

# 第2章 数字图像表示 及其处理

本章重点：

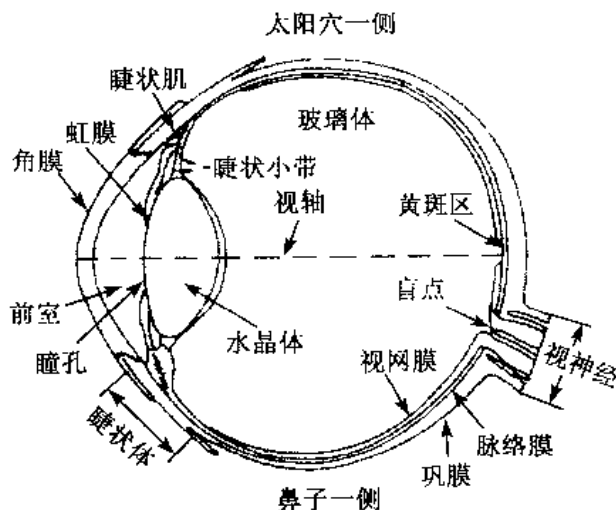
- 图像的数字化以及数字化图像表示方法
- 图像存储格式
- 用VC++实现图像的读、写以及显示的编程

# 第二章 数字图像表示及处理

- ▶ 2.1 人眼成像过程
- ▶ 2.2 简单的图像形成模型
- ▶ 2.3 图像的数字化的
- ▶ 2.4 数字图像的基本类型
- ▶ 2.5 数字图像的基本文件格式

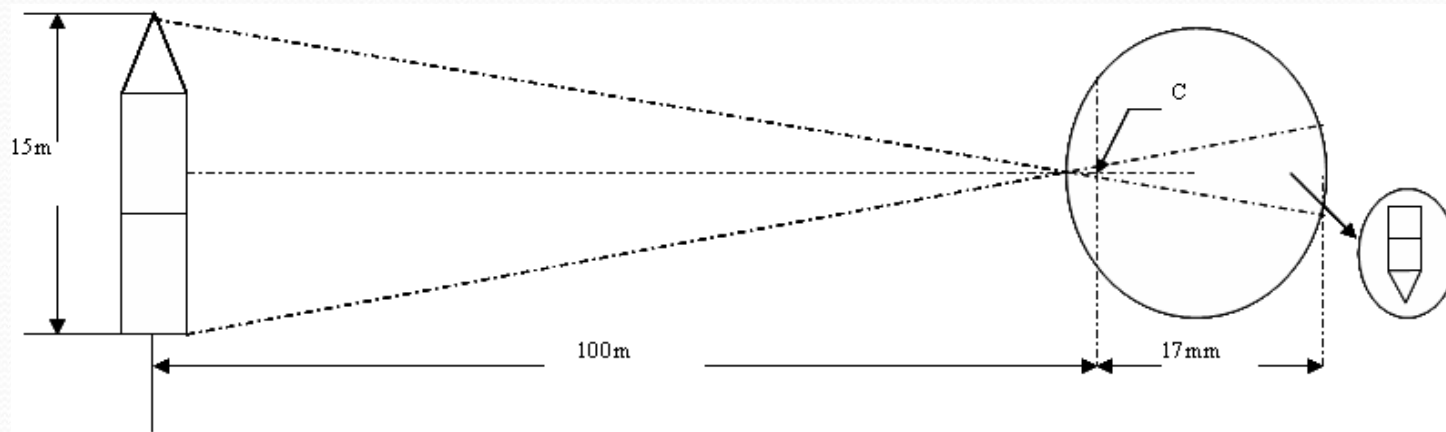
## 2.1 人眼成像过程

- ▶ 人眼为半径约 20mm 球状器官。
- ▶ 三层薄膜包围：
  - ▶ 1) 最外层**硬蛋白质膜**，前方1/6**角膜**，光线入眼。  
其余5/6白色不透明组织，称为**巩膜**，巩固和保护整个眼球
  - ▶ 2) 中间一层，由**虹膜**和**脉络膜**组成，虹膜中间为**瞳孔**。  
瞳孔，控制进入的光量大小，相当于相机光圈。
  - ▶ 3) 最内一层为**视网膜**，表面分布有大量**光敏细胞**。



瞳孔后有一个扁球形的透明水晶体，相当于可变焦透镜，睫状肌进行收缩调节，使景象始终能刚好地聚焦于黄斑区。

。



用眼睛看建筑物侧面的图解，C点是晶状体的光心



## 2.2简单的图像形成模型

- ▶ 图像是物体辐射能量的空间分布，是空间坐标、时间和波长函数，即  $I(x,y,z,\lambda,t)$
- ▶ 图像空间坐标 $z$ 、波长 $\lambda$ 和时间变量 $t$ 去除，图像用二维函数  $f(x,y)$ 表示：

$$f(x,y)=i(x,y)r(x,y)$$

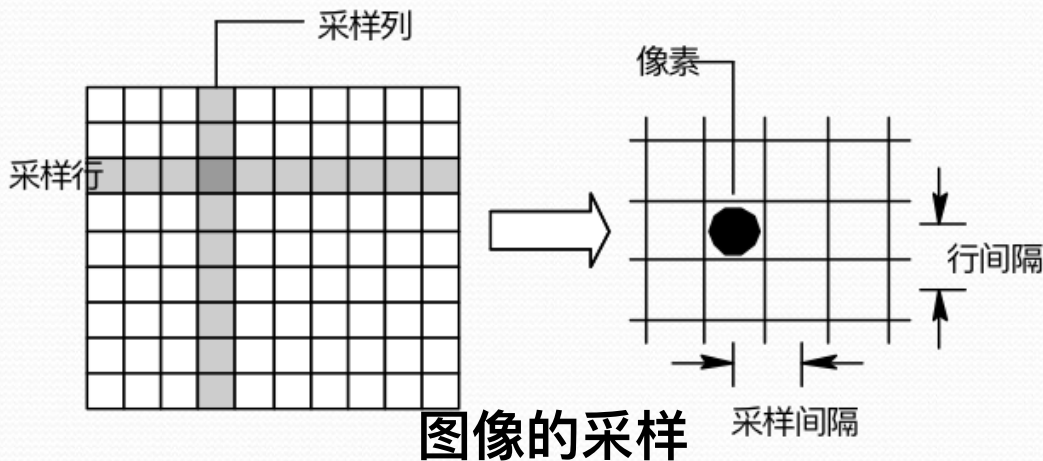
$i(x,y)$ 为照射源，而 $r(x,y)$ 为成像物体特性。

## 2.3 图像的数字化

- ▶ **数字图像**可理解为，对二维函数 $f(x,y)$ **采样和量化**(即离散处理)后得到的图像，常用二维矩阵表示一幅数字图像
- ▶ 图像数字化，生成一个二维矩阵的过程。
- ▶ 数字化包括三个步骤：扫描、采样和量化。

## 2.3.1 采样

- ▶ 采样：离散化图像空间坐标，决定图像的空间分辨率。
- ▶ 网格划分图像，每小格计算模拟图像的亮度平均值；
- ▶ 方格交叉点处，图像亮度值作交叉点的值。





- ▶ 图像采样，每行（横向）像素为N个，每列（纵向）像素为M个，图像大小为M×N个像素。
- ▶  $f(x,y)$ 构成一个M×N实数矩阵：

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \vdots & & & \\ f(M-1,0) & & & f(M-1,N-1) \end{bmatrix}$$

- ▶ 每元素为图像  $f(x,y)$  的离散采样值，称之为像元或像素



## 2.3.2分辨率

- ▶ 分辨率：指映射到图像平面上的单个像素的尺寸。

单位：像素/英寸，像素/厘米

(如：扫描仪的指标 300dpi)

- ▶ 精确测量和再现一幅的图像所必需的像素个数。

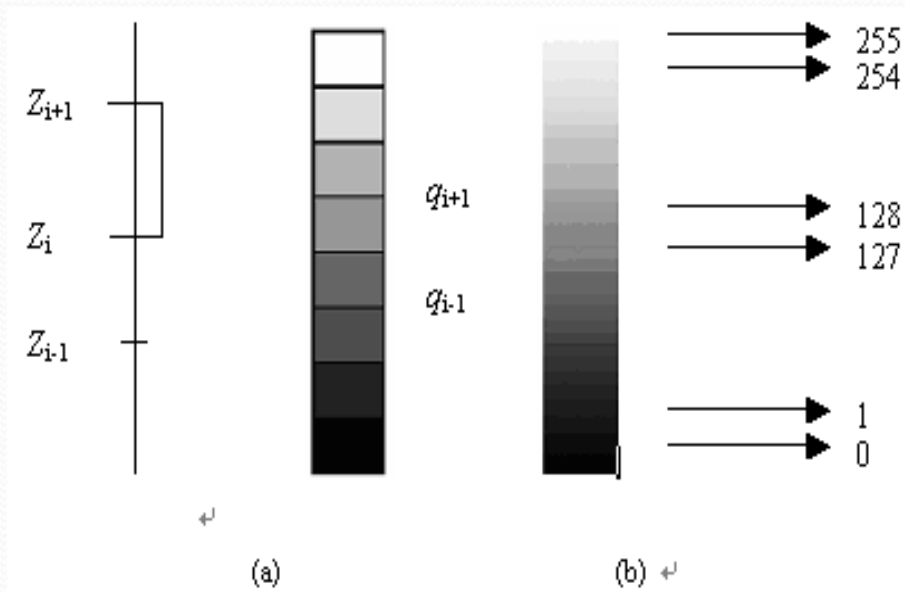
单位：像素\*像素

(如：数码相机指标30万像素 (640\*480) )

## 2.3.3 量化

- ▶ 采样像素灰度值，从模拟量到离散量的转换，称为**图像灰度的量化**。
- ▶ 图像坐标的离散化量化，决定图像的灰度分辨率。
- ▶ 量化方法：分层量化、均匀量化和非均匀量化。
- ▶ **分层量化**，**连续灰度值**分成多个层次。
- ▶ **均匀量化**，灰度层次从最暗至最亮，均匀分为有限个层次
- ▶ 采用不均匀分层，为非均匀量化。

# 量化示意图



(a) 量化

(b) 量化为8 bit

$Z$ 为连续灰度值， $q$ 为量化后的整数值



# 图像量化实例



(a)



(b)

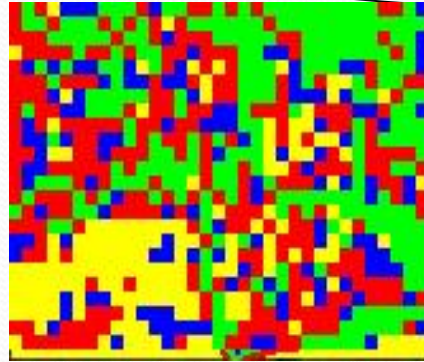
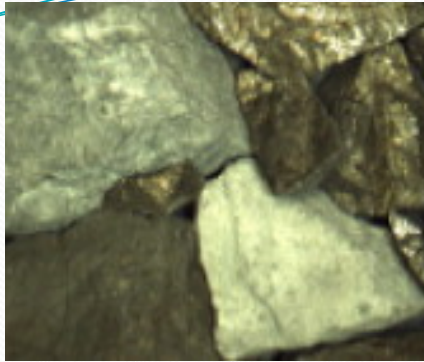
18	17	19	17	21	29	45	59	65	59	58	66	67	61	69	60
22	20	20	17	19	25	51	65	82	90	84	74	73	78	57	56
27	23	23	18	17	21	42	47	66	90	97	90	84	86	58	61
28	25	24	21	19	21	24	24	30	50	77	95	93	84	79	77
26	24	24	23	22	23	26	38	37	28	43	77	93	88	102	91
24	20	20	21	22	23	40	68	75	47	29	48	80	97	109	97
23	16	15	17	19	19	36	55	73	68	44	33	58	92	108	103
23	14	11	13	15	15	16	12	36	69	64	35	42	77	108	110
18	21	20	19	16	7	8	14	31	60	63	30	32	79	106	118
19	18	13	13	18	17	5	11	23	48	57	38	45	84	122	128
21	18	10	13	28	35	29	42	51	53	46	40	63	104	140	137
22	24	15	18	35	46	58	77	82	60	35	42	90	140	152	140
21	27	19	21	35	44	46	53	52	38	36	72	131	172	164	146
20	26	24	31	46	54	28	14	13	31	70	128	174	187	180	156
20	26	36	60	88	101	74	55	63	99	138	178	196	186	190	163
22	28	50	91	133	152	149	140	160	189	197	201	198	182	192	165

- (a) 256级灰度图象
- (b) 子图
- (c) 子图对应的量化数据

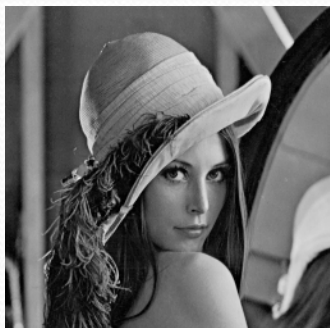


# 采样点数和量化级数的关系

- ▶ 一幅图像，量化级数一定时，采样点数对图像质量有着显著的影响。**采样点越多**，图像质量越好；采样点减少时，图像块状效应就逐渐明显。
- ▶ 图像采样点数一定，不同量化级数影响图像。**量化级数越多**，图像质量越好，当量化级数越少时，图像质量越差。
- ▶ 量化级数最小情况是二值图像，图像会出现假轮廓。



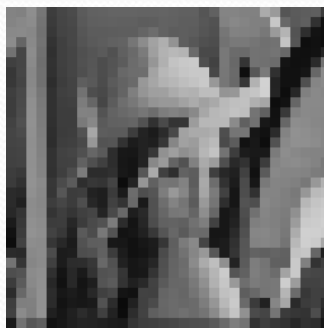
# 采样点数与图像质量之间的关系



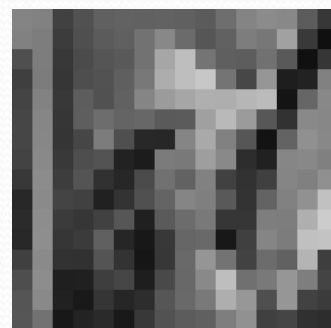
(a)



(b)



(c)



(d)

- a) 采样点 $256 \times 256$ 时的图像
- b) 采样点 $64 \times 64$ 时的图像
- c) 采样点 $32 \times 32$ 时的图像
- d) 采样点 $16 \times 16$ 时的图像



# 量化级数与图像质量之间的关系



(a)



(b)



(c)

(a) 量化为2级的Lena图像

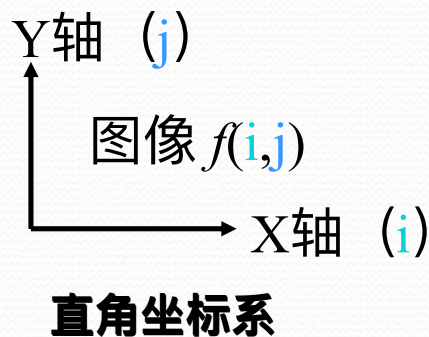
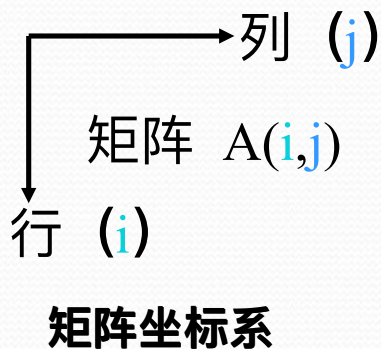
(b) 量化为16级的Lena图像

(c) 量化为256级的Lena图像



## 2.4 数字图像的描述和基本类型

- ▶ 矩阵按照行列定位数据，图像在平面上定位数据，有一个坐标系定义上的特殊性。
- ▶ 为了编程方便起见，以矩阵坐标系来定义图像的坐标。



- ▶ 计算机一般采用两种方式存储静态图像：  
位映射（Bitmap），即位图存储模式；  
向量处理（Vector），也称矢量存储模式。
- ▶ **位图**也称为栅格图像，通过许多像素点表示一幅图像，  
每个像素具有颜色属性和位置属性。
- ▶ **矢量图**只存储图像内容的轮廓部分，而不是存储图像数据的每个点。

## 2.4.1 二值图像

- ▶ 二值图像也叫黑白图像，就是图像像素只存在0,1两个值。



二进制的lenna图像



## 2.4.2灰度图像

- ▶ 灰度图像仅包含灰度级的图像，如64级，256级等。
- ▶ 像素灰度级用8bit表示，每个像素取值是256种灰度中的一种，即每个像素的灰度值为0到255中的一个。
- ▶ 通常，用0表示黑，255表示白，从0到255亮度逐渐增加。





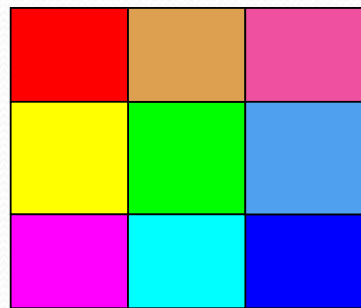
## 2.4.3索引图像

- ▶ 真彩色 $2^{24}=1670$ 万
- ▶ 索引图像：据索引**颜色序号**找到像素的实际颜色，索引图像读入计算机时，索引颜色被存储到调色板。
- ▶ 调色板是**颜色表**，颜色以红、绿、蓝三色组合来表示。
- ▶ 调色板的单元个数与图像颜色数一致，256色图像有256个索引颜色，相应调色板有256个单元。

颜色索引		红	绿	蓝
0	0	0	0	
1	128	0	0	

## 2.4.4 RGB彩色图像

- ▶ RGB图像是一类图像的总称。
- ▶ 图像不使用单独的调色板，像素颜色由存储在相应位置的红、绿、蓝颜色分量决定。
- ▶ RGB图像是24位图像，红、绿、蓝分量分别占用8位，理论上可以包含16M种不同的颜色。
- ▶ 真彩色 $2^{24}=1670$ 万
- ▶ 彩色图像不能用一个矩阵描述，用三个矩阵同时来描述。



$$R = \begin{bmatrix} 255 & 240 & 240 \\ 255 & 0 & 80 \\ 255 & 0 & 0 \end{bmatrix} \quad G = \begin{bmatrix} 0 & 160 & 80 \\ 255 & 255 & 160 \\ 0 & 255 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 80 & 160 \\ 0 & 0 & 240 \\ 255 & 255 & 255 \end{bmatrix}$$



## 2.5 数字图像基本格式

- ▶ 每种图像文件均有一个文件头，文件头后才是图像数据。
- ▶ 文件头内容一般包括文件类型、文件制作者、制作时间、版本号、文件大小等内容。
- ▶ 图像文件制作还涉及图像文件的压缩方式和存储效率等。
- ▶ 常用图像文件存储格式有BMP文件、JPG文件、PCX文件、TIFF文件以及GIF文件等。



## 2.5.1 BMP图像文件格式

文件部分↵	属性↵	说明↵
↵ BTPMAPFILEHEADER↵ (位图文件头)↵	<u>BfType</u> ↵	文件类型，必须是 0x424D,即字符串“BM”↵
	<u>bfSize</u> ↵	指定文件大小，包括这 14 个字节↵
	<u>bfReserved1</u> ↵	保留字，不用考虑↵
	<u>bfReserved2</u> ↵	保留字，不用考虑↵
	<u>bfOffBits</u> ↵	从文件头到实际位图数据的偏移字节数↵
↵ ↵ ↵ ↵ BITMAPINFOHEADER↵ (位图信息头)↵	<u>bfSize</u> ↵	该结构的长度，为 40↵
	<u>biWidth</u> ↵	图像的宽度，单位是像素↵
	<u>biHeight</u> ↵	图像的高度，单位是像素↵
	<u>biplanes</u> ↵	位平面数，必须是 1，不用考虑↵
	<u>biBitCount</u> ↵	指定颜色位数，1 为二色，4 为 16 色，8 为 256 色，16、24、32 为真彩色↵
	<u>biCompression</u> ↵	指定是否压缩，有效值为 BI_RGB、BI_RLE8、BI_RLE4、BI_BITFIELDS↵
	<u>biSizeImage</u> ↵	实际的位图数据占用的字节数↵
	<u>biXPelsPerMeter</u> ↵	目标设备水平分辨率，单位是每米的像素数↵
	<u>biYPelsPerMeter</u> ↵	目标设备垂直分辨率，单位是每米的像素数↵
	<u>biClrUsed</u> ↵	实际使用的颜色数，若该值为 0，则使用颜色数为 2 的 <u>biBitCount</u> 次方种↵
	<u>biClrImprotant</u> ↵	图像中重要的颜色数，若该值为 0，则所有的颜色都是重要的↵
↵  Palette↵ (调色板)↵	<u>rgbBlue</u> ↵	该颜色的蓝色分量↵
	<u>rgbGreen</u> ↵	该颜色的绿色分量↵
	<u>rgbRed</u> ↵	该颜色的红色分量↵
	<u>rgbReserved</u> ↵	保留值↵
↵ <u>ImageData</u> ↵ (位图数据)↵	像素按行优先顺序排列，每一行的字节数必须是 4 的整倍数↵	↵

- 第一部分，**位图文件头** BITMAPFILEHEADER，结构体定义如下：

```
typedef struct tagBITMAPFILEHEADER {  
    WORD    bfType; //文件类型  
    DWORD   bfSize; //文件大小  
    WORD    bfReserved1;  
    WORD    bfReserved2;  
    DWORD   bfOffBits; //文件头到数据的偏移字节  
} BITMAPFILEHEADER;
```

结构长度固定，为**14**个字节（WORD为无符号16位二进制整数，DWORD为无符号32位二进制整数）。

▶ 第二部分，**位图信息头**BITMAPINFOHEADER，定义如下：

```
typedef struct tagBITMAPINFOHEADER{  
    DWORD    biSize;  
    LONG     biWidth;  
    LONG     biHeight;  
    WORD     biPlanes; //必须是1  
    WORD     biBitCount;  
    DWORD    biCompression;  
    DWORD    biSizeImage; //实际位图数据字节数  
    LONG     biXPelsPerMeter;  
    LONG     biYPelsPerMeter;  
    DWORD    biClrUsed;  
    DWORD    biClrImportant;  
} BITMAPINFOHEADER;
```

**LONG为32位，结构长度固定为40个字节**

- ▶ 第三部分，调色板(Palette)。真彩色图像不需要调色板,调色板实际上是一个数组。数组中每个元素占4个字节，其定义如下：

```
typedef struct tagRGBQUAD{  
    BYTE  rgbBlue;           //该颜色的蓝色分量值  
    BYTE  rgbGreen;          //该颜色的绿色分量值  
    BYTE  rgbRed;             //该颜色的红色分量值  
    BYTE  rgbReserved;       //保留值  
} RGBQUAD;
```



- ▶ 第四部分，实际图像数据。
- ▶ 用到调色板的位图，图像数据该像素颜色在调色板中的索引值；
- ▶ 真彩色图像，图像数据是实际的R、G、B值。
- ▶ 2色位图，用1位表示该像素颜色（0表示黑，1表示白），一个字节可表示8个像素。
- ▶ 16色位图，用4位表示像素颜色，1个字节可以表示2个像素。
- ▶ 256色位图，一字节可表示1个像素。

## 2.5.2 TIFF图像文件格式

### ▶ 标记图像文件格式TIFF

Tag Image File Format), 图像文件格式中最复杂的一种, 也是目前流行的图像文件交换标准之一。

- ▶ TIFF格式文件考虑扩展性、方便性和可修改性, 非常复杂, 要求用更多的代码, 导致文件读写速度慢, **TIFF代码**很长。
- ▶ TIFF文件由**文件头**、**参数指针表与参数域**、**参数数据表**和**图像数据**4部分组成。

## (1) 文件头 ↵

表 2-3 TIF 文件文件头结构↵

0~1 字节↵	说明字节顺序，合法值是：↵ 0X4949，表示字节顺序由低到高；0X4D4D 表示字节顺序由高到低↵	↵
2~3 字节↵	TIFF 版本号，总为 0X2A。↵	↵
4~7 字节↵	指向第一个参数指针表的指针↵	↵

## (2) 参数指针 ↵

0~1 字节↵	参数域的个数 $n$ ↵	↵
2~13 字节↵	第一个参数块↵	↵
14~25 字节↵	第二个参数块↵	↵
...↵	...↵	↵
$2+n*12\sim6+n*12$ 字节↵	为 0 或指向下个参数指针表的偏移↵	↵

参数指针，一个2字节整数和其后的一系列12字节参数域构成。

### (3) 参数块结构 ↵

表 2-4 TIF 文件参数块结构 ↵

0~1 字节 ↵	参数码，为 254 到 321 间的整数 ↵
2~3 字节 ↵	参数类型：1 为 BYTE，2 为 CHAR；3 为 SHORT；4 为 LONG；5 为 RATIONAL。 ↵
4~7 字节 ↵	参数长度或参数项个数 ↵
8~11 字节 ↵	参数数据，或指向参数数据的指针 ↵



## 2.5.3 GIF图像文件格式

- ▶ GIF (Graphics Interchange Format) 文件是图形交换文件格式，网上小动画。
- ▶ 该形式的文件为不同系统平台交流和传输图像提供方便，Web服务常用的文件格式，HTML文档中的索引颜色图像。图像最大不能超过64M，颜色最多为256色。
- ▶ GIF图像文件采取LZW压缩算法，存储效率高，支持多幅图像定序或覆盖，交错多屏幕绘图以及文本覆盖。
- ▶ GIF数据流设计传输格式，不作为文件的**存储格式**。

- ▶ GIF有五个主要部分以固定顺序出现，所有部分均由一个或多个块(block)组成。
- ▶ 每个块第一个字节中存放标识码或特征码标识。这些部分的顺序为：文件标志块、逻辑屏幕描述块、可选的“全局”色彩表块(调色板)、各图像数据块（或专用的块）以及尾块（结束码）。

表 2-5 GIF 图像文件格式

文件标志块	Header	识别标识符“GIF”和版本号(“87a”或“89a”)	
逻辑屏幕描述块	Logical Screen Descriptor	定义包围所有后面图像的一个图像平面的大小,纵横尺寸以及颜色深度,以及是否存在全局色彩表	
全局色彩表	Global Color Table	色彩表的大小由该图像使用的颜色数决定,若表示颜色的二进制数为 111,是 7,则颜色数为 $2^{n+1}$	
图像数据块	Image Descriptor	图像描述块	可重复 $n$ 次
	Local Color Table	局部色彩表(可重复 $n$ 次)	
	Table Based Image Data	表式压缩图像数据	
	Graphic Control Extension	图像控制扩展块	
	Plain Text Extension	无格式文本扩展块	
	Comment Extension	注释扩展块	
	Application Extension	应用程序扩展块	
尾块	GIF Trailer	值为 3B(十六进制数),表示数据流已结束	

## 2.5.4 PCX图像文件格式

- ▶ PCX文件格式由ZSoft公司设计，绘图软件生成的图像0000000000000000。
- ▶ PCX支持256种颜色，结构较简单，存取速度快，压缩比适中，适合于一般软件的使用。
- ▶ PCX格式支持RGB、索引颜色、灰度和位图颜色模式，图像颜色的位数可以是1、4、8或24。
- ▶ PCX图像文件由三个部分组成：文件头、图像数据和256色调色板。



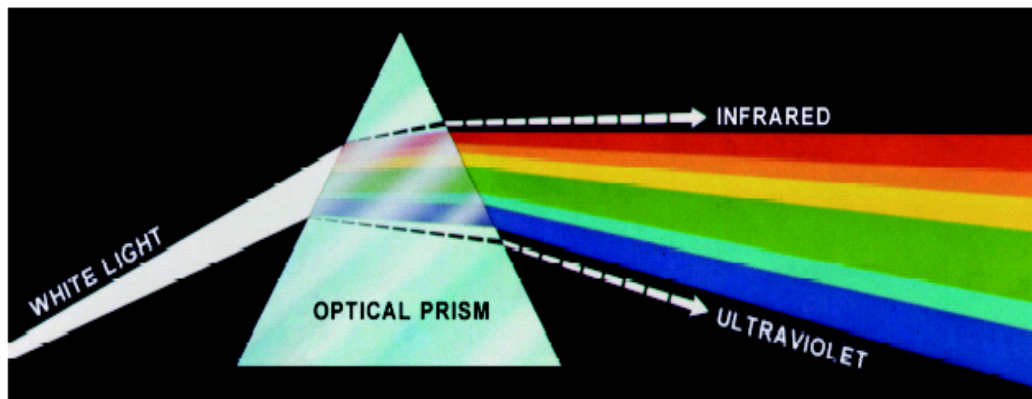
## 2.5.5 JPEG图像格式

- ▶ JPEG(Joint Photographer's Experts Group), 静止图像压缩标准, 是第一个国际数字图像压缩标准, 解决专业摄影师所遇到的图像信息过于庞大的问题。
- ▶ JPEG格式支持24位颜色, 保留照片和图像中存在的亮度和色相的显著和细微的变化。

## 2.6彩色图像

- ▶ 眼睛对于彩色的观察和处理是一种生理和心理现象，其机理还没有完全搞清楚，因而对于彩色的许多结论都是建立在实验基础之上的。

## 2.6 .1 色彩的形成与分布

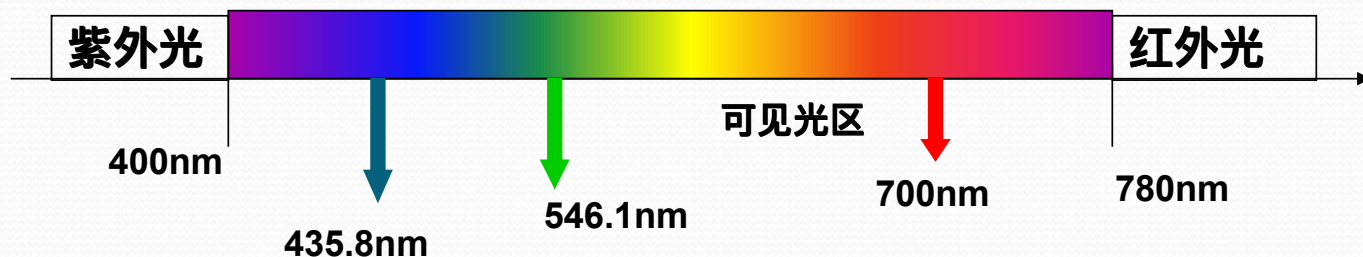


**FIGURE 6.1** Color spectrum seen by passing white light through a prism. (Courtesy of the General Electric Co., Lamp Business Division.)

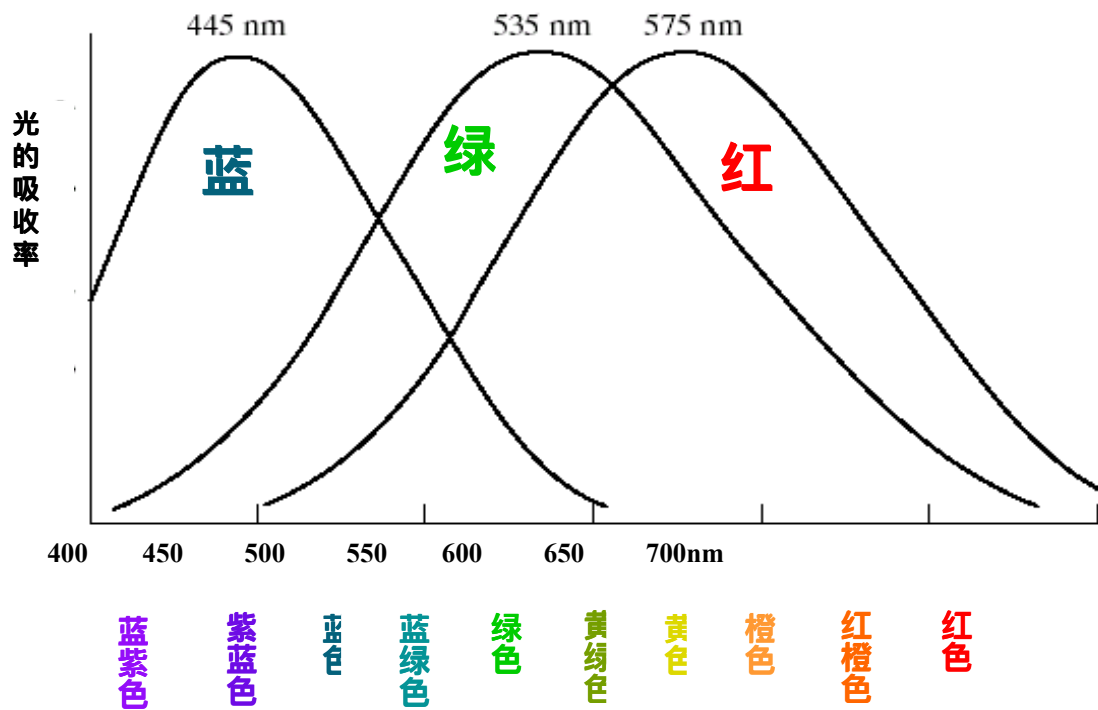
光学原理解释的色彩的形成

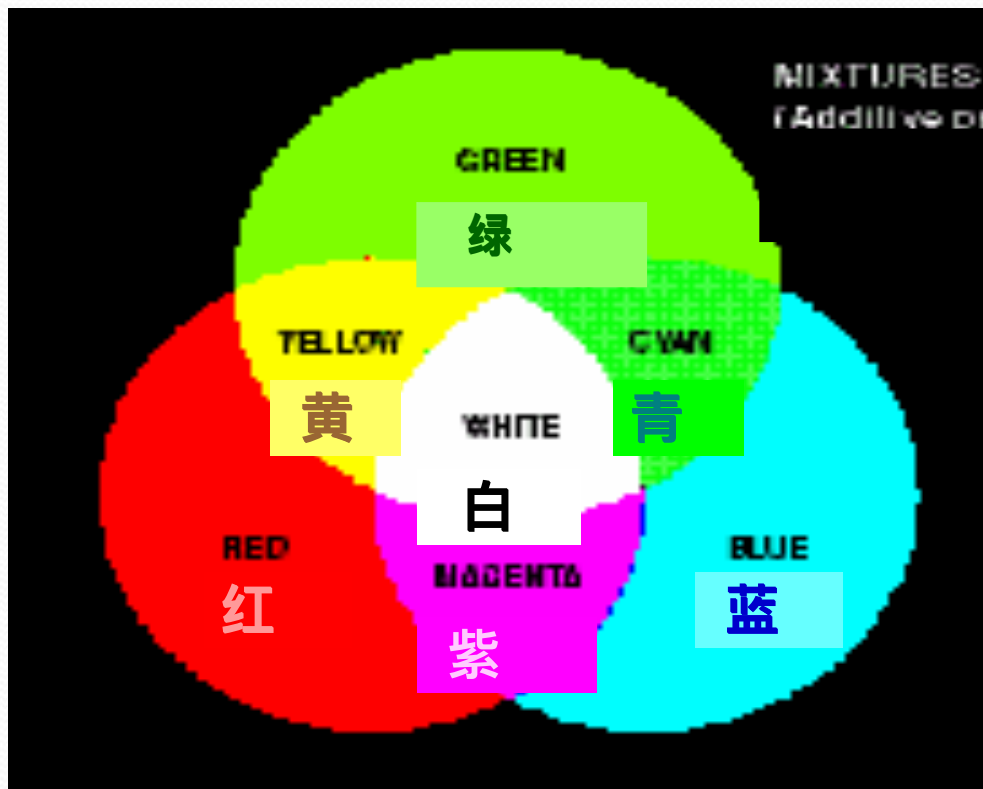


- ▶ 如下图所示,可视光区的波长在400nm ~ 700nm, 当光谱采样限制到三个人类视觉系统敏感的红、绿、蓝光波段时, 对这三个光谱带的光能量进行采样, 就可以得到一幅彩色图像。







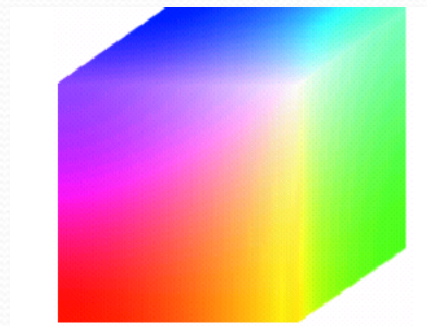
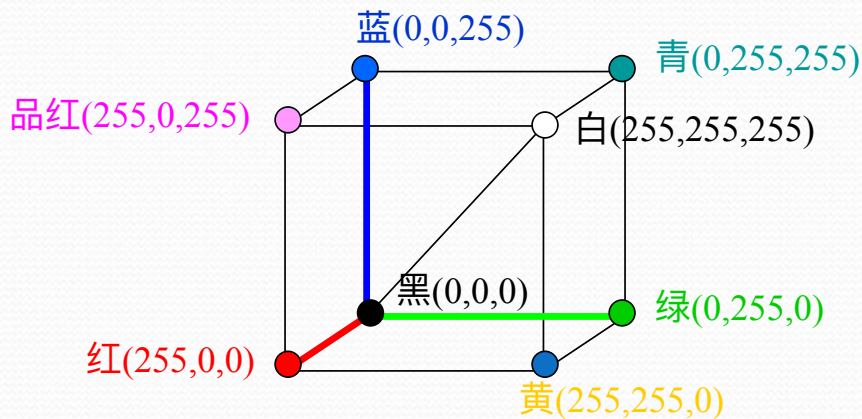


## 2.6.2彩色图像的描述

- ▶ 颜色通过色彩模型描述，不同色彩模型对应不同处理目的。
- ▶ 国际照明委员会CIE, 在大量色彩测试实验基础上, 提出了一系列的颜色模型描述色彩。
- ▶ 各颜色模型之间, 可以通过数学方法互相转换.

## 2.6.2.1 RGB色系

- ▶ CIE规定以700nm(红)、546.1nm (绿)、435.8nm (蓝)三个色光为三基色。又称为物理三基色。
- ▶ 自然界的所有颜色都可用三基色的不同比例混合而成。





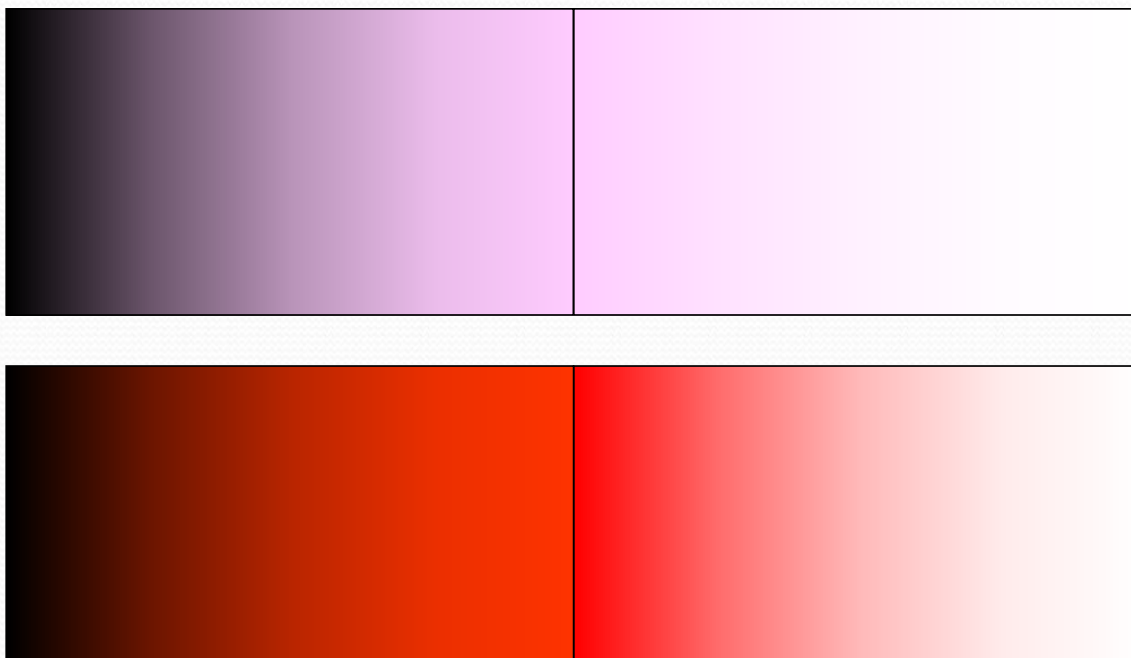
## 2.6.2.2 HSI色系

- ▶ 彩色格式反映了人类观察彩色的方式。
- ▶ 例如，红色又分为浅红和深红色等。
- ▶ I 分量(intensity)
- ▶ I：表示光照强度或称为亮度，确定了像素的整体亮度，不管其颜色是什么。

I: 小 → 大

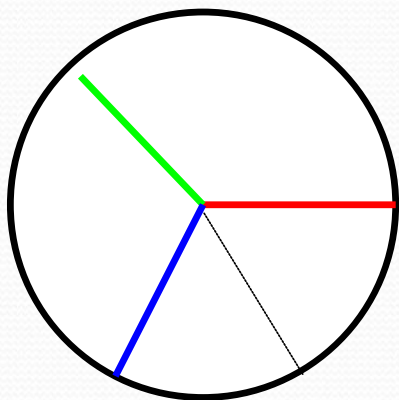


# 亮度(I)效果示意图

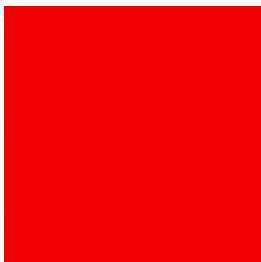


# H分量(hue)

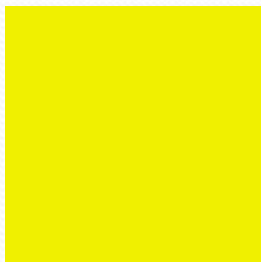
- ▶ H：表示色度，由角度表示。
- ▶ 反映了该颜色最接近什么样的光谱波长。0°为红色，120°为绿色，240°为蓝色。



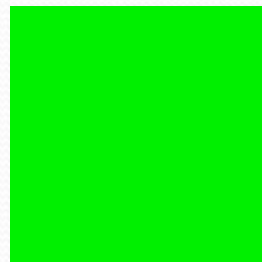
# 色度(H)的效果示意图



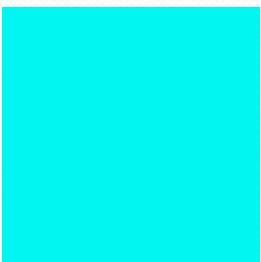
**H=0°**



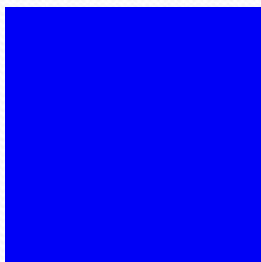
**H=60°**



**H=120°**



**H=180°**



**H=240°**

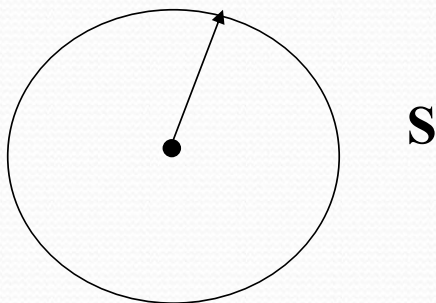


**H=300°**



# S分量(satisfy)

- ▶ S：表示饱和度，饱和度参数是色环的原点到彩色点的半径长度。
- ▶ 在环的外围圆周是纯的或称饱和的颜色，其饱和度值为1。在中心是中性（灰）色，即饱和度为0。



# 饱和度(s)的效果示意图



**S=0**



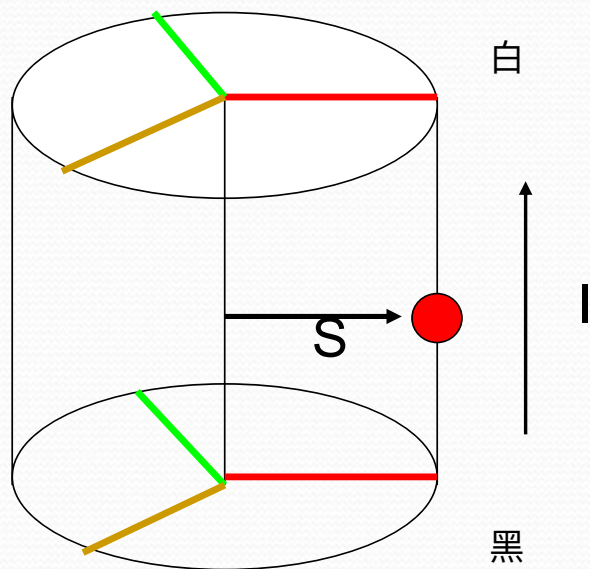
**S=1/4**



**S=1/2**

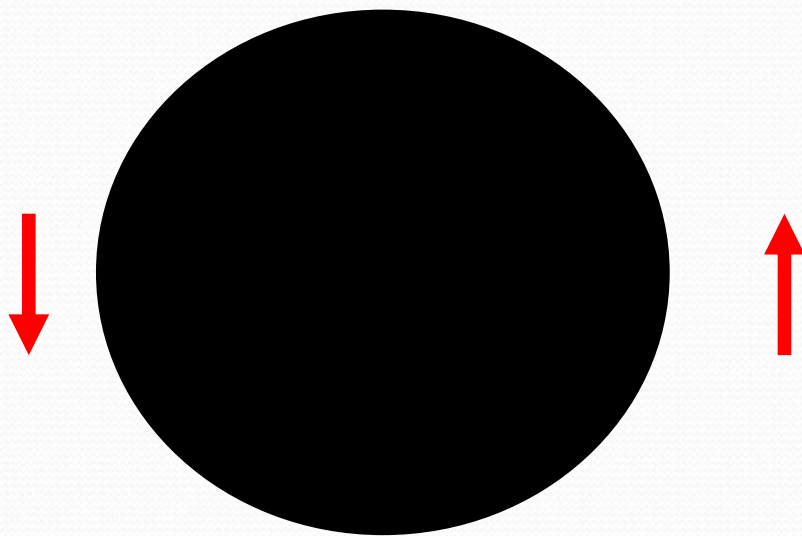


**S=1**



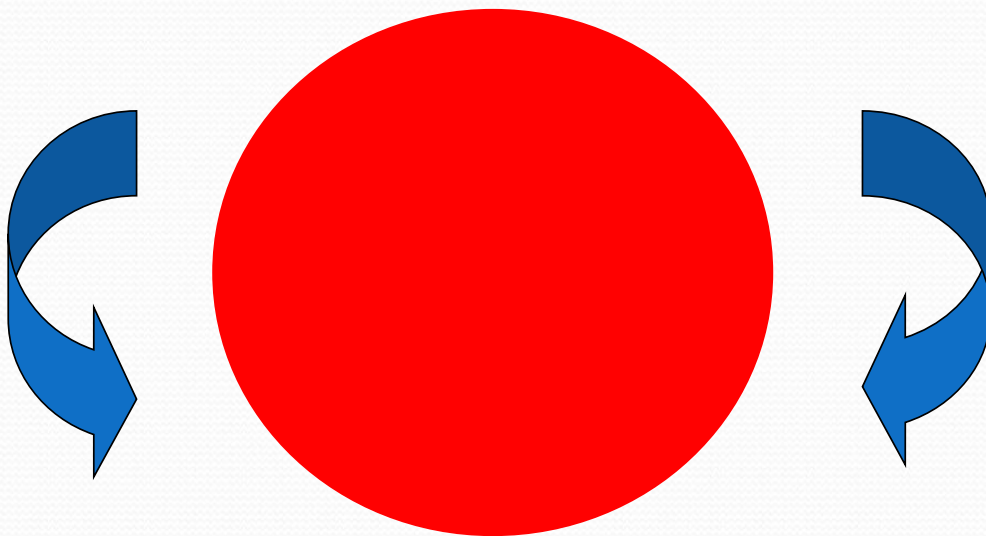
思考问题：在这个圆柱体上，红色的点顺（逆）时针旋转会变成什么样？上下移动呢？向圆心方向移动呢？

# 红点的上下移动

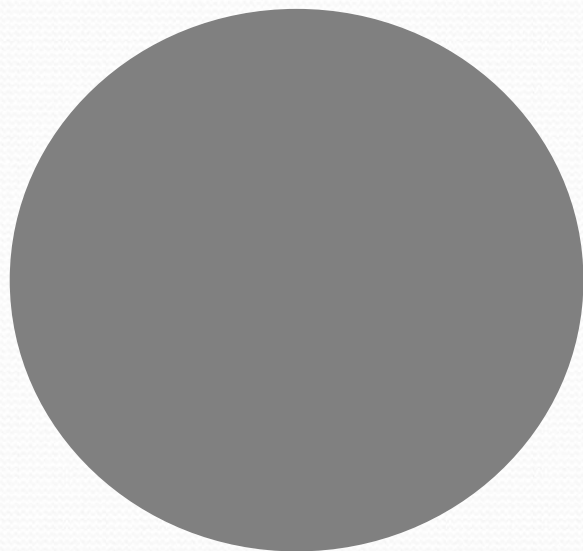




红点的顺（逆）时针转动



红点向圆心方向移动



## 2.6.2.3 HSI与RGB色系的相互转换

### ► RGB到HSI的转换

$$I = \frac{1}{\sqrt{3}} (R + G + B)$$

$$S = 1 - \frac{3 \min(R, G, B)}{R + G + B}$$

$$H = \begin{cases} \theta & G \geq B \\ 2\pi - \theta & G < B \end{cases}$$

$$\theta = \cos^{-1} \left[ \frac{\frac{1}{2}[(R - G) + (R - B)]}{\sqrt{(R - G)^2 + (R - B)(G - B)}} \right]$$

- 当  $0^\circ \leq H \leq 120^\circ$  时

$$R = \frac{I}{\sqrt{3}} \left[ 1 + \frac{S \cos(H)}{\cos(60^\circ - H)} \right]$$

$$B = \frac{I}{\sqrt{3}} (1 - S)$$

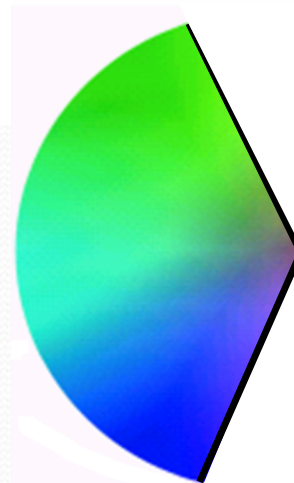
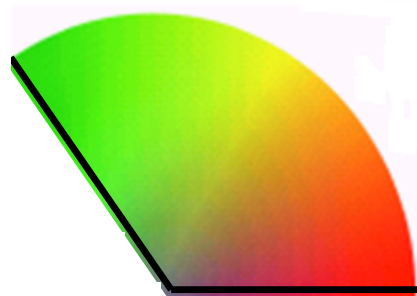
$$G = \sqrt{3}I - R - B$$

- 当  $120^\circ \leq H \leq 240^\circ$  时

$$R = \frac{I}{\sqrt{3}} (1 - S)$$

$$G = \frac{I}{\sqrt{3}} \left[ 1 + \frac{S \cos(H - 120^\circ)}{\cos(180^\circ - H)} \right]$$

$$B = \sqrt{3}I - R - G$$





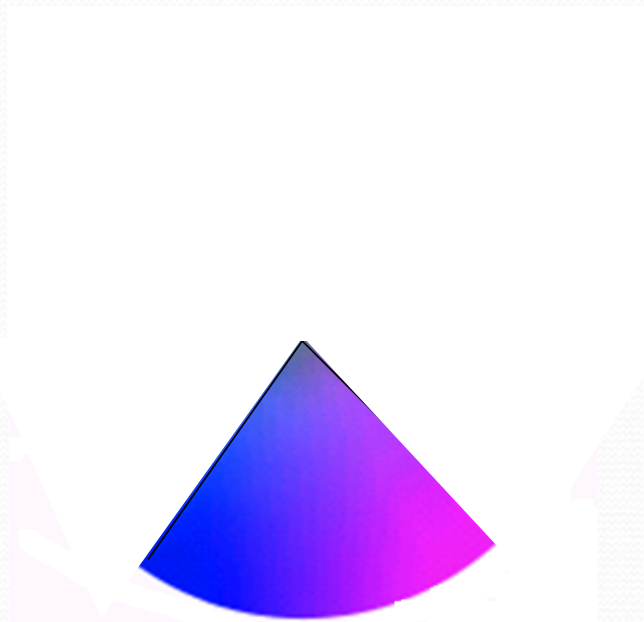
➤ 当  $240^\circ \leq H < 300^\circ$  时

$$B = \frac{I}{\sqrt{3}} \left[ 1 + \frac{S \cos(H - 240^\circ)}{\cos(300^\circ - H)} \right]$$

$$G = \frac{I}{\sqrt{3}} (1 - S)$$

$$R = \sqrt{3}I - G - B$$

注意：300~360之间为非可见光谱色，没有定义



## 2.6.2.4 YUV电视信号表色系

▶ 在这种表色系统中

Y: 亮度; U, V: 色差信号

U为蓝色与亮度的差; V为红色与亮度的差

▶ 目的是为了可以与黑白电视兼容

▶ 电视信号在发射时, 转换成YUV形式, 接收时再还原成RGB三基色信号, 由显像管显示。



Y,U,V

Y



Y

Y,0,0



## 2.6.2.5 YUV与RGB色系的转换

### ► RGB到YUV的转换

$$Y = 0.299R + 0.587G + 0.114B$$

$$U = B - Y$$

$$V = R - Y$$

特点：

这两个色系的转换非常简单，  
所以可满足转换的快速性要求。

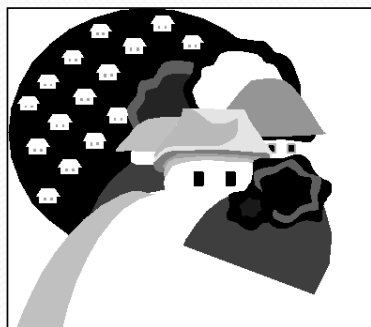
### ► YUV到RGB的转换

$$R = Y + V$$

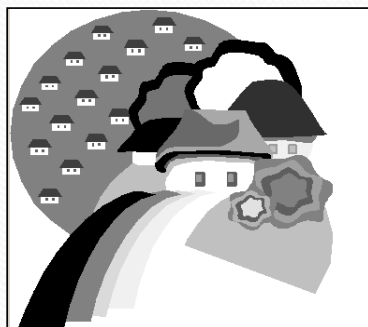
$$G = Y - 0.192U - 0.509V$$

$$B = Y + U$$

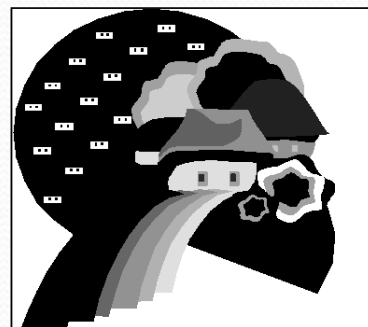




R



G



B

## 2.6.2.6 YCbCr表色系

- ▶ 常用于彩色图像压缩时的一种表色系。

**Y:** 代表亮度;


**Cb、Cr:** 代表色差, 即色彩信号。

- ▶ 亮度信号Y与R、G、B信号之间的关系是色彩信号与R、G、B信号关系的表达式

$$R = Y + Cr$$

$$G = Y - 0.3Cr / 0.59 - 0.11Cb / 0.59$$

$$B = Y + Cb$$

- 
- ▶ 与YUV表色系统不同, 充分考虑RGB三色的重要因素。
  - ▶ YUV考虑色系转换的简单;
  - ▶ YCbCr考虑压缩时可以充分取出冗余量。

## 2.6.2.7 YCbCr与RGB表色系的转换

### ► RGB到YCbCr的转换

$$Y = 0.3R + 0.59G + 0.11B$$

$$Cr = R - Y = 0.7R - 0.59G - 0.11B$$

$$Cb = B - Y = -0.3R - 0.59G + 0.89B$$

Cr、Cb是分别从R、B中把亮度信号减去后的形式，故称为色差信号。



## ► YCbCr到RGB的转换

$$R = Y + Cr$$

$$G = Y - 0.3Cr/0.59 - 0.11Cb/0.59$$

$$B = Y + Cb$$

- 色彩三要素 (Elements of color) 是指色度 (色调) (hue)、饱和度 (纯度) (Saturation) 和亮度 (明度) (Brightness)。人眼看到的任一彩色光都是这三个特性的综合效果。

## 2.6.3 色彩空间

- ▶ “色彩空间”源于西方的“Color Space”，又称作“色域”。
- ▶ 色彩学的多种色彩模型，以一维、二维、三维甚至四维空间坐标来表示某一色彩。
- ▶ 坐标系统定义色彩范围，即色彩空间，也称色彩（色调）模型。
- ▶ 色彩空间模型有：RGB色彩模型，CMYK色彩模型，LAB色彩模型，HSI色彩模型，YIQ、YUV色彩模型，CMY色彩模型。

# RGB模型

- ▶ RGB色彩空间能产生多达1670万种颜色。
- ▶ 1670万种颜色，但与可见光谱的色域范围相比要窄得多
- ▶ RGB色彩空间是与设备有关的，不同RGB设备再现的颜色不可能完全相同。

# 下表是常见的七种颜色RGB组合值

颜色名	R 值	G值	B值
红	255	0	0
绿	0	255	0
蓝	0	0	255
白	255	255	255
黑	0	0	0
青	0	255	255
紫	255	0	255
黄	255	255	0



## 2.6.3.1 CMY模型

- ▶ CMY颜色模型是以红、绿、蓝三色的补色青(Cyan)、品红(Magenta)、黄(Yellow)为原色构成的颜色模型。
- ▶ CMY颜色模型从白光中滤去某种颜色，称为减色原色空间。

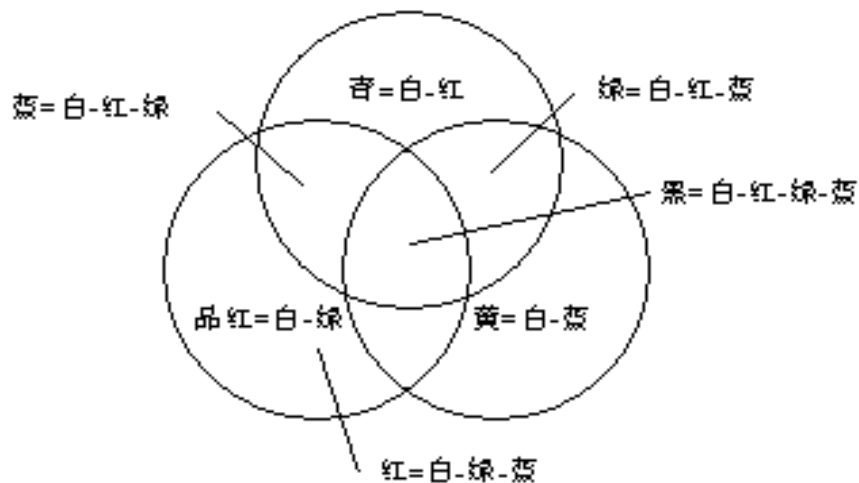


图3 CMY在原色的减色效果示意图

- CMY颜色模型对应直角坐标系子空间，与RGB颜色模型对应的子空间几乎完全相同。但RGB和CMY颜色模型仍具有一定的区别。

	RGB 颜色模型	CMY 颜色模型
三原色	R、G、B	C、M、Y
成色基本规律	$R+G=Y$ $R+B=M$ $G+B=C$ $R+G+B=W$	$Y+M=R$ $C+Y=G$ $C+M=B$ $C+M+Y=K$
实质	色光相加，光能量增大	色料混合，光能量减小
效果	明度增大	明度减小
成色方式	视觉器官外      空间混合 视觉器官内      静态混合 动态混合	色料掺合 透明色层迭合
补色关系	补色光相加，愈加愈亮，形成白色	补色料相加，愈加愈暗，形成黑色
主要应用	彩色电影、电视、测色计	彩色绘画，摄影、印刷、印染

## 2.6.3.3 不同色彩空间之间的转换

### ► RGB与CMY

青、品红、黄(CMY)彩色模型是彩色图象印刷行业使用的彩色空间。

彩色立方体中是红、绿、蓝的补色，称为减色基。

而红、绿、蓝称为加色基。

在CMY模型中，颜色是从白光中减去一定成分得到的。

$$C = 1 - R$$

$$M = 1 - G$$

$$Y = 1 - B$$

## ► RGB与YUV

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.435 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$



## ► RGB 与 YCbCr

$$R = Y + 1.402 (Cr - 128)$$

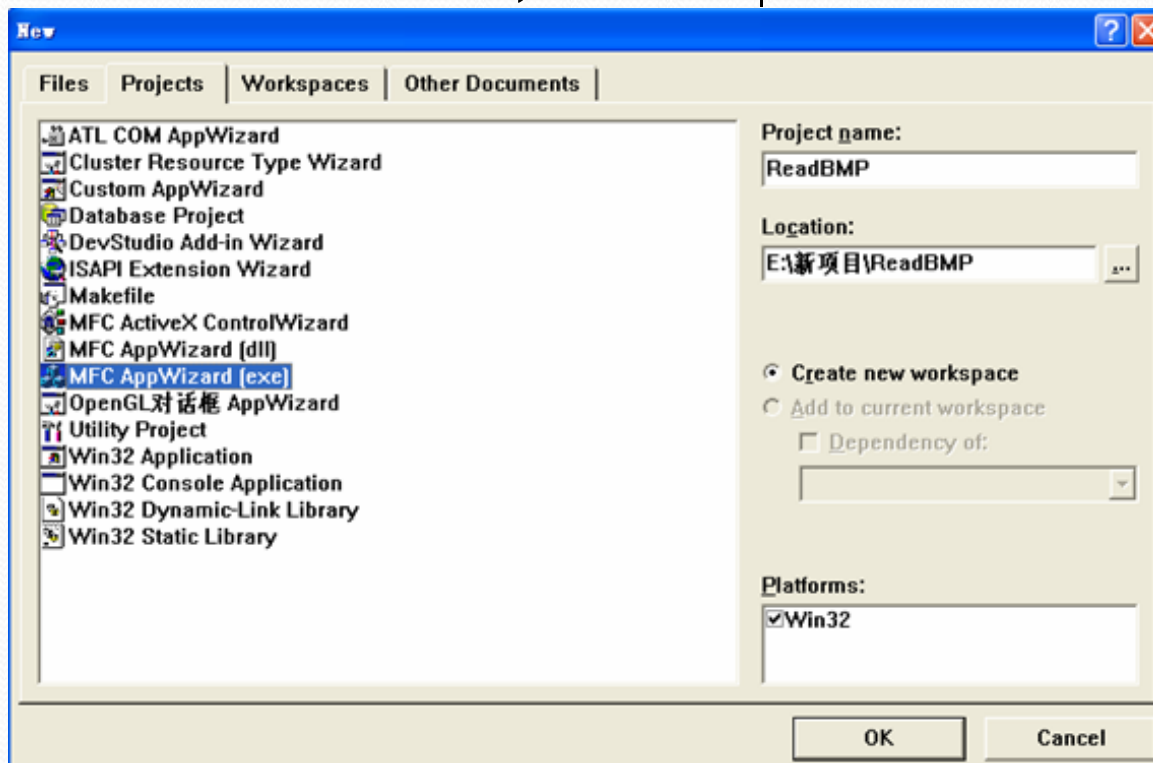
$$G = Y - 0.34414 (Cb - 128) - 0.71414 (Cr - 128)$$

$$B = Y + 1.772 (Cb - 128)$$

## 2.7用VC++实现BMP文件的显示

- ▶ 本节介绍如何在VC++ 6.0中实现BMP图像的显示。
- ▶ 下面介绍具体步骤：

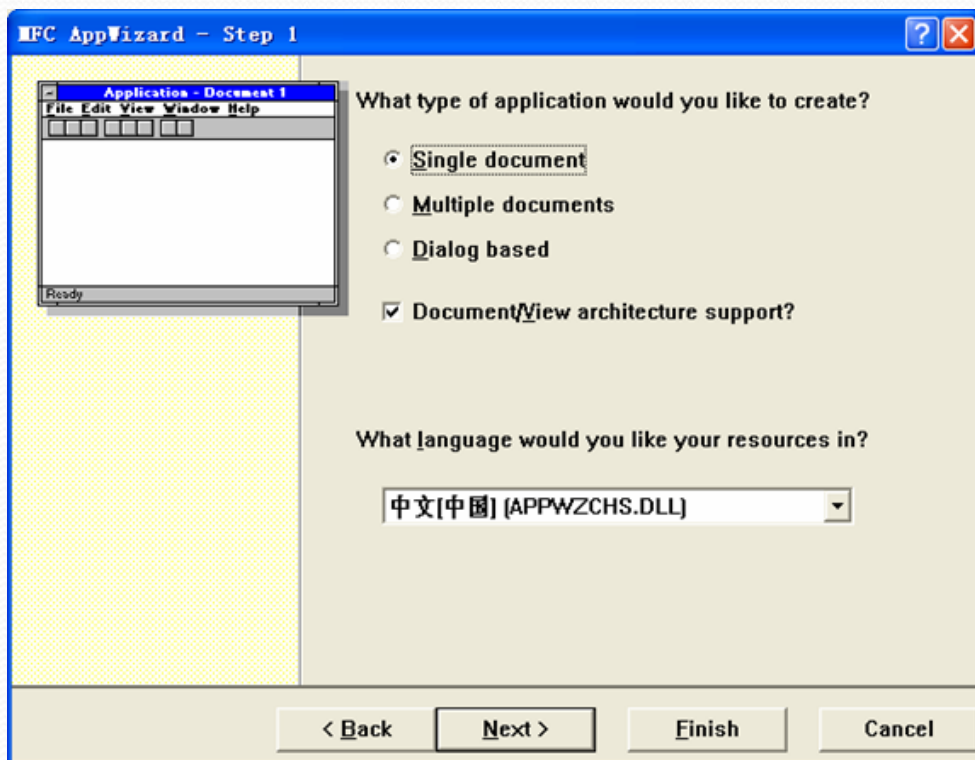
➤ 步骤一、打开VC++ 6.0，选择File|New进入界面



在Projects中选择MFC AppWinzard(exe)，在Project name中输入项目名称，ReadBMP，在Location中输入项目要保存的文件夹。点击“OK”进入下一步。

## ➤ 步骤二、选择文档类型

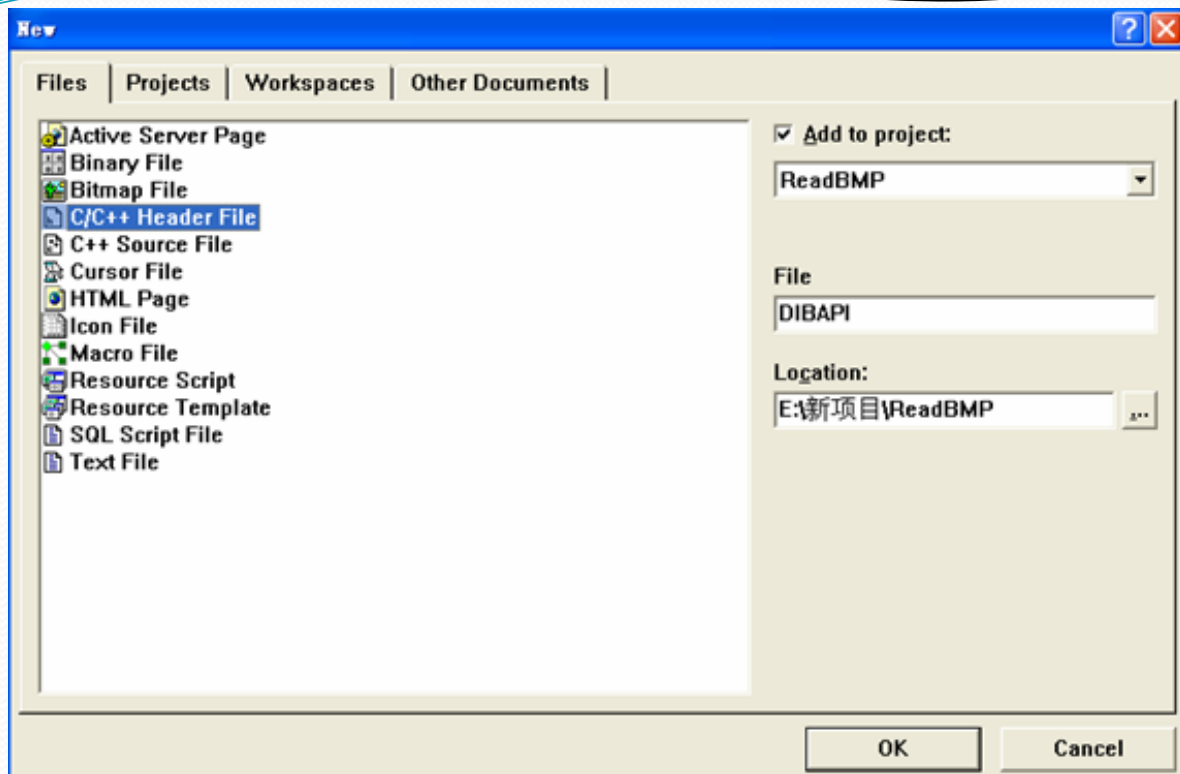
单文档视图结构，选择Single document。其余部分设置使用VC++ 6.0的默认设置，点击“Finish”完成项目创建。





- ▶ 步骤三、为了将BMP中的数据读入到内存中，在项目中建立专门处理BMP文件头和数据文件：DIBAPI.H和DIBAPI.CPP，在其中实现对BMP文件的大部分处理。

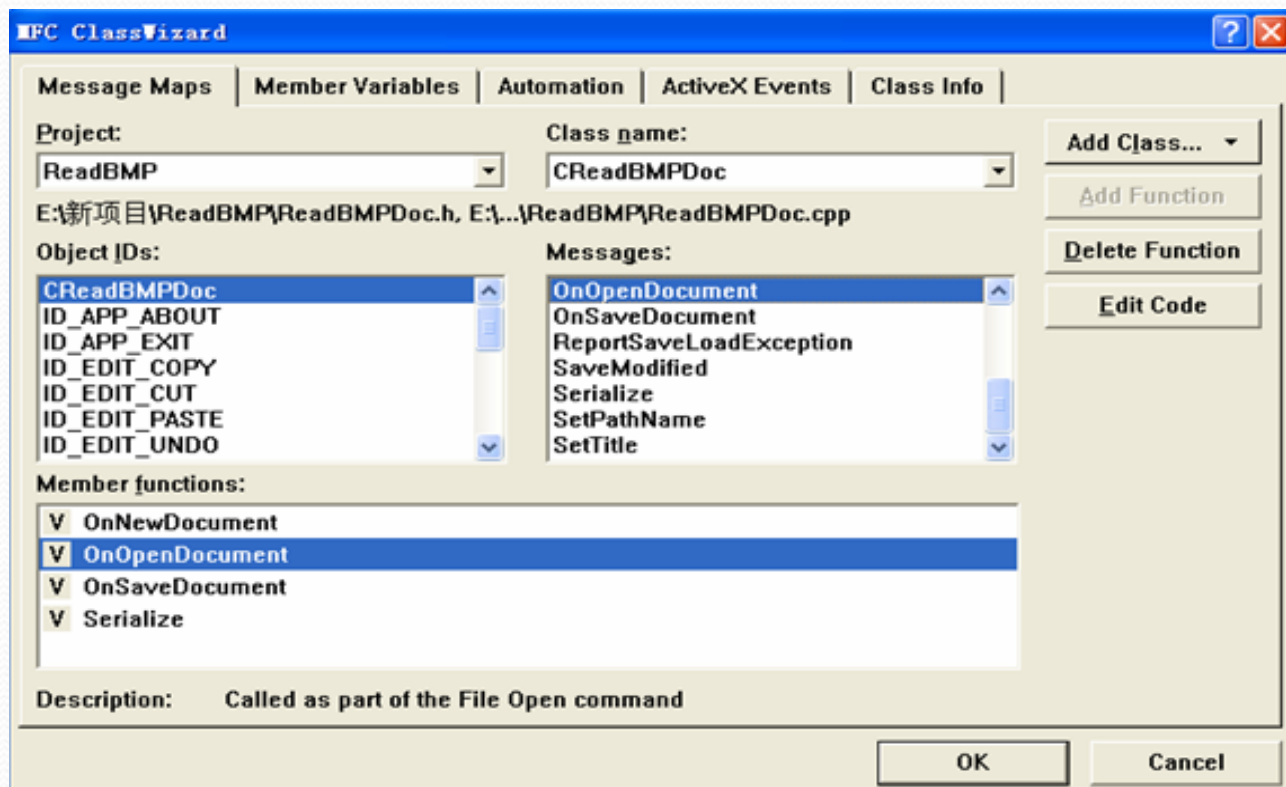
选择File|New从弹出界面Files选项中选择C/C++ Header File，建立一个新的头文件。在右边的File输入框中输入文件名，命名为DIBAPI，默认后缀为.H。



同上类似，选择C++ Source File建立DIBAPI.CPP文件。

- ▶ 步骤四、在CReadBMPDoc类中添加变量CPalette\* m\_palDIB 和HDIB m\_hDIB。
- ▶ m\_hDIB用于保存当前BMP图像句柄；
- ▶ m\_palDIB用于指向BMP图像对应的调色板；
- ▶ 在CReadBMPDoc的构造函数中初始化：
- ▶ m\_hDIB = NULL;m\_palDIB = NULL。
- ▶ 步骤五、为了取得保存在当前文档中的HDIB和Palette数据，在CReadBMPDoc类中添加方法：GetHDIB和GetDocPalette，

- ▶ 步骤六、响应类CReadBMPDoc OnOpenDocument事件，以实现打开文件的操作。从“View|Class Wizard”进入“MFC Class Wizard”界面，在“Message Maps”选项中完成消息映射。



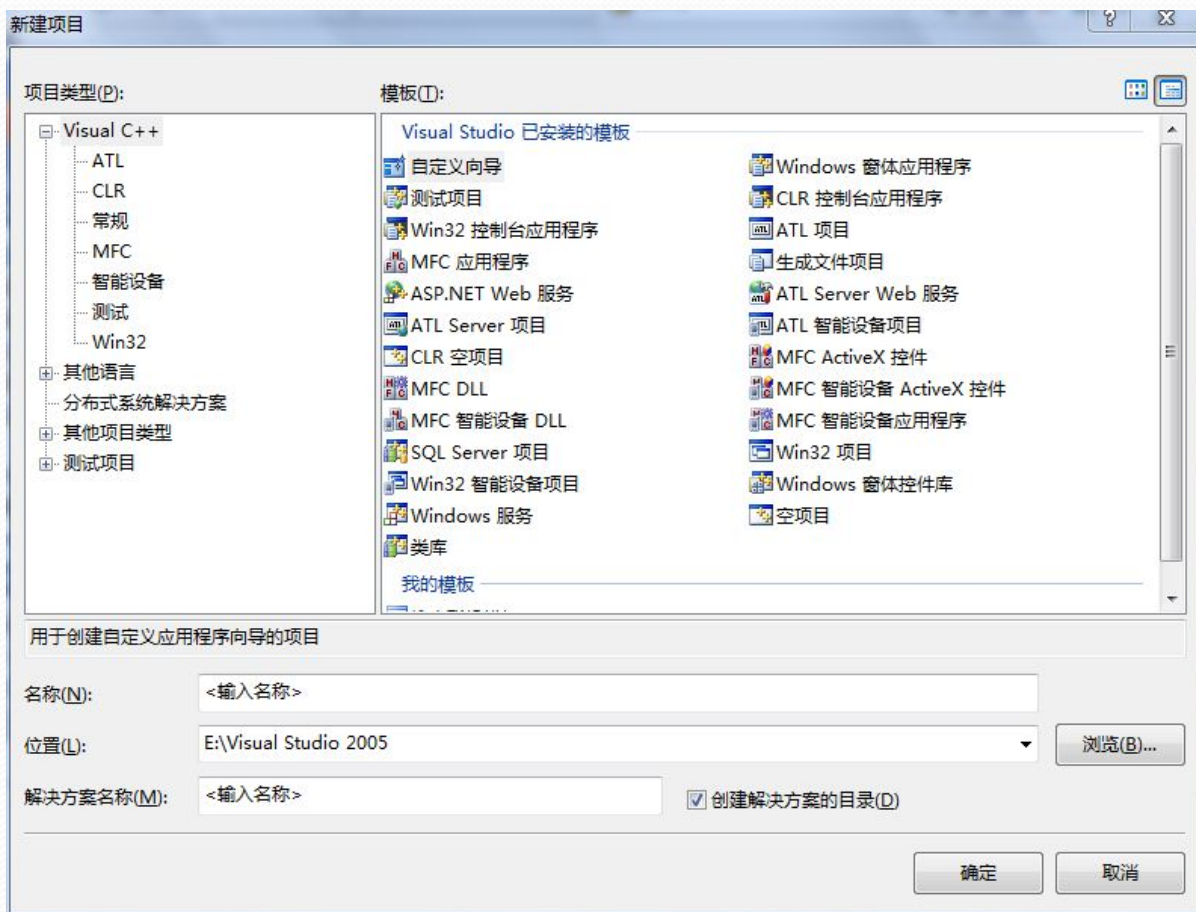


- ▶ 步骤七、完成图片打开操作后，图片数据已经被保存在程序中，为将图片显示出来，要响应类CReadBMPView的OnDraw事件，在其中完成图像显示。
- ▶ 步骤八、编译并运行程序，自此一个用于打开BMP图像的单文档视图结构的程序。通过修改当前位图句柄m\_hDIB中存放像素的数据就可以对图像进行改变。

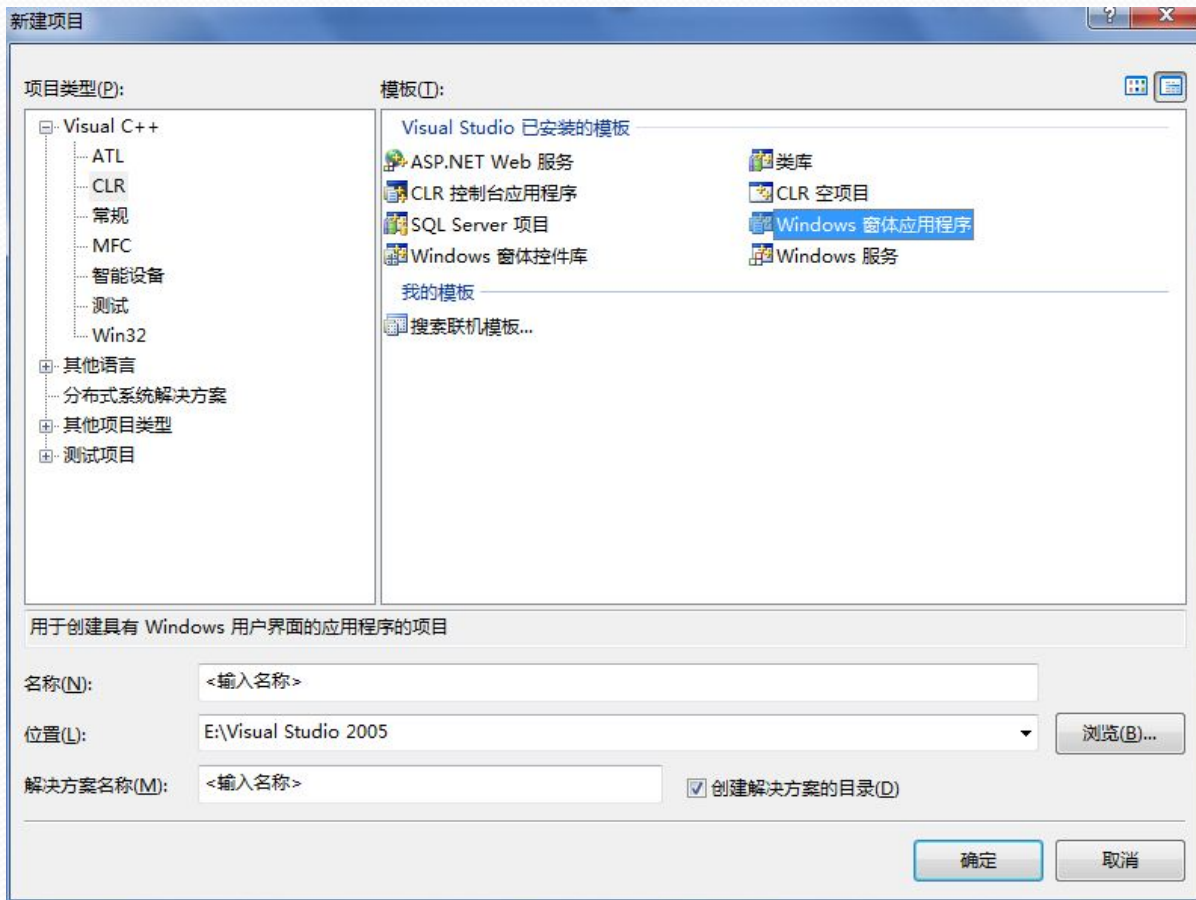
## 2.8用.NET实现图像文件的打开与显示

- ▶ Visual C++ 2005 创建一个Windows Form 窗体应用程序，并实现打开和显示图像文件。
- ▶ 具体步骤如下：

- 选择菜单栏上的“文件”→“新建”→“项目”命令，打开“新建项目”对话框，如图所示。

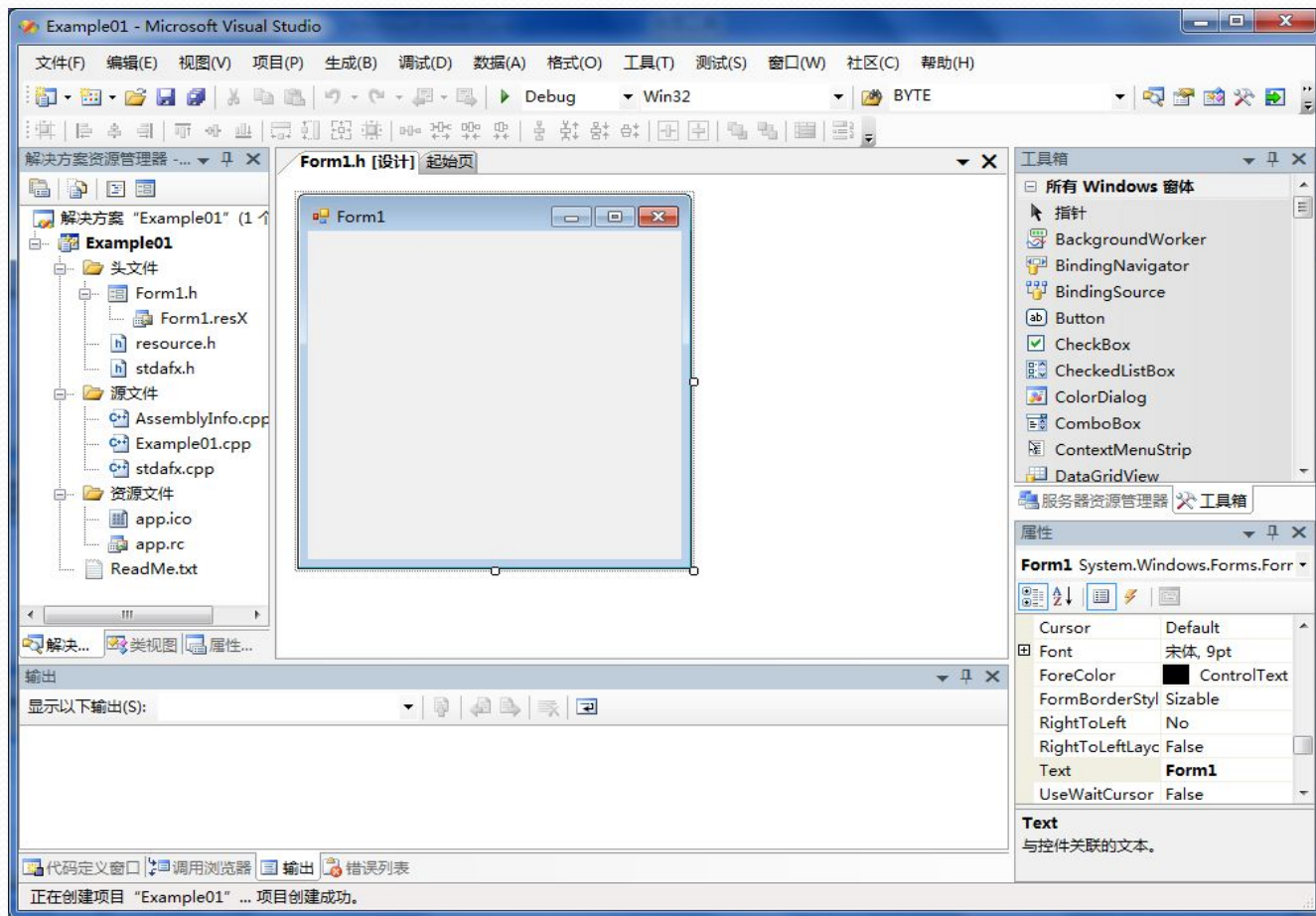


- 在左侧面板中选择“Visual C++”中的CLR子项，在右侧面板中选择“Windows窗体应用程序”。

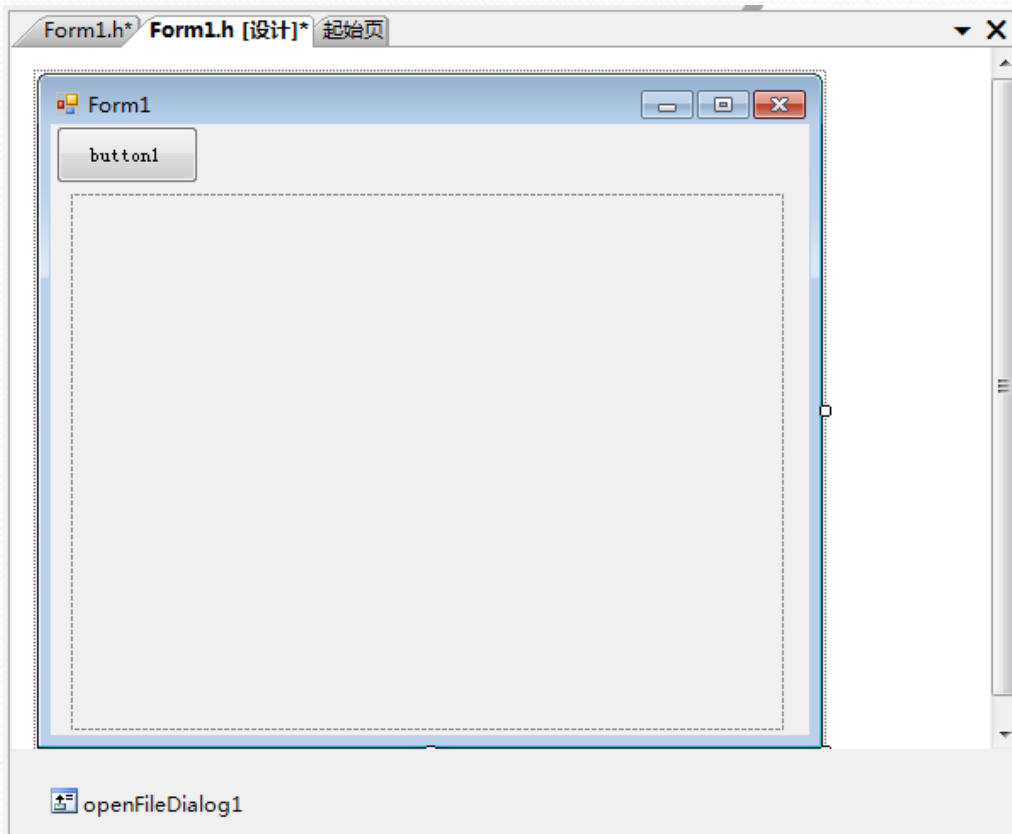




- 输入“名称”，“位置”和“解决方案名称”，单击“确定”按钮，便创建了一个新项目。



- ▶ 将控件Button、PictureBox、OpenFileDialog分别拖入窗体中，并调整到合适的位置。



- ▶ “左键 | 查看代码”命令可以查看生成代码。

```
//定义句柄 button1  
private: System::Windows::Forms::Button^ button1;  
//定义句柄 pictureBox1  
private: System::Windows::Forms::PictureBox^ pictureBox1;  
//定义句柄 openFileDialog1  
private: System::Windows::Forms::OpenFileDialog^ openFileDialog1;
```

// button1的属性

```
this->button1->Location = System::Drawing::Point(3, 1);  
this->button1->Name = L"button1";  
this->button1->Size = System::Drawing::Size(84, 34);  
this->button1->TabIndex = 0;  
this->button1->Text = L"button1";  
this->button1->UseVisualStyleBackColor = true;
```

// pictureBox1的属性

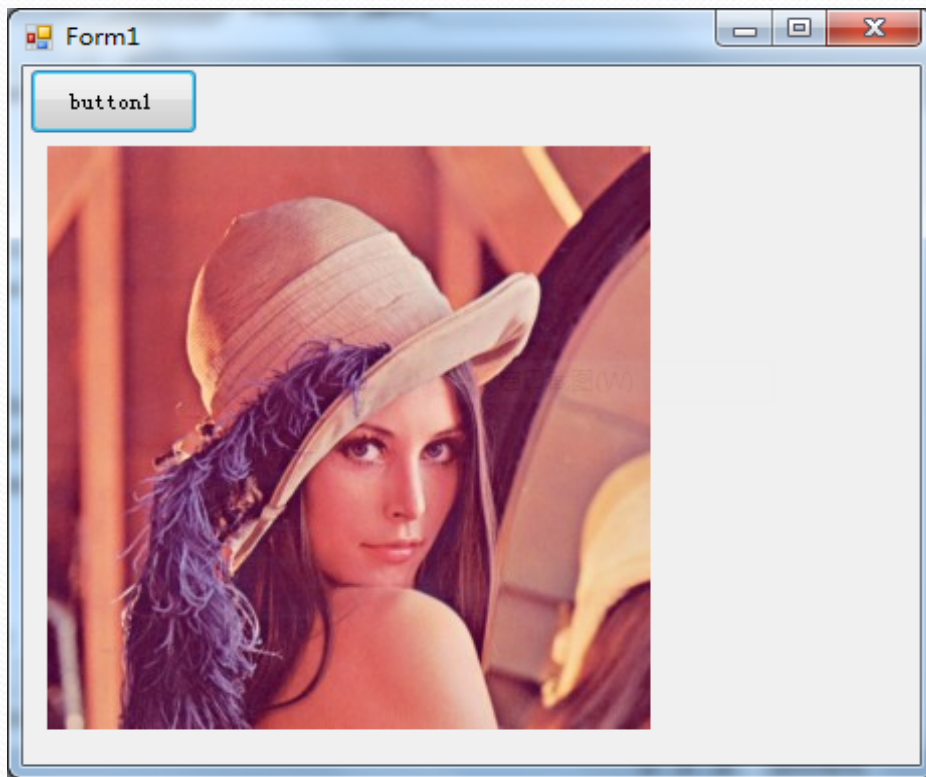
```
this->pictureBox1->Location = System::Drawing::Point(12, 41);  
this->pictureBox1->Name = L"pictureBox1";  
this->pictureBox1->Size = System::Drawing::Size(419, 315);  
this->pictureBox1->SizeMode=System::Windows::Forms::  
    PictureBoxSizeMode::StretchImage;  
this->pictureBox1->TabIndex = 1;  
this->pictureBox1->TabStop = false;
```



- ▶ 双击窗体设计器上的“button1”按钮，便可对编写代码响应鼠标事件。

```
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {  
    OpenFileDialog^ openFile1 = gcnew OpenFileDialog();  
    openFile1->InitialDirectory = "E:\\";  
    openFile1->Filter = "Bitmap Image(*.bmp)|*.bmp";  
    openFile1->RestoreDirectory = true;  
    if (openFile1->ShowDialog() == Windows::Forms::DialogResult::OK)  
    {  
        String^ fileName = openFile1->FileName->ToString();  
        Bitmap^ bmap = gcnew Bitmap(fileName);  
        pictureBox1->Image = bmap;  
        pictureBox1->Refresh();  
    }  
}
```

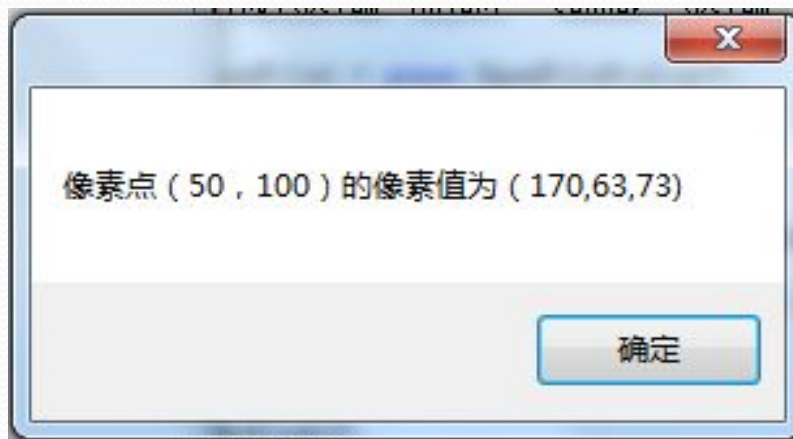
- ▶ 连接、编译并运行程序，最后结果如图所示。



- ▶ 读取图像中的一个像素值。在函数button1\_Click() 的if语句中再加上如下语句。

```
String^ redValue = bmap->GetPixel(50,100).R.ToString();  
String^ greenValue = bmap->GetPixel(50,100).G.ToString();  
String^ blueValue = bmap->GetPixel(50,100).B.ToString();  
MessageBox::Show("像素点 (50, 100) 的像素值为  
("+redValue+", "+greenValue+", "+blueValue+)");
```

- ▶ 重新编译和运行，弹出对话框如图所示。



## 2.6小结

- ▶ 一幅图像实际上记录的是物体辐射能量的空间分布，这个分布是空间坐标、时间和波长的函数。当一幅图为平面单色静止图像时，图像可以用二维函数 $f(x,y)$ 来表示，它是一个有界函数。
- ▶ 将一幅图像数字化的过程主要包括扫描，采样和量化其结果就是在计算机内生成一个二维矩阵。
- ▶ 计算机一般采用两种方式存储静态图像：一种是位映射，即位图存储模式；另一种是向量处理，也称矢量存储模式。



- 
- ▶ 位图可以分成二值图像、索引图像、灰度图像和RGB图 像四种。
  - ▶ 常用的图像文件存储格式主要有BMP图像文件、JPEG图像文件、PCX图像文件、TIFF图像文件以及GIF图像文件等。

再见