因为\_\_weak变量有一个特性就是它不会影响所指向对象的生命周期

控制器的生命周期 ①viewDidLoad ②viewWillAppear ③viewDidAppear，这三个方法背后真相如何？？

pthread\_getpecific和pthread\_setspecific实现同一个线程中不同函数间共享数据的一种很好的方式。

赋值对象指针时，所有权修饰符必须一致

NSError \*error = nil; // 默认修饰符是 \_\_strong

NSError \*\_\_strong\* pError = &error;

NSError \_\_weak \*error = nil;

NSError \*\_\_weak\* pError = &error;

NSError \_\_unsafe\_unretain \*error = nil;

NSError \*\_\_unsafe\_unretain\* pError = &error;

全局变量和实例变量是无法用\_\_autoreleasing来修饰的，

CG\_EXTERN CGContextRef \_\_nullable CGBitmapContextCreate(void \* \_\_nullable data, size\_t width, size\_t height, size\_t bitsPerComponent, size\_t bytesPerRow, CGColorSpaceRef cg\_nullable space, uint32\_t bitmapInfo)

创建一个位图上下文。 上下文绘制成宽度为“宽”像素和“高度”像素为高的位图。 每个像素的组件数量由`space'指定，它也可以指定目标颜色配置文件。 像素的每个分量的位数由“bitsPerComponent”指定。 每个像素的字节数等于'（bitsPerComponent \*分量数+7）/ 8'。 位图的每一行由'bytesPerRow'字节组成，它们必须至少为每个像素的'width \* bytes'字节; 另外，`bytesPerRow'必须是每个像素字节数的整数倍。 `data'，如果非NULL，则至少指向`bytesPerRow \* height'字节的内存块。 如果`data'为NULL，上下文数据将自动分配，并在上下文解除分配时释放。 `bitmapInfo'指定位图是否应该包含一个alpha通道以及如何生成，以及组件是浮点还是整数。

图片解压缩的过程其实就是将图片的二进制数据转换成像素数据的过程

## Bitmap Images and Image Masks

位图图像和图像掩码就像Quartz中的任何绘图基元。 Quartz中的图像和图像掩码均由CGImageRef数据类型表示。正如您在本章后面会看到的，您可以使用各种功能来创建图像。其中一些需要数据提供者或图像源来提供位图数据。其他功能通过复制图像或对图像执行操作来从现有图像创建图像。无论您如何在Quartz中创建位图图像，都可以将图像绘制到任何图形上下文。请记住，位图图像是以特定分辨率的位数组。如果您将位图图像绘制为与分辨率无关的图形上下文（例如PDF图形上下文），则位图会受到您创建分辨率的限制。

有一种方法可以通过调用函数CGImageMaskCreate来创建Quartz图像蒙板。您将看到如何在创建图像蒙版中创建一个。应用图像蒙版不是掩盖绘图的唯一方法。这些部分用颜色掩盖图像，用图像掩码掩盖图像以及通过剪切上下文掩盖图像讨论Quartz中可用的所有掩膜方法。

## About Bitmap Images and Image Masks

位图图像（或采样图像）是像素（或采样）的阵列。 每个像素代表图像中的单个点。 JPEG，TIFF和PNG图形文件是位图图像的示例。 应用程序图标是位图图像。 位图图像被限制为矩形形状。 但是通过使用alpha分量，它们可以呈现出各种形状，并且可以旋转和剪切，如图11-1所示。



位图中的每个样本包含指定颜色空间中的一个或多个颜色组件，以及一个指定alpha值以指示透明度的附加组件。 每个组件可以从1到多达32位。 在Mac OS X中，Quartz还提供对浮点组件的支持。 Mac OS X和iOS中支持的格式在“位图图形上下文支持的像素格式”中进行了描述。 ColorSync为位图图像提供色彩空间支持。

Quartz也支持图像掩码。 图像蒙板是一个位图，用于指定要绘制的区域，但不指定颜色。 实际上，图像蒙板充当模板以指定在页面上放置颜色的位置。 Quartz使用当前的填充颜色绘制图像蒙版。 图像蒙板可以具有1到8位的深度。

## Bitmap Image Information

Quartz支持各种图像格式，并且内置了几种常用格式的知识。在iOS中，格式包括JPEG，GIF，PNG，TIF，ICO，GMP，XBM和CUR。其他位图图像格式或专有格式要求您为Quartz指定图像格式的详细信息，以确保图像正确解释。您提供给函数CGImageCreate的图像数据必须以每个像素为单位进行交错处理，而不是以每个扫描线为基准。 Quartz不支持平面数据。

本节介绍与位图图像相关的信息。当您创建和使用Quartz图像（使用CGImageRef数据类型）时，您会看到一些Quartz图像创建功能要求您指定所有这些信息，而其他功能则需要这些信息的子集。您提供的内容取决于用于位图数据的编码，以及位图是表示图像还是图像掩码。

注意：为了在处理原始图像数据时获得最佳性能，请使用vImage框架。您可以使用vImageBuffer\_InitWithCGImage函数将图像数据从CGImageRef引用导入vImage。有关详细信息，请参阅加速发行说明。

Quartz在创建位图图像（CGImageRef）时使用以下信息：

位图数据源，可以是Quartz数据提供者或Quartz图像源。 Quartz 2D中的数据管理描述了这两者并讨论了提供位图数据源的功能。

可选的解码阵列（解码阵列）。

一个插值设置，它是一个布尔值，用于指定在调整图像大小时Quartz是否应应用插值算法。

一种呈现意图，指定如何映射位于图形上下文的目标颜色空间内的颜色。图像遮罩不需要此信息。请参阅设置渲染意图以获取更多信息。

图像尺寸。

像素格式，包括每个组件的位数，每像素的位数和每行的字节数（像素格式）。

对于图像，颜色空间和位图布局（颜色空间和位图布局）信息来描述alpha的位置以及位图是否使用浮点值。图像蒙版不需要这些信息。

### Decode Array

解码阵列将图像颜色值映射到其他颜色值，这对于诸如使图像去饱和或反转颜色等任务是有用的。该数组包含每个颜色分量的一对数字。当Quartz渲染图像时，它应用线性变换将原始分量值映射到适合目标色彩空间的指定范围内的相对数字。例如，RGB颜色空间中的图像的解码阵列包含六个条目，每个红色，绿色和蓝色颜色分量一对。

### Pixel Format

像素格式由以下信息组成：

每个组件的位数，这是像素中每个单独颜色组件的位数。对于图像蒙板，该值是源像素中有效掩蔽位的数量。例如，如果源图像是8位掩码，则指定每个组件8位。

每像素比特数，即源像素中的总比特数。此值必须至少为每个组件的位数乘以每个像素的组件数。

每行字节数。图像中每个水平行的字节数。

### Color Spaces and Bitmap Layout

为了确保Quartz正确解释每个像素的位，您必须指定：

* 位图是否包含Alpha通道。 Quartz支持RGB，CMYK和灰色空间。它同时也支持alpha通道或透明度，尽管alpha信息不适用于所有位图图像格式。当它可用时，alpha分量可以位于像素的最高有效位或最低有效位中。
* 对于具有alpha分量的位图，颜色分量是否已经乘以alpha值。预乘alpha描述了一个源颜色，其组件已经乘以了alpha值。通过消除每个颜色分量的多余乘法运算，预乘可以加快图像的渲染速度。例如，在RGB色彩空间中，使用预乘alpha生成图像会消除图像中每个像素的三次乘法运算（红色时间alpha，绿色时间alpha和蓝色时间alpha）。
* 样本的数据格式 - 整数或浮点值。

当您使用函数CGImageCreate创建图像时，您需要提供一个类型为CGImageBitmapInfo的bitmapInfo参数来指定位图布局信息。以下常量指定alpha分量的位置以及颜色分量是否预乘：

kCGImageAlphaLast - alpha分量存储在每个像素的最低有效位中，例如RGBA。

kCGImageAlphaFirst - alpha分量存储在每个像素的最高有效位中，例如ARGB。

kCGImageAlphaPremultipliedLast - alpha分量存储在每个像素的最低有效位中，并且颜色分量已经乘以该alpha值。

kCGImageAlphaPremultipliedFirst - alpha分量存储在每个像素的最高有效位中，并且颜色分量已经乘以该alpha值。

kCGImageAlphaNoneSkipLast-没有alpha组件。如果像素的总尺寸大于颜色空间中颜色分量数量所需的空间，则忽略最低有效位。

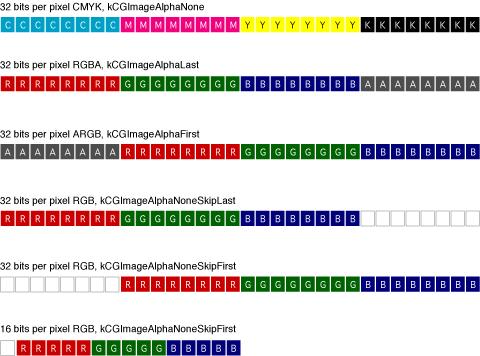
kCGImageAlphaNoneSkipFirst-没有alpha组件。如果像素的总大小大于色彩空间中色彩分量数量所需的空间，则忽略最高有效位。

kCGImageAlphaNone-相当于kCGImageAlphaNoneSkipLast。

您可以使用常量kCGBitmapFloatComponents来指示使用浮点值的位图格式。 对于浮点格式，您可以将此常量与前一列表中的适当常量进行逻辑或运算。 例如，对于使用预乘alpha的128位/像素浮点格式，alpha位于每个像素的最低有效位中，则向Quartz提供以下信息：kCGImageAlphaPremultipliedLast| kCGBitmapFloatComponents

图11-2可视化地描述了如何在使用16位或32位整数格式的CMYK和RGB色彩空间中表示像素。 32位整数像素格式使用每个组件8位。 16位整数格式使用每个组件5位。 Quartz 2D还支持每个组件使用32位的128位浮点像素格式。 图中未显示128位格式。

图11-2石英2D中CMYK和RGB颜色空间的32位和16位像素格式



创建图像

表11-1列出了Quartz提供的用于创建CGImage对象的函数。图像创建功能的选择取决于图像数据的来源。最灵活的功能是CGImageCreate。它从任何类型的位图数据创建一个图像。但是，这是最复杂的功能，因为您必须指定所有位图信息。要使用此功能，您需要熟悉位图图像信息中讨论的主题。

如果您想从使用PNG或JPEG等标准图像格式的图像文件创建CGImage对象，最简单的解决方案是调用函数CGImageSourceCreateWithURL来创建图像源，然后调用CGImageSourceCreateImageAtIndex函数以创建一个来自图像源中特定索引处的图像数据。如果原始图像文件只包含一个图像，则提供0作为索引。如果图像文件格式支持包含多个图像的文件，则需要将索引提供给相应的图像，请记住索引值从0开始。

如果您已将内容绘制到位图图形上下文中并想将该图形捕捉到CGImage对象，请调用函数CGBitmapContextCreateImage。

几个功能是对现有图像进行操作的实用程序，可以进行复制，创建缩略图或从较大的一部分创建图像。无论您如何创建CGImage对象，都可以使用函数CGContextDrawImage将图像绘制到图形上下文中。请记住，CGImage对象是不可变的。当你不再需要一个CGImage对象时，通过调用函数CGImageRelease来释放它。

表11-1用于创建图像的功能

|  |  |
| --- | --- |
| **Function** | **Description** |
| CGImageCreate | A flexible function for creating an image. You must specify all the bitmap information that is discussed in [Bitmap Image Information](https://developer.apple.com/library/content/documentation/GraphicsImaging/Conceptual/drawingwithquartz2d/dq_images/dq_images.html#//apple_ref/doc/uid/TP30001066-CH212-BBCJBICF). |
| CGImageSourceCreateImageAtIndex | Creates an image from an image source. Image sources can contain more than one image. See [Data Management in Quartz 2D](https://developer.apple.com/library/content/documentation/GraphicsImaging/Conceptual/drawingwithquartz2d/dq_data_mgr/dq_data_mgr.html#//apple_ref/doc/uid/TP30001066-CH216-TPXREF101) for information on creating an image source. |
| CGImageSourceCreateThumbnailAtIndex | Creates a thumbnail image of an image that is associated with an image source. See [Data Management in Quartz 2D](https://developer.apple.com/library/content/documentation/GraphicsImaging/Conceptual/drawingwithquartz2d/dq_data_mgr/dq_data_mgr.html#//apple_ref/doc/uid/TP30001066-CH216-TPXREF101) for information on creating an image source. |
| CGBitmapContextCreateImage | Creates an image by copying the bits from a bitmap graphics context. |
| CGImageCreateWithImageInRect | Creates an image from the data contained within a sub-rectangle of an image. |
| CGImageCreateCopy | A utility function that creates a copy of an image. |
| CGImageCreateCopyWithColorSpace | A utility function that creates a copy of an image and replaces its color space. |

以下部分讨论如何创建：

来自现有图像的子图像

来自位图图形上下文的图像

您可以咨询这些来源获取更多信息：

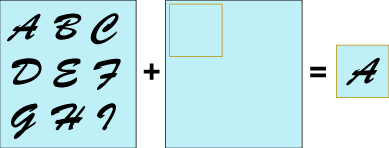
Quartz 2D中的数据管理讨论如何读取和写入图像数据。

CGImage参考，CGImageSource参考和CGBitmapContext参考了解表11-1中列出的功能及其参数的更多信息。

从较大图像的一部分创建图像

函数CGImageCreateWithImageInRect允许您从现有的Quartz图像创建子图像。 图11-3举例说明了通过提供指定字母“A”的位置的矩形从大图中提取包含字母“A”的图像。

图11-3从较大图像创建的子图像

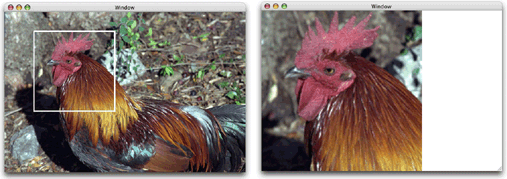


函数CGImageCreateWithImageInRect返回的图像保留对原始图像的引用，这意味着您可以在调用此函数后释放原始图像。

图11-4显示了提取图像的一部分以创建另一个图像的另一个示例。在这种情况下，从较大的图像中提取公鸡的头部，然后将其绘制到比子图像更大的矩形，从而有效地放大图像。

清单11-1显示了创建并绘制子图像的代码。函数CGContextDrawImage将公鸡的头部绘制到的矩形的尺寸是提取的子图像尺寸的两倍。列表是一个代码片段。您需要声明适当的变量，创建公鸡图像，并处理公鸡图像和公鸡头部子图像。由于代码是一个片段，因此不会显示如何创建图像所绘制的图形上下文。你可以使用任何你喜欢的图形上下文。有关如何创建图形上下文的示例，请参阅图形上下文。

图11-4一张图像，一张从其中取出并绘制的子图像，因此被放大



**Listing 11-1**  Code that creates a subimage and draws it enlarged

|  |
| --- |
| myImageArea = CGRectMake (rooster\_head\_x\_origin, rooster\_head\_y\_origin, |
| myWidth, myHeight); |
| mySubimage = CGImageCreateWithImageInRect (myRoosterImage, myImageArea); |
| myRect = CGRectMake(0, 0, myWidth\*2, myHeight\*2); |
| CGContextDrawImage(context, myRect, mySubimage); |

### Creating an Image from a Bitmap Graphics Context

要从现有位图图形上下文创建图像，请按如下方式调用函数CGBitmapContextCreateImage：

|  |
| --- |
| CGImageRef myImage; |
| myImage = CGBitmapContextCreateImage (myBitmapContext); |

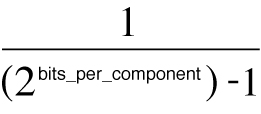
该函数返回的CGImage对象是通过复制操作创建的。因此，对位图图形上下文所做的任何后续更改都不会影响返回的CGImage对象的内容。在某些情况下，复制操作实际上遵循写时复制语义，因此仅当位图图形上下文中的基础数据被修改时才会发生实际的位复制。在向位图图形上下文中执行附加绘图之前，您可能需要使用生成的图像并将其释放，以便避免数据的实际物理副本。有关如何创建位图图形上下文的示例，请参阅创建位图图形上下文。

## Creating an Image Mask

Quartz位图图像蒙版的使用方式与艺术家使用丝网印刷的方式相同。位图图像蒙版确定如何传输颜色，而不是使用哪种颜色。图像遮罩中的每个样本值指定当前填充颜色在特定位置被遮罩的量。样本值指定掩码的不透明度。较大的值表示较大的不透明度，并指定Quartz绘制较少颜色的位置。您可以将样本值视为逆阿尔法值。值1是透明的，0是不透明的。

图像掩模是每个组件1个，2个，4个或8个位。对于1位掩码，采样值1指定掩码的当前填充颜色部分。样本值为0指定蒙版部分，用于显示蒙版被绘制时图形状态的当前填充颜色。您可以将1位掩码视为黑白;样品要么完全阻挡油漆，要么完全允许油漆。

每个组件具有2,4或8位的图像掩码表示灰度值。使用以下公式将每个组件映射到0到1的范围：



例如，一个4位掩码值的范围从0到1，增量为1/15。 0或1的组件值表示极值 - 完全阻止绘画并完全允许绘画。 介于0和1之间的值允许使用公式1 - MaskSampleValue进行局部绘制。 例如，如果8位掩码的样本值缩放到0.7，则颜色被绘制为具有（1-0.7）的α值，即0.3。

CGImageMaskCreate函数根据您提供的位图图像信息创建一个Quartz图像遮罩，并在位图图像信息中进行讨论。 除了不提供色彩空间信息，位图信息常量或渲染意图之外，您提供的用于创建图像蒙版的信息与创建图像所提供的信息相同，如通过查看函数 清单11-2中的原型。

清单11-2函数CGImageMaskCreate的原型

|  |
| --- |
| CGImageRef CGImageMaskCreate ( |
| size\_t width, |
| size\_t height, |
| size\_t bitsPerComponent, |
| size\_t bitsPerPixel, |
| size\_t bytesPerRow, |
| CGDataProviderRef provider, |
| const CGFloat decode[], |
| bool shouldInterpolate |
| ); |

## Masking Images

遮罩技术可以通过控制图像的哪些部分被绘制来产生许多有趣的效果。 您可以：

将图像蒙版应用于图像。 您还可以使用图像作为蒙版来实现与应用图像蒙版相反的效果。

使用颜色遮盖图像的某些部分，其中包括称为色度键遮罩的技术。

将图形上下文剪切到图像或图像蒙板上，当Quartz将内容绘制到剪切的上下文中时，蒙版将有效地蒙版图像（或任何类型的图形）。