# 浙江大学

## 本科生毕业设计


题目 家用小型智能机器人的灯光交互子系统的设计与实现

姓名 王灵珍

学号 3130102140

指导教师 张宏鑫

年级与专业 2013 级 软件工程

学院 计算机科学与技术学院

提交日期 2017 年 5 月 29 日

A Thesis Submitted to Zhejiang University
for the Degree of Bachelor of Engineering


TITLE Design and Implementation of Lighting Interaction Subsystem for Intelligent Household Robot

Author WANG Lingzhen

Student ID 3130102140

Supervisor ZHANG Hongxin

Major Software Engineering

College College of Computer Science and Technology

Submitted Date May 29, 2017

# 浙江大学本科生毕业论文（设计）独创性声明

本人声明所呈交的毕业论文（设计）是本人在导师指导下进行的研究工作及取得的研究成果。除了文中特别加以标注和致谢的地方外，文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 浙江大学 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在文中作了明确的说明并表示谢意。

作者签名：　　日期：　　年　　月　　日

# 毕业论文（设计）版权使用授权书

本文作者完全了解 浙江大学 有权保留并向国家有关部门或机构送交本文的复印件和磁盘，允许本文被查阅和借阅。本人授权 浙江大学 可以将毕业论文（设计）的全部或部分内容编入有关数据库进行检索和传播，可以采用影印、缩印或扫描等复制手段保存、汇编毕业论文（设计）。

（保密的毕业论文（设计）在解密后适用本授权书）

作者签名：　　导师签名：

日期：　　年　　月　　日　　日期：　　年　　月　　日

# 摘要

近年来，智能家居领域发展飞速，家用智能机器人因其逐渐成熟的技术以及智能家居市场逐渐增长的需求，也越来越被大众关注。ROKID 公司是国内一家人工智能公司，智能机器人 Pebble（中文名：月石）是该公司研发的第二款产品。Pebble 外观形似鹅卵石，采用磨砂合金材质，相较于第一款产品 Alien 取消了半透明屏幕显示，因此 Pebble 周圈的环形发光二极管（Light Emitting Diode，LED）为其主要的显示交互方式。本设计的主要工作为对 Pebble 产品的灯光交互子系统进行设计与实现，并且同时需要考虑灯光交互子系统的对于不同产品的适配性，能够将其移植到公司其他的产品如 Alien 中，实现灯光交互子系统的统一维护。

本设计基于 Android 平台进行开发，采用了生产者/消费者模式，开发工作主要分为硬件抽象层以及应用程序框架层。在应用程序框架层中，通过定义绘制节点以及将灯光变化抽象化为作用在节点上的逻辑处理，来实现绘制节点的属性变化，并且采用渲染树结构来管理绘制节点，通过光栅化算法将绘制节点转化为 LED 灯光的像素值，使用 Alpha 混合算法来实现混合渲染效果。硬件抽象层实现将像素数据传递到 Linux 内核空间，从而实现对硬件设备文件的读写。

该设计实现了 Pebble 产品的所有灯光需求，满足了良好的跨产品可移植性，满足了良好的人机交互原则，能够让用户在使用产品过程中获得良好的显示交互体验。

**关键词**：家用智能机器人，灯光交互，Android 开发，光栅化

# Abstract

In recent years, the development of intelligent home field is rapid. Household intelligent robots gain more public attention because of the more mature technology and growing demand of smart home market. ROKID Company is a domestic artificial intelligence company. Intelligent robot Pebble is the company's second product. Pebble looks like a pebble. Pebble uses a frosted alloy material. And by compared to the first product Alien, it gave up the translucent screen display. So the LEDs around the Pebble become its main display interactive way. The main work of this design is to design and implement the lighting interaction subsystem of Pebble. And at the same time the work will take the adaptability of different products into consideration when designing the lighting interaction subsystem.

The design and development is based on the Android platform. And the work will use the producer / consumer model. The development work is divided into two parts: Hardware Abstraction Layer and Application Framework Layer. In the Application Framework Layer, the work define the drawing node and abstract the lighting change into the logical processing on the node. And in this way, the attribute change of the drawing node is realized. And the structure of tree is used to manage the drawing nodes. The rasterization algorithm will be used to change drawing Node into the LED pixel value. And then the work will use Alpha hybrid algorithm to achieve hybrid rendering effect. Hardware abstraction layer to achieve transporting the pixel data to the Linux kernel space, in order to write and read the hardware device file.

The design achieves all the lighting requirements of Pebble. The work meets good portability and the principle of good human-computer interaction. It will help users to have a good display experience when using the product.

**Keywords** HouseHold Intelligent Robots、Light Interaction、Android Develop、Rasterization

# 目录

# 第1章 绪论

## 1.1 项目的环境背景

随着科学技术的快速发展以及人类社会的进步，人们对生活水平的要求不断提高，家居环境是人们生活中接触最频繁的场所，因此智能家居技术自从问世以来，与其相关的产品和技术发展迅速，应用于各个领域的智能家居产品相继推出并应用到人们的日常生活中。家用智能机器人作为近年来越来越成熟的产品，被越来越多注重智能化家居生活的人们所选择。

家用智能机器人注重人与产品之间的互动，优秀的交互设计能更大程度地消除人机之间的隔阂和沟通障碍，从而使得人与产品的互动更加有效和便捷，使得用户的家居环境更加舒适，工作学习更加高效，达到智能家居产品服务人们生活的目的。

在计算机产业发展的过程中，人与计算机的交互模式从最简单的命令行开始，慢慢出现简单的图形界面 GUI，之后随着 Web 和 Mobile 的出现以及广泛应用，图形界面包含的内容也越来越丰富，呈现方式也越来越多元化，只是这些都是基于传统的人机交互方式：视觉/手动交互方式。但是随着语音技术的发展，人机智能语音交互方式被越来越广泛地应用在智能家居方面。智能语音交互方式不仅解放了人们的双手，也将人们对于接收信息的注意力从图形界面转移到语音方式上来，使得人机交流越来越自然，越来越接近于人与人之间的交流和沟通模式。

语音交互模式通过将大量的信息从图形界面转移到语音交流中，大大减少了显示界面需要包含的信息，使得人机沟通的过程更加愉快轻松直观。但是语音交互并不能完全代替显示交互，因为语音交互具有局限性：例如语音交互易受到家庭环境中噪声的干扰，用户在接听电话、与他人交谈或者收看电视等情况下，不适合收听语音；另外语音传递消息的瞬时性较强，不适合表达持续性状态，当智能机器人不播放语音时，用户不易得知其当前状态。

因此为了达到更好的人机交互体验，家用智能机器人的交互方式应当是多感官的、多维度的。这种情况下，能够表现智能语音交互机器人基础状态与情绪的灯光显示交互模块必不可少，设计良好的灯光显示系统可以满足人们的视觉

感官需求，配合智能语音机器人的各项行为，实现更轻松自然的人机交互。

## 1.2 家用智能机器人的发展状况

然而自机器人问世以来，各国对机器人未有一个统一准确的定义，如中国科学家对机器人的定义是“机器人是一种自动化的机器，所不同的是这种机器具备一些与人或生物相似的智能能力，如传感能力、规划能力、动作能力和协同能力，是一种具有高级灵活性的自动化机器”[1]。目前联合国标准化组织采纳了美国机器人协会对机器人下的定义：“一种可编程和多功能的操作机或是为了执行不同的任务而具有可用电脑改变和可编程动作的专门系统。一般由执行机构、驱动装置、检测装置和控制系统和复杂机械等组成”[1]。

在大部分文学与影视电影中出现的机器人大多是人形的。但是面对市场需求和实际应用，综合了众多技术突破的耗资巨大的人型拟态机器人未必是个经济合算且实用的选择，智能机器人的产品形式不应当是单一的[2]。同时，随着数字化的进展、云计算等网络平台的充实以及人工智能技术的进步，智能机器人技术的开发速度越来越快，智能度越来越高，更多的机器人能通过独立的智能控制系统驱动来联网访问现实世界的各种物体或人类，因而而机器人的概念涵盖的概念更加广泛。

家用智能机器人产品是直接面向普通用户人群的，所使用的环境是复杂的日常生活区域。因此，家用智能机器人更注重高效的人机交互方式以及更优秀的环境适应能力。目前市场上，家用机器人涵盖了各种功能：有玩具型、家政服务型、交通工具型等。另外还有以智能音箱为主要产品形态的家庭陪伴型机器人，这一产品形态的家用智能机器人有不少：Amazon 公司的 Echo、Google 公司的 Google Home、科大讯飞与京东联手推出的 DingDong 智能音箱以及 Rokid 公司的 Alien 和 Pebble。这几款产品都是主要通过智能语音交互方式来控制的，同时也配备灯光显示交互系统。

## 1.3 项目的基本介绍

本设计项目将基于上一小节中提到的 ROKID 公司的产品 Pebble 来进行，Pebble 是一款以陪伴为目的，通过语音控制实现百科问答、播放音乐新闻、设置闹钟、控制智能家居等功能性以及娱乐性服务的家用智能机器人。其设计符合当

前智能家居机器人的发展趋势，主要采用语音技术进行交互，语音与灯光作为主要的信息反馈方式。

Pebble 外形整体呈鹅卵石形状，据设计师介绍其设计灵感来源于单细胞生物，暗示 AI 正处于元年阶段。Pebble 的光滑曲面金属边沿配置了全色 LED 环形阵列，LED 外圈采用光学性能良好的树脂材料，打造柔光效果，如图 1-1 所示。

LED 光源具有寿命长、光效高、无辐射、低功耗、响应快等特点，并且 LED 点阵的尺寸设计灵活，其小规格设计适合不需要显示高分辨率图形的语音交互机器人。通过该 LED 光环能够表现家用智能机器人的交互行为与工作状态。

智能机器人的实现不仅仅依赖于成熟的硬件设施，更依赖能实现用户个性化需求体验的软件环境。Android 作为基于 Linux 的开源手机操作系统，在嵌入式便携设备的环境中应用发展良好，为智能机器人的软件开发模块提供了良好的软件平台。

基于 Android 平台来开发实现 Pebble 系统，有较为成熟的框架体系作为参考，而且 Android SDK 的可裁剪性高且可定制化自由度高，节约了开发成本，也满足系统结构层次清晰的要求。因此本设计需要实现的灯光交互子系统作为 Pebble 系统的子系统，同样也是基于 Android 平台来实现的。

图 1-1 Pebble 外形

## 1.4 全文结构介绍

本设计基于 Android 平台进行开发，软件结构采用层次化系统结构，主要实现部分为硬件抽象层以及应用程序框架层。通过对绘制节点、处理逻辑、渲染树等数据结构的设计以及光栅化算法与 Alpha 混合算法等方法的使用，设计并实

现满足良好人机交互原则的家用小型智能机器人 Pebble 的灯光交互子系统。全文分为五个章节，其内容安排如下。

第一章：介绍目前智能家用机器人的发展现状和趋势以及目前国内外一些较为出众的智能语音交互机器人，同时介绍家用小型智能机器人的灯光交互子系统的项目背景与意义，阐明灯光显示系统在家用智能机器人交互设计上的重要性。

第二章：介绍家用小型智能机器人的灯光交互子系统的整体架构与实施方案，并对使用的关键技术进行介绍与分析。根据灯光需求设计对该子系统进行建模以及模块划分。

第三章：介绍本人在整个项目过程中所做的主要工作，详细介绍该基于 Android 平台开发的家用小型智能机器人的灯光交互子系统的硬件抽象层的实现，以及应用程序框架层的代码逻辑与功能模块的设计与实现。

第四章：展示该家用小型智能机器人的灯光交互子系统的最终成果以及使用方式，对测试结果进行分析和总结。

第五章：对全文进行总结，提出本设计未来需要改进的方向，并对未来家用智能机器人的发展和前景提出期望。

# 第2章 项目实施方案

## 2.1 关键技术分析

### 2.1.1 Android 系统架构

本项目是基于 Android 平台进行开发的，而整个灯光交互子系统的目的是将 LED 颜色变化的功能进行封装，向应用程序框架层的其他模块开发以及应用程序层的 APP 开发提供可供调用的服务和 API 接口等。因此整个灯光交互子系统的架构按照 Android 标准体系架构来设计。

Android 是 Google 发布的基于 Linux 平台的开源移动操作系统，它采用软件叠层（Software Stack）的架构[3]，主要由五个部分组成：linux 内核层（Linux Kernel），硬件抽象层（Hardware Abstract Layer,HAL），系统运行时库层（Libraries 和 Android Runtime），应用程序架构层（Application Framework）和应用程序层（Applications）组成，如图 2-1 所示。

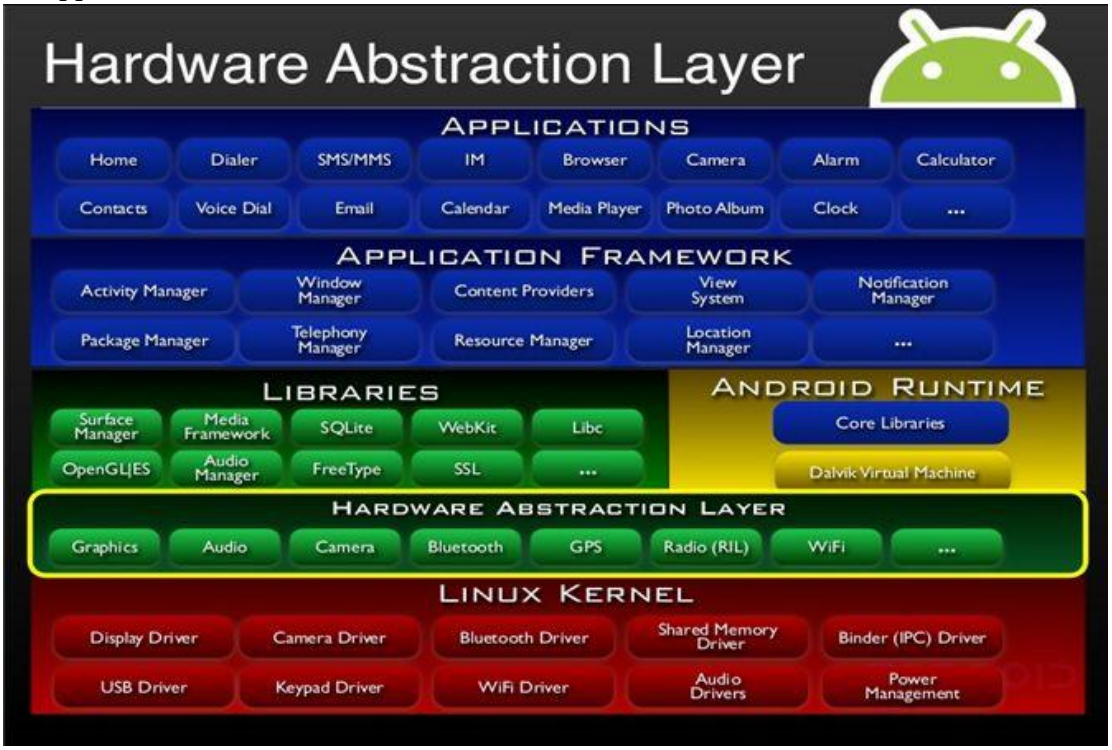


图 2-1 Android 体系架构

#### 2.1.1.1 Linux 内核层

Android 以 Linux 操作系统内核为基础，借助 Linux 内核服务实现硬件设备驱动，进程和内存管理，网络协议栈，电源管理，无线通信等核心功能。

Android4.0 版本之前基于 Linux2.6 系列内核，4.0 及之后的版本使用更新的 Linux3.X 内核。Android 内核对 Linux 内核进行了增强，增加了一些面向移动计算的特有功能。如低内存管理器 LMK（Low Memory Keller），匿名共享内存（Ashmem）,以及轻量级的进程间通信 Binder 机制等。这些内核的增强使 Android 在继承 Linux 内核安全机制的同时，进一步提升了内存管理，进程间通信等方面的安全性。

#### 2.1.1.2 硬件抽象层

硬件抽象层存在于内核驱动和用户软件之间。Android 系统为了维护硬件产商的利益，将硬件设备的实现分为两层：一层以硬件驱动模块的形式实现在内核空间中，只提供简单的硬件访问通道，另一层以硬件抽象层的形式实现在用户空间中，封装了硬件实现的细节和参数。

#### 2.1.1.3 系统运行库层

系统运行库层是应用程序框架的支撑，为 Android 系统中的各个组件提供服务。系统运行库层由系统类库和 Android 运行时构成。

系统类库大部分由 C/C++编写，所提供的功能通过 Android 应用程序框架为开发者所使用。

Android 运行时包含核心库和 Dalvik 虚拟机两部分。核心库提供了 Java5 se API 的多数功能，并提供 Android 的核心 API，如 android.os，android.net，android.media 等。Dalvik 虚拟机是基于 apache 的 java 虚拟机，并被改进以适应低内存，低处理器速度的移动设备环境。Dalvik 虚拟机依赖于 Linux 内核，实现进程隔离与线程调试管理，安全和异常管理，垃圾回收等重要功能。

#### 2.1.1.4 应用程序框架层

应用程序框架层提供开发 Android 应用程序所需的一系列类库，使开发人员可以进行快速的应用程序开发，方便重用组件，也可以通过继承实现个性化的扩展。

#### 2.1.1.5 应用程序层

应用程序层上包括各类与用户直接交互的应用程序，或由 java 语言编写的运行于后台的服务程序。

### 2.1.2 生产者/消费者模式

考虑到整个子系统的数据传递结构：由应用层以及应用程序框架层来产生 LED 像素数据，由硬件抽象层将 LED 像素的数据写入 LED 设备文件。因此可以采用生产者/消费者模式来传递 LED 数据。

在软件开发过程中，生产者/消费者模式如下：某个模块负责产生数据，将产生的数据放入缓冲区，另一个模块从缓冲区中取出数据，并处理这些数据，产生数据的模块就称为生产者，处理数据的模块就称为消费者[4]。如图 2-2 所示。

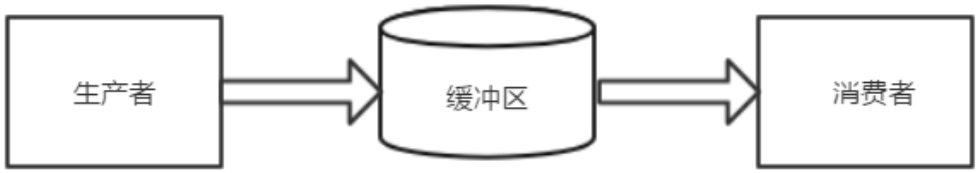


图 2-2 生产者/消费者模式

采用生产者/消费者模式有几个优点：消费者与生产者之间没有直接的函数调用关系，实现对生产者和消费者之间的解耦；生产者不需要以阻塞的方式等待消费者处理完数据后的返回，实现了生产者生产数据和消费者消费数据的并发；支持忙闲不均，生产者生产数据过快时，可以将数据放入缓冲区，当生产者生产数据慢下来时，消费者可以逐渐消费掉数据[4]。

## 2.2 项目模块划分

上一小节确定了本设计的灯光交互子系统结构按照 Android 系统体系架构来实现，需要实现以下几个模块：硬件抽象层，Native service 模块，匿名共享内存 Ashmem，应用程序框架层。

本设计的项目的开发工作是基于 API Level 23 的 AOSP（Android Open-Source Project，Android 开放源代码项目）源代码来进行的。

### 2.2.1 硬件抽象层

其中硬件抽象层位于用户空间，定义了模块结构体与设备结构体，保存 LED

硬件设备的重要参数，并且对硬件驱动模块提供的硬件访问接口函数进行封装，实现访问流程控制。LED 硬件抽象层编译后生成以.so 为后缀的动态链接库文件。

### 2.2.2 Native Service

Native Service 是一个由 C++编程实现的服务，通过动态加载（DL）LED 硬件抽象层生成的动态链接库文件来调用其中定义的数据结构与函数。这个服务的主要功能是：申请创建一块匿名共享内存 Ashmem，；充当生产者/消费者模式中的消费者的角色，开启一个线程来轮询读取匿名共享内存中的数据；通过调用硬件抽象层提供的函数写入硬件设备驱动。

### 2.2.3 匿名共享内存

匿名共享内存由 Android 系统中的 Ashmem 驱动程序负责分配和管理，使用一个文件描述符来标记。本子系统中的匿名共享内存由 Native Service 申请创建，用于存放 LED 硬件参数以及应用程序框架层需要写入 LED 硬件设备文件的像素数据，充当了生产者/消费者模式中缓冲区的角色。这块匿名共享内存的地址通过 binder 机制来实现应用程序框架层和 Native Service 之间的传递。

### 2.2.4 应用程序框架层

应用程序框架层作为生产者/消费者模式中的生产者，实现了以下几个子模块：

（1） 能够提供实现各种灯光变化功能的 API 接口的 Java Service；

（2） 能够实现 Java Service 调用本地 C/C++方法的 Java 本地接口（Java Native Service, JNI）；

（3） 能够实现通过 binder 获取 Native Service 创建的匿名共享内存地址的 C/C++

（4） 方法，获得 LED 硬件设备的参数以及可写入像素数据的内存地址。

## 2.3 代码编译与调试

开发环境：Ubuntu 14.04.3 LTS，Linux 3.19.0-25-generic

开发工具：Android Studio 2.3.1

编译环境：JDK 1.8.0_111

调试工具：Dalvik 虚拟机调试监控服务（Dalvik Debug Monitor Service，DDMS）

本项目的开发环境部署在 Linux 操作系统 Ubuntu 上，通过 AOSP 源代码中的/build/envsetup.sh 脚本初始化编译环境，通过编译命令配置编译目标，对源代码进行编译。对所有源代码编译生成的产物为 XXX.img 文件，需要烧录到 Pebble 中，通过对源代码中单独模块进行编译生成的 XXX.so、XXX.apk、XXX.lib 等文件直接拷贝到 Pebble 设备的相应目录下。运行 Pebble，通过 DDMS 来查看运行时的进程、堆信息、Logcat 等，根据进程信息或者日志信息来判断系统模块是否运行正确、排查 Bug。

## 2.4 版本控制

Git：一个开源的分布式版本控制系统，能够有效、高速地管理项目版本。

Repo：Google 使用 Python 脚本写的调用 Git 的脚本，能够管理多个 Git 项目，主要用来下载、管理 Android 项目的软件仓库。

Gerrit：一个面向 Git 版本控制系统的基于 Web 的代码评审和项目管理的工具。

本项目使用 Git 来实现对单个项目代码的版本控制，使用 Repo 来对所有 Git 项目进行管理，使用 Gerrit 来对代码进行评审核查。

## 2.5 本章小结

本章第一小节主要介绍了项目的关键技术 Android 系统体系结构以及生产者/消费者的设计模式以及为什么选用这些技术。第二小节介绍了灯光交互子系统的模块划分以及设计思路。第三小节介绍了项目的开发环境以及编译环境，同时介绍了编译方法和调试方法。第四小节介绍了对项目进行代码版本控制的方法和工具。

# 第3章 在项目中负责的具体工作

## 3.1 灯光需求分析

以下灯光需求分析是根据产品经理与设计师针对 Pebble 产品提出的灯光需求来总结的得到的，如表格 3-1 所示。

虽然项目的开发是基于 Pebble 的，但是也考虑到了该灯光系统对于不同产品的可移植性以及对于灯光需求变更的可修改性。

表格 3-1 灯光需求分析表

| 灯光功能 | 灯光描述 |
|---|---|
| 开机加载灯光 | 四颗一组蓝色跑马灯闪烁转动 |
| 联网加载灯光 | 四颗一组橙色跑马灯闪烁转动 |
| 休眠灯光 | 灯光全灭 |
| 睡眠灯光 | 深蓝色灯光 |
| 寻向灯灯光 | 语音唤醒方向有单颗白色灯光，底色为蓝色 |
| 提示灯光 | 灯光闪烁 |
| 异常灯光 | 灯光常亮 |
| 关闭麦克风灯光 | 红色灯光常亮 |
| 电量音量灯光 | 按刻度显示灯光区域 |
| 应用灯光 | 灯光全灭 |
| 语音灯光 | 白色灯光模拟语音频率闪动 |
| 关机灯光 | 灯光闪烁 |

## 3.2 系统架构

整个灯光交互子系统的架构图如图 3-1 所示。

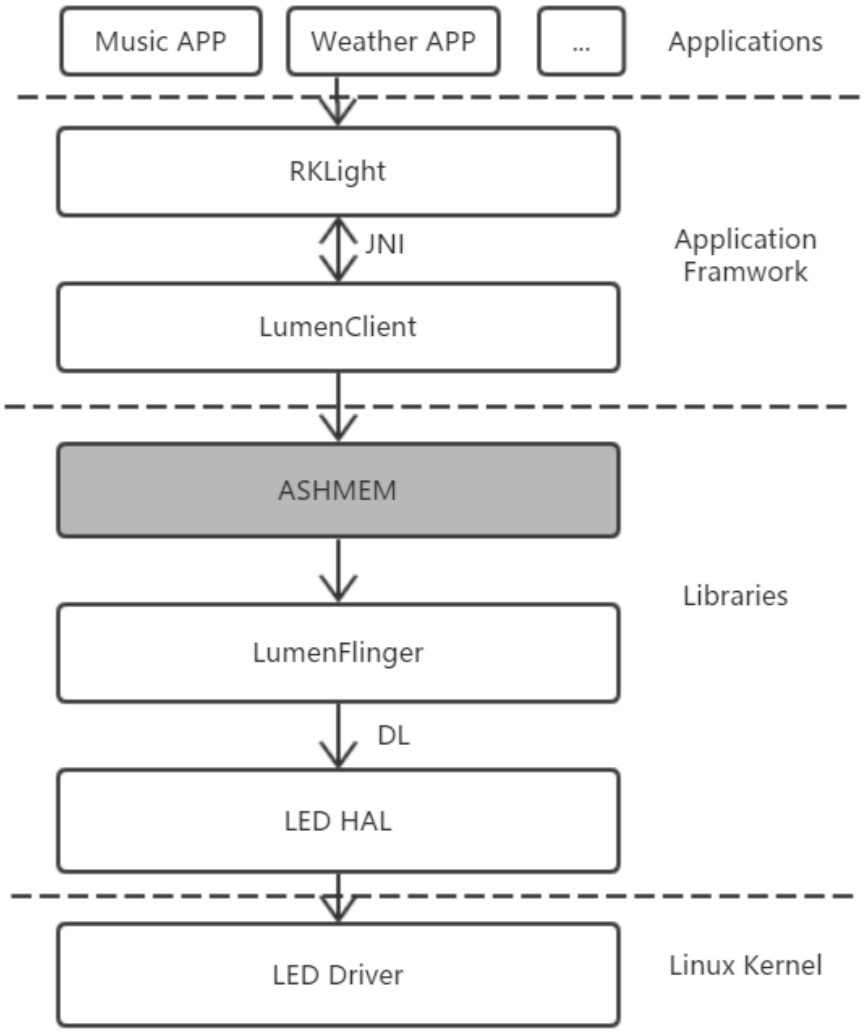


图 3-1 灯光交互子系统的架构图

LED Driver 是硬件驱动模块，位于内核空间，具有直接操作硬件设备的权限，提供了打开、读、写、关闭 LED 硬件设备文件的接口。

LED HAL 是硬件抽象层，对硬件驱动模块提供的硬件访问接口函数进行封装，实现访问流程控制。

LumenFlinger 是一个 Native Service，通过动态加载硬件抽象层生成的动态链接库文件来调用其中定义的数据结构与函数，创建匿名共享内存 Ashmem 实现进程间内存共享，开启一个线程来轮询读取匿名共享内存中的数据，并通过调用硬件抽象层提供的函数写入硬件设备。

LumenClient 通过 Binder 获得 LumenFlinger 中开辟的匿名共享内存的地址，

向这块地址的内存中写入数据。

RKlight 是一个 Service，完成 LED 灯光颜色变化的具体实现，并通过 JNI 来调用 LumenClient 中定义的函数将 LED 数据传递给 LumenClient。

位于 Applications 的各种 APP 或者 Application Framework 的其他模块通过 Binder 机获得 RKLight 的代理对象接口，通过调用接口函数实现进程间通信，实现对灯光功能的调用。

如图 3-2 所示的时序图描述了两个过程，一个是初始化过程，一个是 Client 调用 RKLight 服务接口中定义的 shotFlash 函数的整个流程。

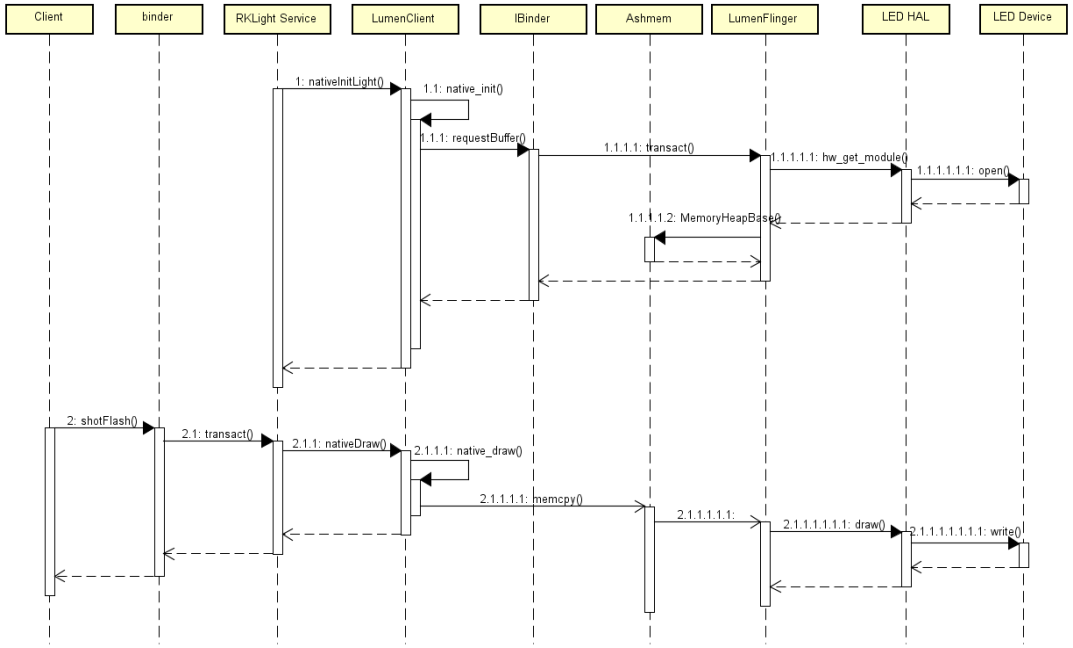


图 3-2 灯光交互子系统时序图

## 3.3 硬件抽象层

Android 系统为硬件抽象层中的模块接口定义了编写规范，必须按照这个规范来编写自己的硬件模块接口。Android 系统的硬件结构抽象层以模块的形式来管理各个硬件访问接口。每个硬件抽象层模块都使用结构体 hw_module_t 来描述，硬件设备则使用结构体 hw_device_t 来描述。另外还需要定义结构体 hw_module_methods_t，这个结构体只包含一个成员变量，用来打开硬件抽象层模块中的硬件设备[5]。

LED 硬件抽象层模块定义了个结构体以及一个结构体变量：LED 硬件抽象

层模块结构体 led_module_t 和 LED 设备结构体 ledarray_device_t 以及一个模块操作方法的结构体变量 led_module_methods。具体定义如下：

```
struct led_module_t {
    struct hw_module_t common;
};
typedef struct ledarray_device_t {
    struct hw_device_t common;
    ledarray_dev_code_t dev_code;
    int led_count;
    ledarray_pxl_fmt_t pxl_fmt;
    int fps;
    int fd;
    int (*dev_close) (struct ledarray_device_t *dev);
    int (*draw) (struct ledarray_device_t *dev,  unsigned char *buf,  int len);
    int (*set_brightness) (struct ledarray_device_t *dev, unsigned char br);
    int (*draw_one) (struct ledarray_device_t *dev, int idx, int br, int r, int g, int b);
    int (*set_enable) (struct ledarray_device_t *dev, int cmd);
} ledarray_device_t;
static struct hw_module_methods_t led_module_methods = {
    .open = led_dev_open,
};
```

按照编写规范，led_module_t 的第一个成员变量的类型为 hw_module_t，ledarray_device_t 的第一个成员变量的类型为 hw_device_t。

ledarray_device_t 中还定义了文件描述符 fd，用来描述打开的 LED 设备文件，led_count 描述 LED 的数量、pxl_fmt 描述 LED 的像素格式、dev_code 代表设备类型、fps 描述像素读取帧率，另外还定义了一些操作硬件设备文件的函数指针：dev_close 关闭设备文件，draw 写入像素值，set_brightness 写入亮度，draw_one 写入单颗 LED 像素值，set_enable 设置硬件抽象层是否可用。

在 LED 硬件抽象层模块中添加 Android.mk 编译脚本文件，指定 include 命令的参数为$(BUILD_SHARED_LIBRARY)，可以将硬件抽象层模块编译生成动态链接库文件。

## 3.4 Native Service 模块以及匿名共享内存

Native Service 模块定义了一个叫做 lumenFlinger 的 Native Service。

lumenFlinger 通过 hw_get_module 方法加载 LED 硬件抽象层模块编译生成的动态链接库文件，并得到 hw_module_t 接口，这个接口实际指向 led_module_t 结构体对象，通过 hw_module_t 的操作方法列表调用中的 open 方法来打开 LED 设备文件，得到 ledarray_device_t 结构体对象，从而得到 LED 设备的参数数据以及操作 LED 硬件设备文件的函数指针，实现读写 LED 硬件设备的功能。

lumenFlinger 在启动之后，会通过 defaultServiceManager 的 addService 方法将自己注册到 ServiceManager 中，之后就可以通过 defaultServiceManager 的 getService 方法获取 lumenFinger 的代理对象，实现通过 binder 机制访问该服务，binder 机制是通过在服务端与客户端之间传递 Parcel 包来进行数据传递的。

lumenFlinger 中创建了一块匿名共享内存，其内存分配定义如图 3-3 所示。

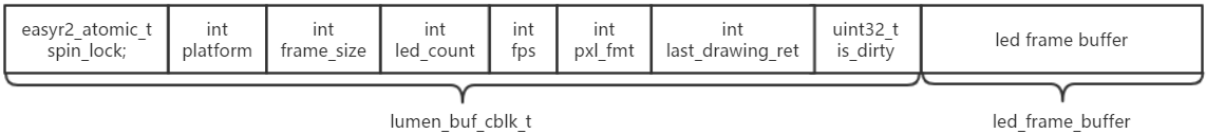


图 3-3 Ashmen 结构

这块匿名共享内存主要分为两个部分，第一部分存储了一个名为 lumen_buf_cblk_t 的结构体，定义的变量有：自旋锁 spin_lock，平台类型 platform，led_frame_buffer 的区域长度 frame_size，LED 灯数 led_count，渲染帧率 fps，像素格式 pxl_fmt，上一帧是否写入成功的标记 last_drawing_ret，以及当前 led frame buffer 是否有新数据的标记 is_dirty。第二部分 led_frame_buffer 用来存放 LED 像素数据，长度为 led_count*pxl_fmt。

lumenFlinger 作为一个 service，还开启了一个线程，这个线程每秒读取 fps 次 Ashmem 中的 led_frame_buffer 区域的数据，如果成功读取到新的数据，就调用 LED 设备结构体的函数指针将数据写入硬件设备文件，从而实现 LED 的颜色渲染。

这样，通过这个模块我们就完成了将 LED 像素数据写入 LED 硬件设备文件实现 LED 渲染的功能。

## 3.5 应用程序框架层

### 3.5.1 LumenClient 类

LumenClient 类中定义的方法，通过调用 defaultServiceManager 的 getService 方法获取 lumenFinger 的代理对象接口，然后 LumenClient 通过这个代理对象接口向 lumenFlinger 请求匿名共享内存的地址，lumenFlinger 通过 Binder 将匿名共享内存的地址打包在 Parcel 包中，传递过来。

LumenClient 获取到匿名共享内存的地址，就获得了 LED 硬件设备的参数信息，也获得了写入 LED 像素数据的地址。

### 3.5.2 JNI 方法

JNI 方法实现在 Java 语言中调用使用 C/C++实现的方法[6]。因此这一模块是为了实现 RKlight 服务调用 LumenClient 中的方法。

该模块定义了四个函数：native_init、native_draw、native_set_enable、native_finalize。在 native_init 函数中，实例化 LumenClient，得到一个 LumenClient 对象，获取匿名共享内存的地址。native_draw 函数实现将 LED 像素数据写入匿名共享内存区域。native_set_enable 实现控制硬件抽象层的可用性，控制 lumenFlinger 的线程是否循环读取匿名共享区域的数据。native_finalize 函数关闭 LED 硬件设备文件。

然后在 registerLightNativeMethods 函数中定义了 JNI 方法表，分别将 RKLight 服务中的函数 nativeInitLight、nativeDraw、nativeSetEnable、nativeFinalize 的 JNI 方法注册为 native_init、native_draw、native_set_enable、native_finalize，并通过 jniRegisterNativeMethods 函数将 JNI 方法表注册到 Java 虚拟机中。再在 JNI_OnLoad 函数中调用 registerLightNativeMethods 函数，最终实现将 native_init、native_draw、native_set_enable、native_finalize 方法注册到 Java 虚拟机。

### 3.5.3 RKLight 服务

RKLight 服务实现了第三章第一小节中定义的灯光需求，即实现了 LED 像素值的改变。其他进程可以通过访问该服务来实现灯光变化。

为了能使其他进程访问该服务，需要定义服务接口。使用服务接口来进行进程间通信时，RKLight 作为提供服务的进程称为 Server 进程，使用服务的进程称为 Client 进程。在 Server 进程中，每个服务都对应一个 Binder 本地对象，它通

过一个 Stub 等待 Client 进程发送进程间通信请求。Client 进程在访问运行 Server 进程中的服务之前，需要获得他的 Binder 代理对象接口，然后通过这个接口向 Server 发送进程间请求[5]。

将灯光变化的需求进行总结，可以在服务接口中定义一系列能够实现不同灯光变化功能的函数。Client 进程通过调用这些函数改变灯光。

实现 LED 像素变化主要分为几个步骤：

（1） Client 进程调用 API 函数；

（2） Server 进程处理 Client 请求；

（3） 按照灯光需求将灯光显示进行层次化划分；

（4） 定义绘制节点；

（5） 改变绘制节点的属性值；

（6） 将绘制节点的属性转化为像素数据；

（7） 使用基于 Alpha 混合的算法将像素数据写入 LED 数据缓冲区；

（8） 调用 JNI 方法将 LED 像素数据写入 LED 设备文件。

#### 3.5.3.1 Client 进程调用 API 函数

RKLight 服务将灯光变化的实现封装成了一系列 API 函数，Client 通过调用函数向 Server 进程请求实现某种灯光变化。Binder 将 Client 的请求数据打包成 Parcel 对象，使用 transact()传递给 Server 进程，同时还会将 Client 进程调用的函数对应的标识传递给 Server 进程，以便 Server 进程知道 Client 调用了哪个函数。

#### 3.5.3.2 Server 进程处理 Client 请求

RKLight 服务接收到请求，可以知道 Client 调用了哪个 API 函数，然后从 Parcel 对象中读取实现灯光变化所需的数据，比如寻向灯需要角度值、音量灯光需要音量大小等。接下来是选择如何实现灯光变化。

#### 3.5.3.3 对灯光显示进行层次化划分

因为不同模块可能同时调用不同的灯光，所以为了清晰地显示灯光，需要为不同的灯光划分优先级，优先级高的灯光叠加在优先级低的灯光上面。本设计定义了 Layout 类，不同优先级的灯光变化在不同的 Layout 中实现。

如图 3-4 所示，基于 Layout 类定义了 8 个派生类。根据 mZOder 的值来确定哪一层优先级高，mZOder 越小，优先级越高。

其中 PowerLayout 显示关机灯光，优先级最高。LockLayout 显示关闭麦克风灯光。HibernateLayout 显示休眠灯光。ModalLayout 显示音量灯光和电量灯光。FlashLayout 显示闪烁灯光。ConfigLayout 显示开机灯光和配网灯光。MindLayout 和 ExecLayout 显示唤醒、睡眠、加载、应用语音播放等灯光。

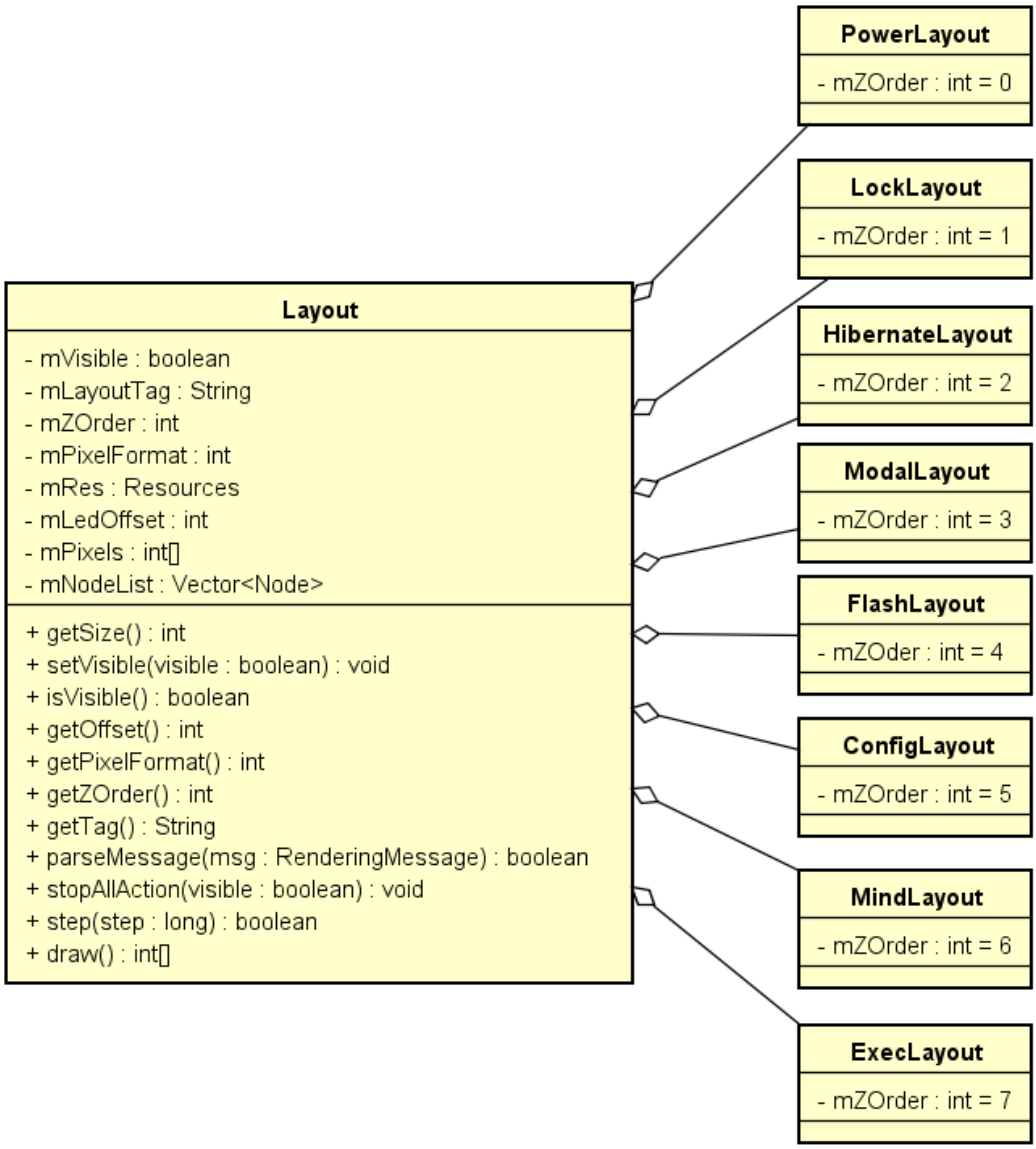


图 3-4 Layout 类图

本设计定义了 LayoutManager 来管理 Layout，实现在每一帧计算像素数据时，先将优先级低的 Layout 中计算得到的像素值写入 LED 像素缓冲区。

#### 3.5.3.4 定义绘制节点

为了实现在每一层 Layout 中实现灯光颜色变化，本设计定义了绘制节点 Node，Node 描述了一个一维图形，可以看做是一条线段。每一层 Layout 中定义不同的 Node 来实现不同的灯光变化。

如图 3-5 所示，定义了两个类 Vertex 和 Node，其中 Vertex 用来描述绘制节点中某一点的属性。Node 定义的成员变量有：vertices 代表了一组描述 Node 各点属性的 vertex，position 描述 Node 在 Layout 中的位置，anchor 描述 Node 的锚点，scale 描述 Node 的比例大小，opacity 描述 Node 的透明度，size 描述 Node 的整体大小，visibility 描述该 Node 是否可见，tag 描述该 Node 的标签，parentNode 描述该 Node 的父节点，childrenNode 描述该 Node 的子节点，actionList 描述定义在该 Node 上的一系列需要执行的属性变化。Node 还定义了一系列改变属性的函数。

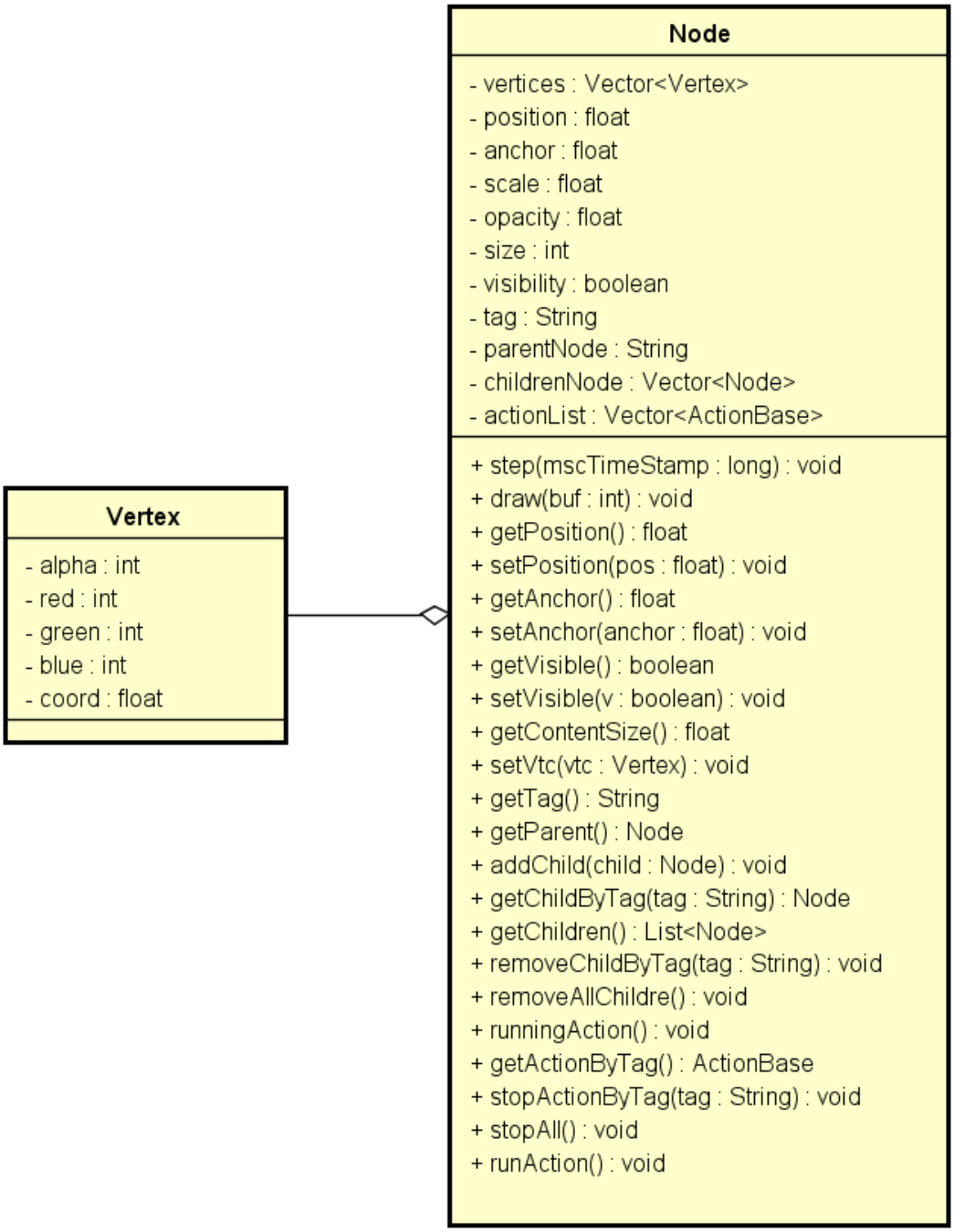


图 3-5 Node 类图

PowerLayout、LockLayout、HibernateLayout、ModalLayout、FlashLayout 中都各自定义了一个长度等于 LED 所有像素大小的 Node，根据不同的属性变化来实现灯光变化。ConfigLayout 中定义了一个包含四个长度为 1、距离间隔相同的

子节点的 Node。MindLayout 中定义了一个长度为 1 的节点，ExecLayout 中定义了一个长度等于 LED 所有像素大小的 Node。

#### 3.5.3.5 改变绘制节点属性值

本设计通过定义 Action 来实现对绘制节点属性的改变，首先定义基类 ActionBase，包含的属性有：动作的周期，动作的已执行时长，重复次数、是否恢复 node，是否可见、是否可超出父节点区域显示。针对 Node 属性定义了许多派生类，主要是实现对于 Node 的单一属性的改变，如图 3-6 所示。

ScaleToAction 描述 Node 绝对比例的变化过程。ScaleByAction 描述 Node 相对比例的变化过程。MoveToAction 描述了 Node 的绝对位置的变化过程。MoveByAction 描述了 Node 绝对位置的变化过程。Show 描述了 Node 瞬间出现的动作。HideAction 描述了 Node 消失的动作。ChangeColorAction 描述了 Node 上某个点的属性变化。FadeAction 描述了 Node 透明度的变化。FadeOutAction 描述了 Node 逐渐消失的过程。FadeInAction 描述了 Node 逐渐出现的过程。

其中比较特别的是 Sequece，它定义了一个 ActionBase 的列表，该列表中的 Action 将按顺序执行。而 Node 的 Action 列表中的 Action 将同时执行。

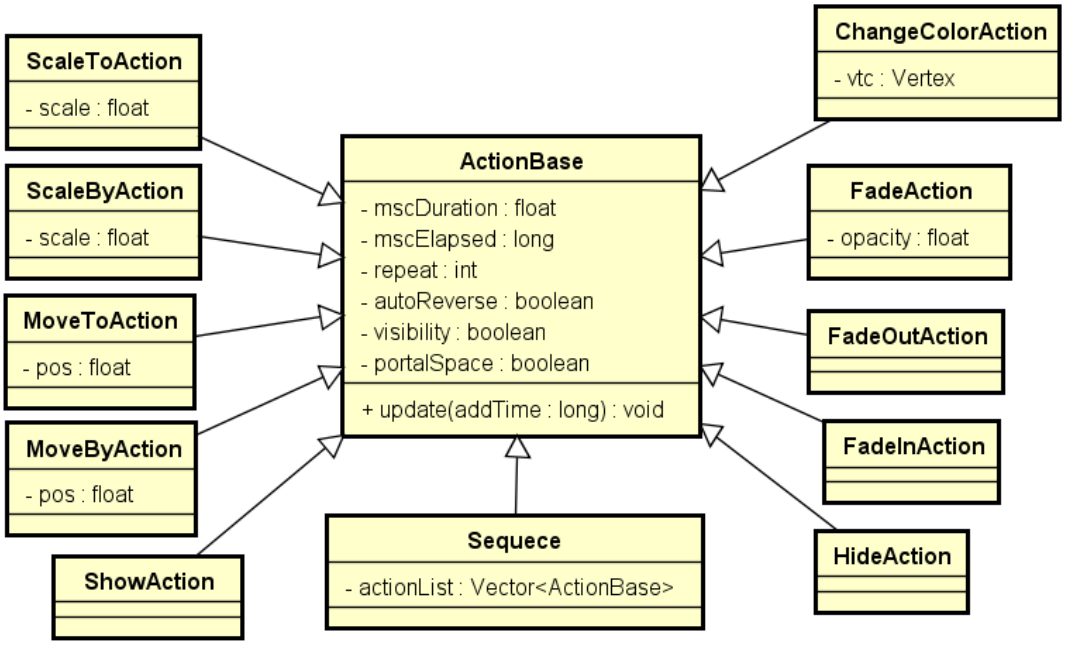


图 3-6 ActionBase 以及派生类的类图

PowerLayout 层中定义的 Node 主要是通过添加 FadeAction 和 ChangeColorAction 来实现关机时快速闪烁三次再逐渐变暗的关机灯效。

LockLayout 层中定义的 Node 主要是通过 ScaleToAction 来实现关闭麦克风时红色灯光延伸出现和打开麦克风时红色灯光收缩消失的灯效。

HibernateLayout 层中定义的 Node 主要是通过 ScaleToAction 来实现休眠时时灯光收缩消失和结束休眠时灯光延伸出现的灯效。

ModalLayout 层中定义的 Node 主要是通过 ScaleToAction 根据电量和音量按比例显示灯光。

FlashLayout 层中定义的 Node 主要通过 FadeAction 和 ChangeColorAction 来实现灯光闪烁的效果。

ConfigLayout 层中定义的 Node 主要通过 FadeInAction、FadeOutAction 来实现四颗灯闪烁转动的灯效，通过 ChangeColorAction 来实现开机灯光到联网灯光的变化。

MindLayout 和 ExecLayout 层是一起工作的，MindLayout 层通过 ScaleToAction、MoveToAction、FadeAction 来实现唤醒时单颗灯显示的灯效、加载时单颗灯快速移动的灯效以及语音播放时灯光不规则闪动的灯效，ExecLayout 层通过 FadeAction 和 ChangeColorAction 来实现使用应用时全黑的灯效、睡眠时蓝色的灯效以及唤醒和加载时灯效的深蓝色底色效果。

#### 3.5.3.6 光栅化操作

图形光栅化的过程其实就是：确定一个最佳逼近图形的像素集合，并用指定属性写像素的过程。对于一维图形，若不考虑线宽，则使用一个像素宽的直线来显示图形[6]。

本项目只需要对一维图形进行光栅化，因此本设计中使用的光栅化方法是：根据绘制节点的位置和长度，计算绘制节点覆盖了哪些像素点。当一个像素点被部分覆盖时，如果去除该像素点，则会导致子像素失真，因此我们参考基于采样的抗锯齿技术[8]，当采样点趋于无穷多的时候，就相当于计算绘制节点在像素上的覆盖率，然后将覆盖率作为比例计算出应该写入该像素点的颜色值。

光栅化过程如图 3-7 所示，先计算绘制节点覆盖了哪些 LED 像素点，将绘制节点进行分割成一像素宽或者不足一像素宽的片段，计算每个片段中心点的颜色，以覆盖率作为比例乘以颜色透明度，将计算后得到的颜色值写入对应的

LED 像素点。

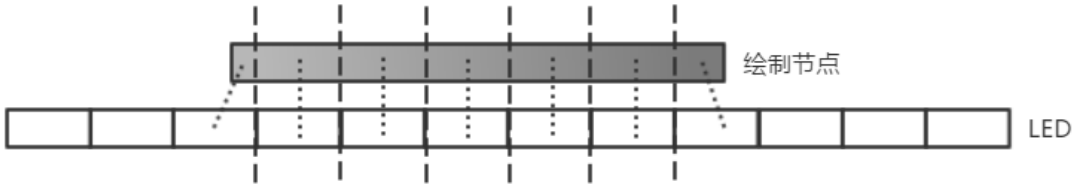


图 3-7 光栅化模型

#### 3.5.3.7 混合渲染

计算出应该写入哪些像素点以及对应的像素颜色值后，还需要考虑到像素的颜色叠加，为了达到更好的叠加效果，采用基于 Alpha 混合的方法来计算叠加之后的像素颜色值[9]。

该 Alpha 混合方法的公式为：

$$A = A_1 + A_2 - A_1 \times A_2$$

$$C_{12} = [C_1 \times A_1 \times (1 - A_2) + C_2 \times A_2]/(A_1 + A_2 - A_1 \times A_2)$$

其中，A 为混合后的透明度，$A_1$ 为背景色透明度，$A_2$ 为前景色透明度。$C_1$ 为背景色的 r、g、b 值，$C_2$ 为前景色的 r、g、b 值，$C_{12}$ 为混合后的 r、g、b 值。

#### 3.5.3.8 调用 JNI 方法

RKlight 服务通过调用 native_draw 的 JNI 方法，将最终得到的像素点的颜色数据写入到匿名共享内存中。

## 3.6 本章小结

本章节介绍了本设计的主要工作：首先是根据对灯光需求进行分析，然后是设计系统架构，并给出了时序图，然后介绍了硬件抽象层、Native Service 模块、应用程序框架层的具体实现细节与实现原理。

# 第4章 项目成果

## 4.1 灯光交互子系统提供的 API

本设计实现的灯光交互子系统实际上是实现将灯光变化封装成了一系列 API 接口，其他模块或应用通过调用 API 接口来实现灯光变化。灯光交互子系统所定义的灯光 API 如表格 4-1 所示：

表格 4-1 灯光 API 函数

| API | 功能 |
|---|---|
| void setMind(float angle) | 激活寻向灯 |
| void setMindPause() | 加载跑马灯 |
| void setMindSleep() | 睡眠状态灯 |
| void setMindExec() | 退出加载跑马灯 |
| void setMindExecInReal() | 应用状态的灯光 |
| void setMindTTS(boolean cmd) | 语音播放时的灯光 |
| void setLock(boolean cmd) | 关闭麦克风的灯光 |
| void setHibernate(boolean cmd) | 休眠状态的灯光 |
| void setPowerOff() | 关机时的灯光 |
| void setStandby(boolean cmd) | 联网时的加载灯光 |
| void setBusy(boolean cmd) | 开机时的加载灯光 |
| void shotConfirm(int repeat) | 确认的闪烁灯光 |
| void shotException(int repeat) | 异常时的闪烁灯光 |
| void shotFlash(int color, long duration, int repeat) | 通用闪烁灯光 |
| void shotRing(int color) | 异常灯光 |
| void dismissModal() | 取消异常灯光 |
| void setProgress(int type, float ratio) | 音量电量灯光 |
| void shotVolumnFlash() | 音量最小提示灯光 |

使用方法：通过 RemoteServiceHelper 的 getService 函数，获得 RKLight 服务的代理接口，然后可以直接调用上面列出的 API 函数。

## 4.2 灯光交互子系统运行效果

目前 Pebble 上面的灯光功能已基本实现，灯光调用分布在各个模块中。我

们可以通过直接使用 Pebble 来观察灯光效果。

因灯光变化情况较多，所以以下只贴出部分实际灯光效果：

图 4-1 开机加载灯光效果

图 4-2 关闭麦克风状态下的灯光

图 4-3 激活状态下的寻向灯

图 4-4 睡眠状态下的灯光

## 4.3 成果分析

本设计基本完成了项目目标，实现了 Pebble 的灯光需求，并且在设计整个灯光架构时考虑到了不同产品的适应性，以及灯光需求的扩展性。该灯光交互子系统具有良好的可移植性和可维护性。

而且该产品已经由公司测试组多次迭代测试，已基本修复所有排查出来的 bug，目前该子系统在 Pebble 产品上运行状况稳定。

# 第5章 本文总结

## 5.1 工作总结

本文对家用智能机器人的发展进行了介绍，分析了家用智能机器人的交互方式，指出了灯光显示交互对于智能语音机器人的重要性。本文还介绍了 Android 的体系架构以及生产者/消费者模式，并说明了本文描述的子系统为什么是根据 Android 体系架构和生产者/消费者模式来进行设计的。本文还简略介绍了能够提升开发效率的开发工具，编译工具以及版本控制工具，

本文重点是介绍硬件抽象层、Native Service、应用程序框架层的开发细节，详细描述了整个子系统的工作逻辑。

最后给出项目成果：一组能够实现灯光变化的 API 函数。并且展示了部分产品使用过程中的实际的灯光效果。

## 5.2 展望

目前该灯光交互子系统基本实现 Pebble 产品的所有灯光需求且运行状况正常。接下来计划尝试将其移植到公司新一代的产品上验证其可移植性，并根据新产品的灯光需求进行进一步迭代开发，使得该灯光交互子系统能以同一套代码同时正常运行在不同的产品中，实现代码的统一维护。

# 参考文献

[1] 孟庆春,齐勇,张淑军,杜春侠,殷波,高云. 智能机器人及其发展[J]. 中国海洋大学学报(自然科学版),2004,(05):831-838.

[2] 赵巍. 家用机器人的产品形式与设计要素[D].浙江大学,2008.

[3] 马越.Android 的架构和应用[D].中国地质大学硕士学位论文,2008

[4] 编程随想. “生产者/消费者”模式初探[J]. 程序员,2009,(09):66-70.

[5] 罗升阳. Android 系统源代码情景分析. 电子工业出版社.2012.13-14

[6] Liang S. The Java Native Interface: Programmer's Guide and Specification[J]. Crossroads, 1999(2):172-173.

[7] 孙家广,胡事民. 计算机图形学基础教程. 清华大学出版社. 2009-08-01

[8] 抗锯齿技术深度解析[EB/OL]. http://www.cbgitek.com/2015/06/06/anti-aliasing

[9] Alpha 混合：两个半透明色的叠加[EB/OL]. http://blog.csdn.net/richardbao2000/article/details/2682018

[10] Google. Android developer [EB/OL].http://developer.android.com

# 致谢

# 本科生毕业论文（设计）任务书

一、题目 家用小型智能机器人的灯光交互子系统的设计与实现

二、指导教师对毕业论文（设计）的进度安排及任务要求

起讫日期 年 月 日 至 年 月 日

指导教师（签名）职称

三、学院审核意见

负责人（签名）

年 月 日

# 毕 业 论 文（设计） 考 核

## 一、指导教师对毕业论文（设计）的评语

指导教师(签名）

年　月　日

## 二、答辩小组对毕业论文（设计）的答辩评语及总评成绩：

| 成绩比例 | 开题报告占（20%） | 外文翻译占（10%） | 中期检查占（10%） | 毕业论文质量及答辩占（60%） | 总成绩 |
|---|---|---|---|---|---|
| 分值 | | | | | |

答辩小组负责人（签名）

年　月　日