**读版图书馆项目开发文档**

1. **原版结构（BookBody）**
2. **~~导读页（BookGuide）~~**

**第一单独的一页，没有过场动画，显示书的概要内容，更像是静态的封面，看过即删；第二版调整，加入过场动画，而且要频繁展示导读页，便砍掉了单独类控制，整合到了BookBody里面，通过**BookBody::doActionOpen()打开导读页，通过BookBody::doActionClose()关闭导读页，而且在BookBody内部控制导读页，更方便灵活。

1. **内容页（PageBase）**
2. **正文页（PageContent）**
3. **正文图片（**eBookResPng）

**正文都必然有一张图片，图文并茂，该图片的展示，是通过下载回调来刷新，默认是空白页，本地存在或者下载成功，刷新图片的大小和坐标。**

1. **正文音频（**eBookResSnd）

**带有文字的正文页，都会有一段范读音频，在音频下载完之前，会禁止用户点“播放”按钮。**

1. **正文文字（**eBookResOth）

**有范读和文字的正文页必然会有一个“歌词”配置文件，Json格式，用于组装字面句子文字，标记范读的时间和节奏。**

1. **交互页（**PageInteract）
2. **交互按钮**

**交互方式有三种（播、录、回放），任意两种交互模式不能并存，否则打乱控制器，影响数据的准确度**

1. **播放**

**播放范读音频，最好控制，就是播和停，服务器有两套音频（ogg+mp3），还好接入了ogg播放组件，资源小，音频质量还行。**

1. **录音**

**录制用户自己的朗读，朗读完后会上传录音资源，并对其打分，资源格式是speex流文件，18000采样。**

1. **回放**

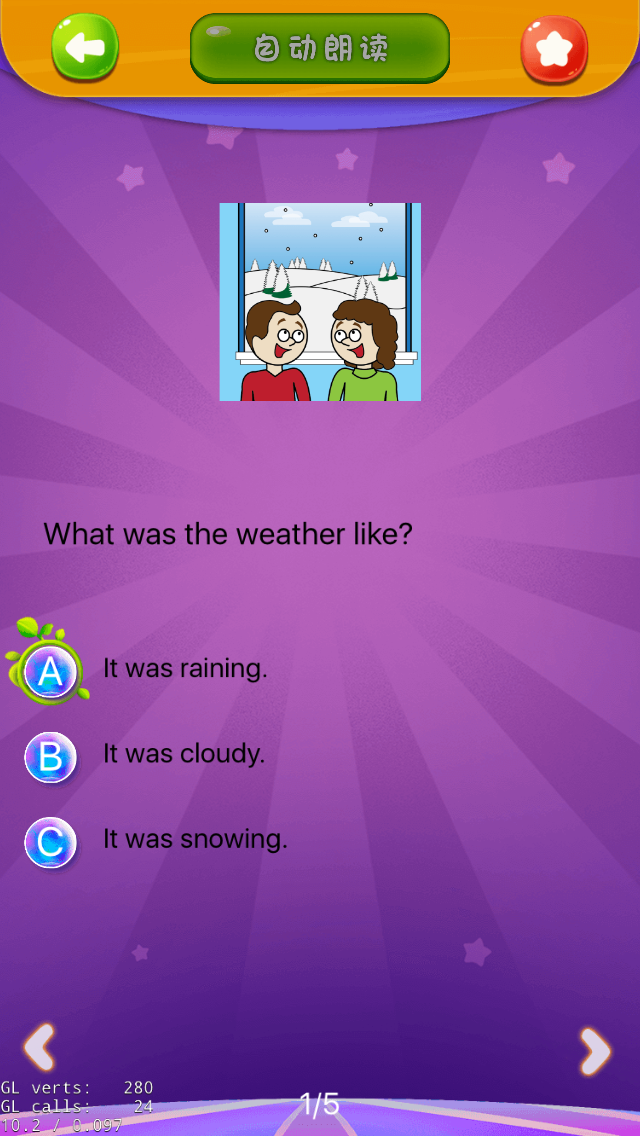
**录制的speex资源回放出用户的朗读，更直观的感受自己朗读的效果，这块播放speex时，比较尴尬的是Android不能在回放中途停止回放，必须要等回放完等回调，这块需要后续的技术人员完善一下SDK。**

1. **伸缩板**

**弹板上会展示文字内容，并且在播放时体现歌词渐变色效果，这块是用MyText控件，传入时间和文字数组，做好文本的监听，通过计时器控制符号的变色。**



1. **答题页（**PageQuizTemp）

**该类只是一个模板，用于展示各种题型（5/5），他是各个题目的parent，老丁会根据这个板子的大小，控制答题内容的展示细节，不再赘述**

1. **口语页（**PageQuizOral）

该类是口语题（3/3），里面也涉及了交互和打分，不过模式相对稳定，资源用法与正文一致，这里需要注意分数的缓存和复位。



1. **过渡页（**PageQuizCover）

**该类是一个无足轻重的过渡页，桌面版，会有一只松鼠说“Start Quiz”，为了兼容这个页面配置，就在这一页展示一个松鼠动画，并配以音效。**

****

1. **结算页（ReportScene-BookReport）**

**该页主要体现用户的读书成绩，前期是通过服务器的反馈展示，容易被不完整读书操作影响，导致成绩不准确 ，现如今已经转移到客户端来计算成绩，不会耽误场景跳转、数据展示，并加入了坚果和面板动画，体验好多了。**

****

1. **教材结构**
2. **教材导读页（PageGuide）**

**该页面也采用窗帘式的过场动画，页面信息相对简单，一次性封装展示，跳转即删除。**

1. **教材互动页（PageText）**

**教材正文不涉及大图片，仅涉及音效、文字和角色。相关资源都是从配置文件解析而来，通过拼装，分别去下载头像和歌词文件，教材比较特殊的是，页面之间通过PageView控制，滚动翻页。**

1. **交互控制**

**交互主要存在于**

**a)原版正文页（播、读、听）**

**b)原版交互页（点、划）**

**c)原版口语页（播、读、听）**

**d)原版答题页（老丁熟悉）**

**e)教材正文页（播、读、听）**

**其基本原则就是**

1. **正在进行某一项操作，其他操作都会被拦截**
2. **正在进行某一项工作，关闭界面都会取消当前的工作**
3. **页面过渡过程中，禁止手动触发任何交互事件**
4. **影响页面展示的回调，成功返回之前，会设置禁用状态等待（比如按钮置灰）**
5. **场景交替**

**本App中主要存在四个场景**

1. **主场景（MainScene-MainPage）**

**这个场景主要是书架、个人中心、用户注册登录等外围模块，老丁比较熟悉**

1. **原版书的场景（BookScene）**

**这里面主要是原版书相关的类，细节参考第（一）条。**

1. **原版读书报告场景（**ReportScene）

**这里面是原版书的报告页，当初决定把它拆成独立场景出于两点考虑：**

1. **从书跳转报告页，最好有过场动画（类似于旋涡、翻转等），因为时间紧，没有做**
2. **书里面信息量很大，定义了很多数据结构，频繁加载图片、文字和音效，通过场景跳转做一次大回收。**

这个界面是很小，结合后期的算分调整，真的应该整合到BookScene里面，作为独立场景，确实有点浪费。

1. **教材书的场景（**TextScene**）**

**教材书跟原版书风格不一样，因此完全是两套数据和模式，独立成一个场景，理所当然**

1. **开场动画（**StartScene）

**开场是一个滚屏的广告页面，没有复杂业务**

1. **日志提交**

**日志大概分为几类：**

1. **开书日志，主要用于取得读书的event\_id，通知服务器开书时间（注：导读页不算）；**
2. **关书日志，通知服务器停止读书时间，并进行成绩结算**
3. **普通页面日志，不涉及交互和数据，比如版权页、Start Quiz页**
4. **正文读书日志，这里涉及的数据比较多，比如朗读、播放的时间统计、开页时间、得分情况等**
5. **口语答题日志，（3/3）**
6. **其他答题日志，（5/5）**
7. **教材书相关的日志，与原版书原理一样，参数也大同小异，不赘述**

**注：日志也分两种类型**

1. **敏感型，有必要信息需要在后面的交互中用到，便需要强制等待回调结果，比如开书日志和关书日志，开书时一定要拿到event\_id，关书时，比如提交成绩成功，否则都不能得到用户的真实数据。**
2. **不敏感型：不需要关心日志是否成功提交，因为强制等待日志提交，虽然提高了数据的准确性，但是大大的干扰用户的读书翻页体验，因此不强制等待，比如普通的翻页日志。**
3. **资源处理**
4. **图片原始资源（repertory/Art/backup）忽略，不重要**
5. **图片合图资源（repertory/Art/plist）**
6. **加密导表（repertory/Art/data\_table）一些用户相关的表格和静态文字**
7. **安卓打包（repertory/apk\_build）安卓打包脚本**
8. **辅助脚本（repertory/build）生成bin和预定义文件的ant脚本，具体命令参考repertory/build/build.xml中的结点，简单易用**
9. **UI编辑器（repertory/ReadingMateStudio）不添加和替换资源，很少动他，注意图片变动后刷新、导入和导出，另外注意图片替换时的大小是否一致，谨防贴图变形、UI失真。**
10. **UI编辑风格**

**由于目前的适配是局部坐标和大小定死，主体区域填充屏幕，随着后期业务的设计，难免会调整已有UI的结构和关系，因为有必要说明一下我设计UI的思路，进而方便其他人编辑更新。**

1. **主体分块**

layer //UI默认节点

│

panel\_root // UI根节点（设计尺寸）

├─ img\_bg // UI大背景（全尺寸居中）  
├─ panel\_center // UI主体面板（拉伸填充）

│ ├─ center\_0

│ └─ center\_1

├─ panel\_top // UI头面板（固定宽高）

│ ├─ top\_0

│ └─ top\_1

├─ panel\_bottom // UI尾面板（固定宽高）

│ ├─ bottom\_0

│ └─ bottom \_1

└─ panel\_popup // UI弹出层（全屏填充）  
 ├─ pop\_0

└─ pop\_1

**这个UI资源跟**BaseLayer.cpp配合使用，调用BaseLayer::resizeUI(Layout\* json\_root)方法，达到拉伸和适配的目的。通常一个UI分上中下三大块，其中上下两块固定宽高，并挂接到根节点的底端和顶端，其次中间一块自适应拉伸填充剩余空间。

1. **模板克隆**

**很多地方会用到TableView\ScrollView\PageView,自然就要设置模板添加到分页、滚动容器中。**

panel\_center // UI根节点  
├─ panel\_scroll // 滚动面板

│ ├─ scroll\_view //滚动容器

│ └─ scroll\_item //滚动模板

├─ panel\_Table // UI头面板

│ ├─ table\_view // table容器

│ └─ table\_item // table模板

└─ panel\_page // 分页面板

├─ page\_view // 分页容器

└─ page\_item //单页模板

**复杂容器都会对应一个专用的模板，而模板在使用前，我会将它removeFromParent，然后再对其clone添加到容器，最后在界面Exit的时候置空释放。**

****

**)**