/h,显示数值 85	显示数值 85,指针位置 85	PASS
001-052	报警	胎压报警	勾选“胎压”,显示胎压报警文字及报警图片	与期望相同	PASS
001-055	报警	报警优先级	同时勾选“胎压”、“门开”,门开报警优先级更高,显示门开报警文字及图片。	与期望相同	PASS

软件性能方面,HMI 软件在嵌入式设备上的每秒显示帧数(fps)是画面流畅度的重要评价指标之一,fps 值受当前设备内存、FLASH 读写速度与 MCU 算力的制约,在汽车仪表 HMI 软件中,场景资源的加载、解析及图形变换对 fps 值影响较大。为测试上述问题对 fps 值的影响,本节使用包含 120 张分辨率为 480 * 800 动画序列图片的场景作为测试目标,在虚拟 HMI 中平台测试,目的是记录单帧渲染时长和内存占用情况。对照组采用以文件路径作为形参调用图形库接口对图片逐一加载的方式,方式一和方式二为 3.2.2 节中提出的两种方式,方式三则是先将场景资源全部解析到内存中,再调用图形库从内存中读取数据,此方式对内存占用较大,不适用嵌入式平台,仅供参考。测试结果如图 6 和表 4 所示,方式二相比于传统方式平均单帧渲染时间降低 4 ms,平均帧率提升 5 fps,而方式三由于渲染时无需加载图片资源,其单帧渲染时间仅为 1 ms,但内存占用为常规方式的 5.6 倍,上述结果说明方式二为更加合理的图片资源加载方式。

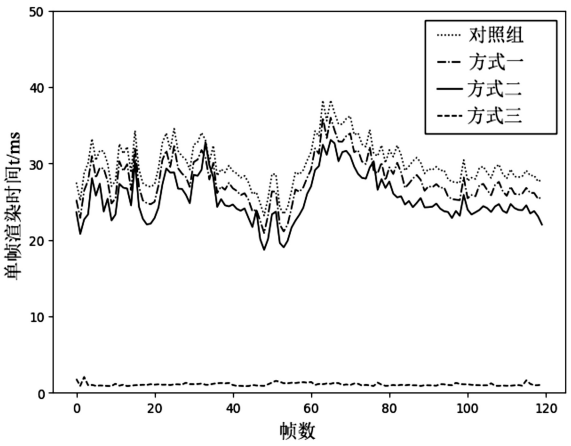

图 6　单帧渲染时间测试

表 4　测试详情

方式	平均单帧渲染时间/ms	平均帧率/fps	虚拟 HMI 平台内存占用情况/MB
对照组	29.99	33.40	57.2
方式一	27.72	36.07	59.4
方式二	25.38	39.40	56.3
方式三	1.08	>60	324.5

5.2　软硬件联合优化

软件方面,方式三单帧渲染时间极低,但需要图片数据常驻内存,因此可应用于变化不大且需长期显示的场景,例如背景、仪表盘及指针所处的模式场景,其他场景则继续采用方式二加载。经测试,嵌入式平台 HMI 软件运行在 480 * 800 分辨率的主界面下,混合加载模式相较于仅采用方式二的模式,平均帧率由 27.2 fps 提升至 35.7 fps,提升幅度为 31.25%。

硬件方面,全志 F1c100s 主频默认 400 MHz,通过修改时钟配置文件,可将主频提升至 720 MHz 从而提升帧率。由于 Flash 闪存在 HMI 软件运行时为只读状态,将 SPI 驱动变更为双线只读模式,能提升一定的读

取速率。经测试验证,平均帧率由软件优化后的 35.7 fps 提升至 42.7 fps,提升幅度为 19.6%,软硬件联合优化后的实时帧率满足汽车组合仪表 HMI 流畅显示的设计需求。

6 结束语

介绍了面向低成本组合式汽车仪表的 HMI 系统设计,提出了以图层与场景为单位的界面元素结构,搭建了基于 Cairo 图形库的虚拟 HMI 平台和嵌入式平台,简化了界面设计、软件编译及调试的繁琐步骤,同时双平台图形环境的一致性使开发效率得以提升,从而降低了软件的研发成本。HMI 程序设计针对汽车信号环境进行了优化,通过监听数据缓冲区中的数据异动情况,根据当前触发事件的优先级响应绘图事件,调用 XML 解析单元和图形抽象单元完成画面绘制。试验表明,组合式仪表能在保证可靠功能性的同时,通过虚拟 HMI 平台的仿真实验和软硬件的联合优化,大幅提升 HMI 实时帧率,保证了仪表显示的流畅性,达成了以低成本实现更佳显示效果的设计目的。

参考文献:

[1] 于 琦. 现代汽车仪表技术及发展趋向研究[J]. 汽车与驾驶维修:维修版,2018(3):107.

[2] HEECHON K,KYUNGHA K. Development of vehicle HMI module using model-based design and RCP[J]. SAE International Journal of Passenger Cars Mechanical Systems,2009,2(1):1215-1221.

[3] 李帅帅. 汽车 HMI 的发展趋势研究[J]. 今传媒:学术版,2020,28(1):91-93.

[4] 李具元. 汽车仪表——从全液晶虚拟仪表到智能座舱域[J]. 汽车电器,2020(8):5-6.

[5] BENEDETTI D,AGNELLI J,GAGLIARDI A,et al. Design of a digital dashboard on low-cost embedded platform in a fully electric vehicle[C]//2020 IEEE international conference on environment and electrical engineering and 2020 IEEE industrial and commercial power systems Europe (EEEIC/I&CPS Europe). Dalian:IEEE,2020:1-5.

[6] 陈 程,张 磊,董延军. 基于嵌入式操作系统的 MINIGUI 图形组件的分析与移植[J]. 信息通信,2018(3):137-138.

[7] 尹玉梅,刘建新,田东亮,等. 基于 CAN 总线和 MiniGUI 的虚拟仪表设计[J]. 电子技术应用,2010,36(2):84-86.

[8] QIU Jinhui,LIU Donghui,YUAN Junchao. The application of Qt/embedded on embedded Linux[C]//Proceedings of the 2012 international conference on industrial control and electronics engineering. Xi'an:IEEE,2012:1304-1307.

[9] 王 凯. 基于 Qt/Embedded 的嵌入式 GUI 显示架构实现[J]. 计算机技术与发展,2017,27(5):144-148.

[10] 李桂兰,阳旭东. DirectFB 和 Cairo 的嵌入式图形开发实践[J]. 单片机与嵌入式系统应用,2008(12):77-79.

[11] 龚 珏. 基于 Unity3D 的移动游戏客户端框架设计与应用[D]. 武汉:华中科技大学,2018.

[12] 蒋明欣,徐永晋. 基于 Cairo 跨平台波形呈现的软件设计与实现[J]. 工业控制计算机,2015,28(12):30-32.

[13] 许 勇,陈蜀宇. 基于 ARM 的嵌入式 Linux 图形界面的研究与实现[J]. 计算机系统应用,2011,20(10):137-141.

[14] 胡练达,张 激. 基于 GTK+的嵌入式图形系统性能优化研究[J]. 计算机工程,2014,40(1):287-290.

[15] KORANNE S. Handbook of open source tools[M]//Graphics and image processing libraries. Boston:Springer,2011.

[16] 李睿琦,牛新环,王征宇,等. 基于 i.MX6Q 和 OpenGL ES 的汽车虚拟仪表的设计[J]. 河北工业大学学报,2017,46(2):1-5.

[17] 乔旭兴. 基于 I.MX6 的数字化仪表设计与实现[J]. 计算机测量与控制,2015,23(11):3899-3903.

[18] 王卫忠. MCU+GDC 双系统的汽车虚拟仪表设计[J]. 单片机与嵌入式系统应用,2019,19(7):68-70.