ANIM控件重构详细设计

——IMG控件详细设计

# 设计目标

出于两方面考虑，我们需要重构ANIM控件：

一方面，整个控件体系都需要对象化改造，所以ANIM控件也需要从基类开始继承，以便重用父类的数据和逻辑，并且，在定制不同的ANIM控件的时候，可以从基础ANIM控件开始继承，以较小的开销完成定制。

另一方面，目前的ANIM是大而全的控件，所有的需求都被包含在一个控件中去完成，耦合度很高，而且，很多后加的需求点没有统一规划，以补丁的方式打上去，整个控件变成了大而全，结构和逻辑变得不清晰，难以维护。

因此我们要对ANIM控件进行重构，为了区别起见，我们把新控件叫做IMG控件。

IMG控件跟ANIM控件比，至少要做到两点：

1. 控件的对象化：可继承和被继承定制，
2. 逻辑清晰化：把各种逻辑分门归类，进行抽象，拆分等，使得逻辑简单，容易理解和构造——这也是对象化的必须的一步：抽象出一类对象。

# Img控件

为了整理Img控件的内部逻辑，我们把img控件拆分成两大部分:ImgView和ImgAgent

ImgAgent是图像控件的数据部分，承担图像数据的获取，维护，处理等功能。

ImgView是图像控件的视图部分，承担图像控件的控件功能。

ImgView拥有一个ImgAgent对象，以对该控件的图像数据进行维护。

简单的说就是Imgview通过ImgAgent获取想要的数据去显示，后者给前者提供不同类型的数据处理能力。

## 根据图像数据来源分类

整个控件都是围绕如何获取用户希望的数据去显示，所以，用户希望有哪些类型的数据是我们所面对的重要问题。

从图像的数据源角度，我们又可以对img agent进一步分类，可以分为三大类：

* Res ID：资源一般可以直接用于显示，也可以根据用户设置，保存数据到控件后再显示。
* data buffer：位图buffer和原数据buffer,前者可以直接用于显示，后者需要对buffer数据进行解码。
* 文件：根据不同的文件类型，在数据获取和处理上有所不同。

我们为这4类数据（data buffer分位图和原数据两大类）分开处理，逻辑独立互不干涉。根据用户的输入，即可区分出是哪类的数据。

### 资源：Res ID

资源可以直接用于显示，速度快，用户知道id后可以直接用于同步显示，

但是，我们的历史中有为资源分配内存，以存储资源图像数据的情况，需要保留这个支持能力。

### 内存位图

用户通过传给控件一个块bitmap数据，让控件负责显示。这里有所不同的在于这块bitmap的内存生命周期的维护。

### 内存原数据

用户会给控件设置一块原数据让控件负责显示，原数据需要解码后再显示，解码我们通过ref提供的接口完成，控件需要的是正确的解码原数据，然后保存解码结果到内存，以便显示的时候用。

### 图像文件

图像文件相对来说比上面的几类要复杂一些，不同的文件类型，不同的内存需求，速度需求，清晰度要求等要求比较多。

总的来说，也是需要解码文件数据，然后保存解码结果到内存，以便显示。

## ImgView和ImgAgent间的交互流程

1. 首先，ImgView需要告诉ImgAgent是哪一类图像数据（包括相应的参数：区域，透明，是否同步解码等）
2. 然后，ImgView等待ImgAgent解码结束。
3. ImgAgent启动解码，

* 如果是res ID和bitmap，不需要解码。
* 如果是内存原数据和文件需要先解码（具体的解码流程后面补充）

1. 解码结束，通知用户
2. ImgView调用ImgAgent的接口进行缩放和旋转处理
3. ImgView调用ImgAgent的接口获取目标数据用于显示，目标数据由ImgAgent根据自身状态给出，可能是：

* 资源直接显示
* Res ID buffer/Bitmap buffer/解码后的buffer,用于显示
* Zoom/rotate后的buffer用于显示。

1. 如果是动画，还需要启动播放动画（ImgAgent提供play/pause/resume/stop等接口，由ImgAgent起定时器）



## ImgView

ImgView从基类控件继承而来。作为基类控件的子控件，它自己需要完成的控件功能比较少，主要是并显示已有的图片（无论这张图片是怎么来的：解码/缩放/旋转等），图片的图像数据处理功能都委托自己所拥有的ImgAgent去完成。ImgView是图像处理接口（都由ImgAgent提供）的调用者。

### ImgView数据结构

typedef struct \_imgview\_obj\_tag

{

CTRLBASE\_OBJ\_T base\_ctrl;

IMG\_DATASRC\_E ImgDataSrcType;

IMG\_DATASRC\_T \*ImgDataSrc\_ptr;

IImgAgent \*pImgAgent; /\*!<img agent 接口\*/

BOOLEAN is\_focus; /\*!<窗口是否处于焦点状态\*/

BOOLEAN is\_low\_mem\_mode;

//可以通过这项来控制显示逻辑来实现，不需要单独的标志

BOOLEAN is\_pause\_without\_update;/\*!<动画是否暂时\*/

//动画播放方式:只播一帧/只播一遍/循环播放

ANIM\_DISP\_TYPE\_E anim\_disp;

uint32 anim\_interval; /\*!<每桢动画时间间隔\*/

//去掉????

BOOLEAN is\_form\_bg; /\*!<动画是否是form控件背景\*/

uint8 \*bg\_buf\_ptr; /\*!<是否备份背景\*/

BOOLEAN is\_auto\_adapt\_disp; /\*!<是否自适应显示,当图片宽大于长,横屏显示,否则竖屏显示\*/

//zoom的其余参数从agent获取

BOOLEAN is\_zoom; /\*!<是否缩放\*/

//旋转的其余参数从agent获取

BOOLEAN is\_rotate; /\*!<是否旋转\*/

//有必要区分是解码完成还是数据变化，用这连个消息代替is\_send\_disp\_ind这个标志

//而解码完成还是数据变化，可以agent的状态查询获取

//BOOLEAN is\_send\_disp\_ind; /\*!<是否已经发过MSG\_CTL\_ANIM\_DISPLAY\_IND消息,防止gif多次发送\*/

//从控件角度来说是需要发的，但是也许父亲不需要关注，所以这个标志可以取消，回调机制既可

//BOOLEAN is\_notify\_update; //当背景为GUI\_BG\_NONE时，是否无条件给父亲发送MSG\_NOTIFY\_UPDATE\*/

uint16 max\_resolution\_width; /\*!<解码后图片最大分辨率\*/

uint16 max\_resolution\_height; /\*!<解码后图片最大分辨率\*/

MMITHEME\_RATIO\_T zoom\_in\_ratio;

MMITHEME\_RATIO\_T zoom\_out\_ratio;

int16 display\_x; /\*!<图片显示在LCD上的起点x坐标,根据Align计算的显示位置\*/

int16 display\_y;/\*!<图片显示在LCD上的起点y坐标,根据Align计算的显示位置\*/

GUIANIM\_DISPLAY\_INFO\_T display\_info; /\*!<图片xianshi信息\*/

GUIANIM\_CALL\_BACK\_T CallBackFunc; /\*!<回调函数\*/

MMI\_IMAGE\_ID\_T default\_img; /\*!<默认img id\*/

GUI\_BG\_T error\_bg; /\*!<error bg\*/

GUI\_BORDER\_T frame; /\*!<img view frame\*/

}CTRLIMGVIEW\_OBJ\_T;

## ImgAgent

ImgAgent主要是根据输入的参数准备图像数据，并为各种类型的数据提供统一的访问接口，用户无需区分是资源数据，还是内存数据，还是文件图像，还为图像数据提供基本的放大/缩小，旋转功能，等方法。

### ImgAgent解码

#### 流程



#### 状态机



### ImgAgent数据结构

#### 属性

typedef enum

{

IMGAGENT\_PROP\_Start,

IMGAGENT\_PROP\_ImgDatSrc,

IMGAGENT\_PROP\_ImgInfo,

IMGAGENT\_PROP\_SynDecode,

IMGAGENT\_PROP\_ImgResolution,

IMGAGENT\_PROP\_BackGround,

IMGAGENT\_PROP\_ZoomInfo,

IMGAGENT\_PROP\_RotateInfo,

IMGAGENT\_PROP\_DispRect,

IMGAGENT\_PROP\_End

}IMGAGENT\_PROP\_E;

#### Img数据源

typedef enum

{

IMG\_DATASRC\_Unknow,

IMG\_DATASRC\_AnimID,

IMG\_DATASRC\_ImgID,

IMG\_DATASRC\_DataBuf\_Bitmap,

IMG\_DATASRC\_DataBuf\_Src,

IMG\_DATASRC\_File,

IMG\_DATASRC\_Max,

}IMG\_DATASRC\_E;

#### Img Buffer

typedef struct

{

IMG\_DATA\_FORMAT\_E data\_format;

uint32 width;

uint32 height;

uint32 \*pBuf;

uint32 BufSize;

uint32 offset;

}IMGDATABUFFER;

#### Img 缩/放信息

typedef struct

{

//缩放比例

uint16 radio1;

uint16 radio2;

//缩 or 放

//缩放源数据

//在缩放源数据中的起点和宽/高

uint32 origin\_x;

uint32 origin\_y;

uint32 origin\_width;

uint32 origin\_height;

//缩放目标数据的宽/高 = 缩放源数据的宽/高\*缩放比例

uint32 zoom\_width;

uint32 zoom\_height;

//缩放目标buffer;

IMGDATABUFFER zoom\_target\_buf;

}IMGZOOMINFO;

#### Img 旋转信息

typedef enum

{

GUIANIM\_ROTATE\_NONE, /\*!<保留位\*/

GUIANIM\_ROTATE\_90, /\*!<90度\*/

GUIANIM\_ROTATE\_180, /\*!<180度\*/

GUIANIM\_ROTATE\_270, /\*!<270度\*/

GUIANIM\_ROTATE\_MIRROR, /\*!<镜像\*/

GUIANIM\_ROTATE\_MAX /\*!<保留位\*/

} GUIANIM\_ROTATE\_E;

typedef struct

{

//旋转角度

/\*!< is\_Roate\_Mode = TRUE:模式旋转,硬件旋转,速度快;=FALSE,角度旋转，软件旋转，慢\*/

BOOLEAN is\_Roate\_Mode;

GUIANIM\_ROTATE\_E rotate\_mode; /\*!<旋转模式,硬件旋转,速度快\*/

uint16 rotate\_angle; /\*!<旋转角度0-360,软件旋转,速度慢\*/

//旋转目标buffer;

IMGDATABUFFER Roate\_target\_buf;

}IMGROATEINFO;

#### 动画

Img Agent要完成动画自动解码过程，对处于play状态的Anim(可能是gif或anim id),需要起动定时器，定时解码下一帧图像数据。

#### Img Agent

typedef struct \_ImgAgent\_tag

{

int nRefs;

IImgAgent mImgAgent;

/\*!<Img Agent的所有者，一般是img view控件，也可以是其它用户实体，但是异步解码的时候需要有接受post消息的能力\*/

uint32 owner;

IMG\_DATASRC\_E ImgDataSrcType;

IMG\_DATASRC\_T ImgDataSrc;

uint32 img\_handle;

IMG\_DECODE\_STATE\_E decode\_state;

//图片显示区大小,一般是用户的最大显示区域,比如控件的大小，这个区域的大小直接影响到解码分辨率?

uint16 user\_special\_width; /\*!<用户指定解码后图片宽度\*/

uint16 user\_special\_height; /\*!<用户指定解码后图片高度\*/

uint16 disp\_rect\_width; /\*!<图片显示区宽度\*/

uint16 disp\_rect\_height; /\*!<图片显示区高度\*/

//图像类型

IMG\_TYPE\_E ImgType;

//图像宽/高

uint32 ImgWidth;

uint32 ImgHeight;

uint16 max\_resolution\_width; /\*!<解码后图片最大分辨率\*/

uint16 max\_resolution\_height; /\*!<解码后图片最大分辨率\*/

uint16 ico\_frame\_index; //ico frame index

IMG\_THUMBNAIL\_T \*thumbnail\_ptr;

//同步/异步解码方式

BOOLEAN isSynDecode;

//是否解析清晰图

BOOLEAN isHighResolution;

//是否需要复制背景数据

BOOLEAN isCopyBg;

//背景

GUI\_BG\_T \*bg\_ptr;

//是否透明

BOOLEAN is\_transparent;

BOOLEAN is\_no\_miniature;

IMG\_MINIATURE\_T \*miniature\_ptr;

//资源img是否需要buffer来保存解码结果，只对ImgObjType\_ResID有效

BOOLEAN isBufferRes;

IMGDATABUFFER decoded\_buf;

IMGAGENT\_DEC\_RESULT\_T dec\_result;

//agent anim相关参数

IMGAGENT\_ANIM\_INFO\_T \*pImgAnim;

IMGROTATEINFO \*pImgRotate;

IMGZOOMINFO \*pImgZoom;

EventHandler\_Descriptor EvDescrip;

}IMGAGENT;

### ImgAgent的对外接口

#### 创建ImgAgent

BOOLEAN IMGAGENT\_New(IImgAgent \*\*ppi,void \*param);

#### 初始化ImgAgent,根据主题和用户参数初始化ImgAgent

#### 设置ImgAgent参数(属性)

参数有：

* 数据来源——资源ID，buffer，文件
* 源图像宽/高
* 当前缩放比例，缩/放区相对源图的起始点坐标，缩放区的宽/高，
* 同步/异步解码方式
* 是否解缩略图（miniature/thumbnail/压缩图）：1，只解缩略图，2，清晰图都解码，
* 是否复制背景数据,只针对Gif动画
* 控件背景
* 资源Img是否需要buffer解码结果，只对img id有效
* 动画相关参数
* Owner //待议

#### 获取ImgAgent参数

#### 放大/缩小

#### 旋转

#### 获取Agent显示数据

#### 直接显示Agent

#### 动画play/stop/resume/pause接口

#### 销毁ImgAgent

#### 释放ImgAgent所占据的内存。

### ImgAgent对象export事件

ImgAgent对象export事件：

1. 解码结束
2. 数据更新
3. 是否是动画，是的话，需要Imgview启动定时器（或者另一种做法，定时器放ImgAgent里，Imgview启动ImgAgent的play/pause/resume/stop等）

# 墙纸控件

墙纸控件的是经过改装的ImgAgent对象（墙纸ImgAgent）+ImgView构成，墙纸ImgAgent继承ImgAgent,然后对一些属性和方法进行墙纸化改造。

需要定制的属性和方法如下：

1. SaveWallpaperBitmap
2. FreeWallpaperTargetBuffer
3. …..