原 最简单的基于FFmpeg的libswscale的示例附件：测试图片生成工具

2014年12月29日 12:07:56 阅读数：9234

==========

最简单的基于FFmpeg的libswscale的示例系列文章列表：

最简单的基于FFmpeg的libswscale的示例（YUV转RGB）

最简单的基于FFmpeg的libswscale的示例附件：测试图片生成工具

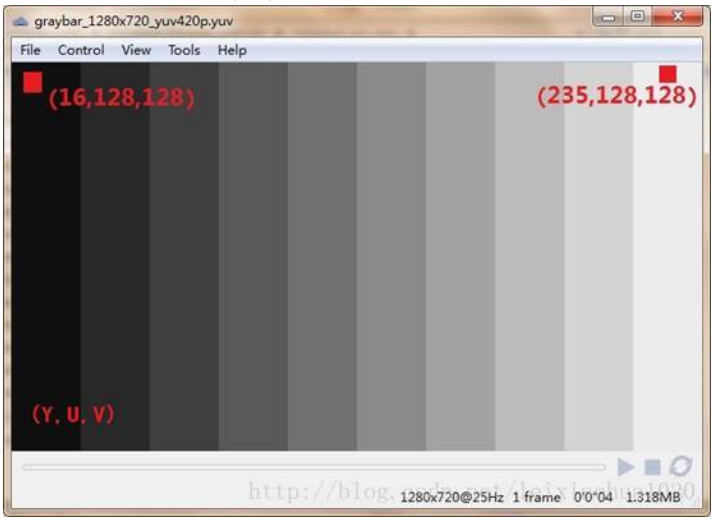
==========

本文记录一个自己写的简单的测试图片生成工具：simplest_pic_gen。该工具可以生成视频测试时候常用的RGB/YUV格式的测试图片。包括灰阶测试图，彩条图，彩色条纹图，RGB渐变彩条图，YUV渐变彩条图，颜色视频等。下面简单介绍一下这些测试图片的生成函数。

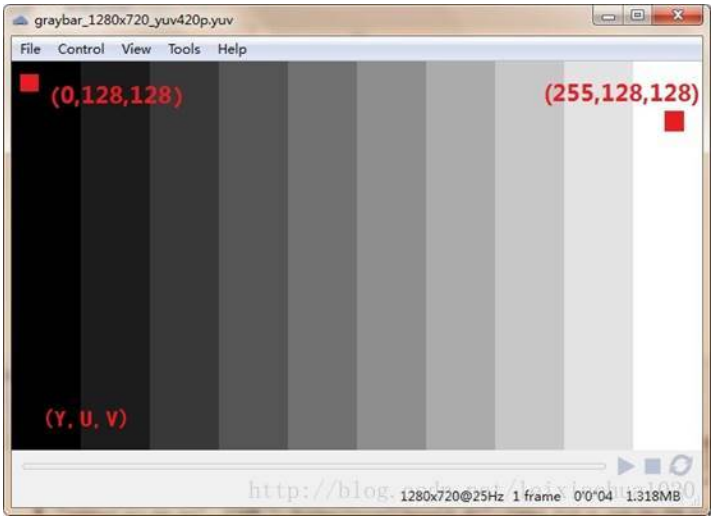
这里有一点需要注意：查看生成的图片需要使用RGB/YUV播放器。

灰阶测试图

亮度取值为16-235的灰阶测试图
下面这张图是一张灰阶测试图的示例。这张图的分辨率是1280x720，像素格式是YUV420P，亮度的取值范围是16-235，一共包含了10级的灰度。最左边的灰度竖条的YUV取值为（16,128,128），最右边的灰度竖条的YUV取值为（235,128,128）。



亮度取值为0-255的灰阶测试图
下面这张图的分辨率是1280x720，像素格式是YUV420P，亮度的取值范围是0-255，一共包含了10级的灰度。最左边的灰度竖条的YUV取值为（0,128,128），最右边的灰度竖条的YUV取值为（255,128,128）。



在生成灰度图的同时，程序会打印出每一个灰阶的YUV取值。

函数原型

gen_yuv420p_graybar()是用于生成灰阶测试图的函数，该函数的定义如下。

```cpp
/**
 * Generate Picture contains Gray Bar changing from Black to White in YUV420P Format
 *
 * @param width             the width of picture.
 * @param height  the height of picture.
 * @param barnum         the number of Bars in the picture.
 * @param ymin              the minimum value of luminance.
 * @param ymax             the maximum value of luminance.
 * @return 0 if finished, -1 if there are errors.
 */
int gen_yuv420p_graybar(int width, int height,int barnum,unsigned char ymin,unsigned char ymax);
```

简单解释每个参数的含义：

width：图像宽

height：图像高

barnum：灰阶数量

ymin：亮度最小取值

ymax：亮度最大取值

如果函数成功运行的话，会生成一个名称为“graybar_%dx%d_yuv420p.yuv”的YUV420P格式的文件（其中%dx%d代表了图像的宽和高）。

例如，生成分辨率为1280x720的上文中的灰阶图的代码如下。

亮度取值范围为16-235：

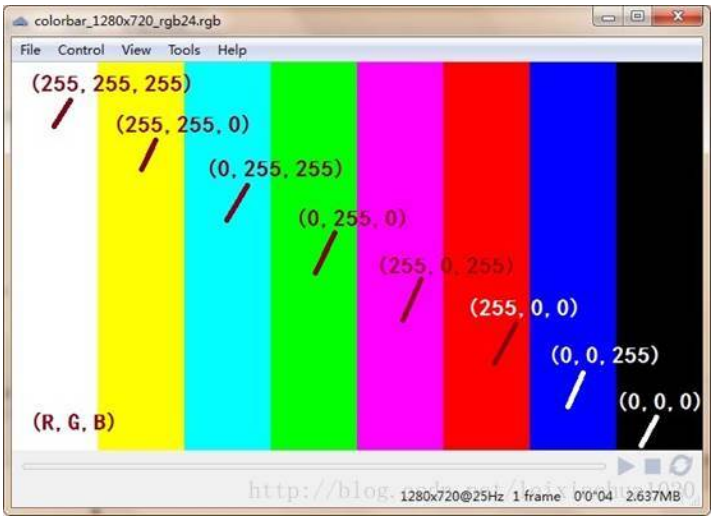
```cpp
gen_yuv420p_graybar(1280,720,10,16,235);
```

亮度取值范围为0-255

```cpp
gen_yuv420p_graybar(1280,720,10,0,255);
```

彩条测试图

在电视节目的制作播出及设备维护中，最常用的莫过于彩条信号了。这是由于彩条信号能正确反映出各种彩色的亮度、色调和饱和度，是检验视频通道传输质量最方便的手段。下面这张图是一张彩条测试图的示例。这张图的分辨率是1280x720，像素格式是RGB24，包含了电视系统中常见的“白黄青绿品红蓝黑”彩条。



“白黄青绿品红蓝黑”彩条中每种颜色的RGB取值如下所示：

颜色	(R, G, B)取值
白	(255, 255, 255)
黄	(255, 255, 0)
青	(0, 255, 255)
绿	(0, 255, 0)
品	(255, 0, 255)
红	(255, 0, 0)
蓝	(0, 0, 255)
黑	(0, 0, 0)

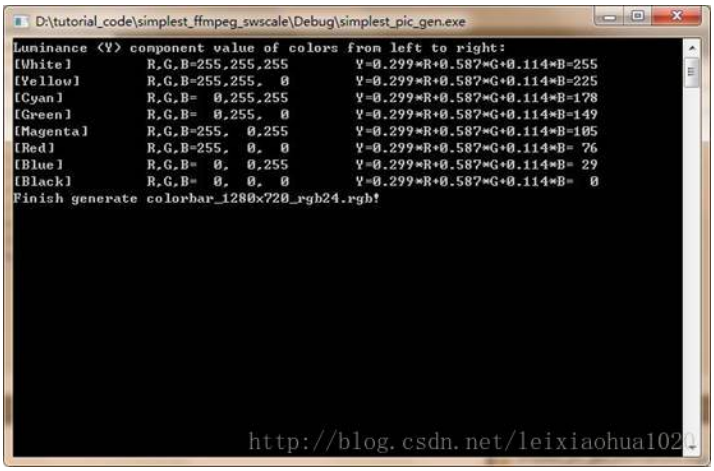
很多人会奇怪，这8个彩条信号的顺序为什么是“白黄青绿品红蓝黑”？其实，它们是按照它们的亮度进行排序的。
RGB转换为YUV的过程中，可以通过RGB计算该颜色的亮度。计算的公式如下所示。

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

把上述8个颜色的R,G,B取值带入上述公式，可以得到每种颜色的亮度取值，如下所示：

颜色	亮度取值
白	255
黄	225
青	178
绿	149
品	105
红	76
蓝	29
黑	0

在生成彩条图像之后，程序会打印出彩条信号的颜色信息，如下图所示。



函数原型

gen_rgb24_colorbar()是用于生成彩条测试图的函数，该函数的原型如下。

```cpp
/**
 * Generate Picture contains standard Color Bar in RGB24 Format
 *
 * @param width             the width of picture.
 * @param height  the height of picture.
 * @return 0 if finished, -1 if there are errors.
 */
int gen_rgb24_colorbar(int width, int height);
```

简单解释每个参数的含义：

width：图像宽

height：图像高

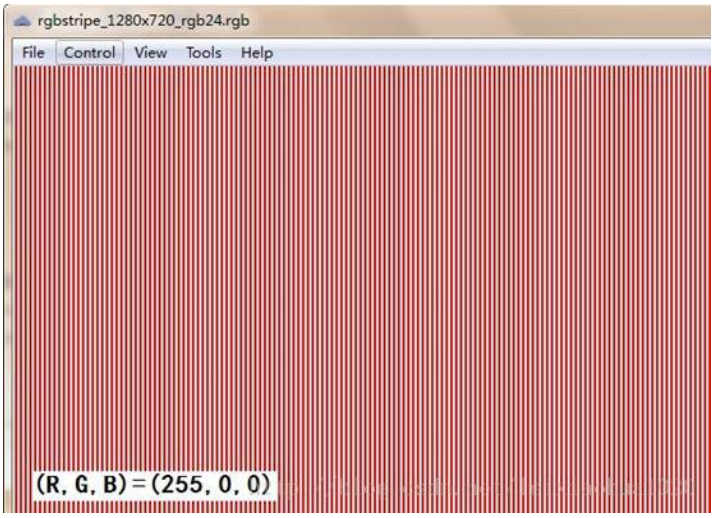
如果函数成功运行的话，会生成一个名称为“colorbar_%dx%d_rgb24.rgb”的RGB24格式的文件（其中%dx%d代表了图像的宽和高）。

例如，生成分辨率为1280x720的上文中的彩条图的代码如下。

```cpp
gen_rgb24_colorbar(1280,720);
```

彩色条纹图

条纹图也是常见的一种测试图。下面这张图是一张彩色条纹图的示例。这张图的分辨率是1280x720，像素格式是RGB24，条纹的颜色为红色。其中竖直条纹的宽度为1像素，条纹之间的间隔也是1像素。



函数原型

gen_rgb24_stripe ()是用于生成会接测试图的函数，该函数的原型如下。

```cpp
/**
 * Generate Picture contains Stripe in RGB24 Format
 *
 * @param width             the width of picture.
 * @param height  the height of picture.
 * @param r                      Red component of stripe
 * @param g                     Green component of stripe
 * @param b                     Blue component of stripe
 * @return 0 if finished, -1 if there are errors.
 */
int gen_rgb24_stripe(int width, int height,
          unsigned char r,unsigned char g,unsigned char b)
```

简单解释每个参数的含义：

width：图像宽

height：图像高

r：条纹的R分量取值

g：条纹的G分量取值

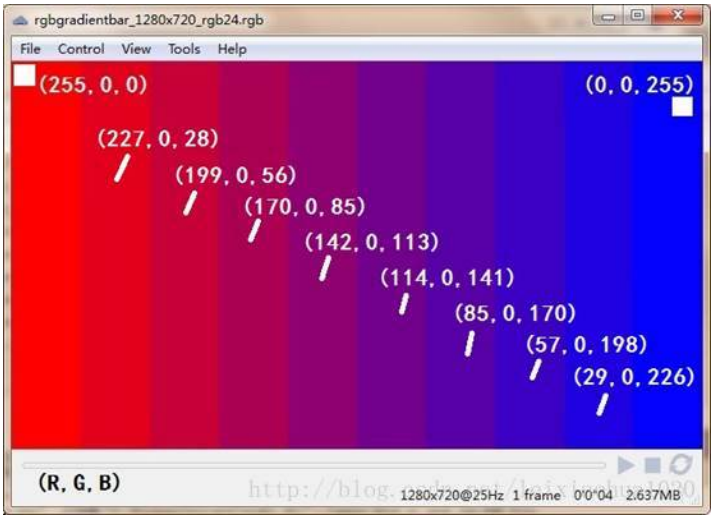
b：条纹的B分量取值

如果函数成功运行的话，会生成一个名称为“rgbstripe_%dx%d_rgb24.rgb”的RGB24格式的文件（其中%dx%d代表了图像的宽和高）。
例如，生成分辨率为1280x720的上文中的彩色条纹图的代码如下。

```cpp
gen_rgb24_stripe(1280,720,255,0,0);
```

RGB渐变彩条图

下面这张图是一张RGB渐变彩条图的示例。这张图的分辨率是1280x720，一共包含了10个彩条，像素格式是RGB24，RGB颜色从红色（RGB分别取值为255，0，0）逐渐变换为蓝色（RGB分别取值为0，0，255）。



每个彩条的RGB取值如下所列：

```
255,   0,   0
227,   0,  28
199,   0,  56
170,   0,  85
142,   0, 113
114,   0, 141
85,    0, 170
57,    0, 198
29,    0, 226
0,     0, 255
```

在生成渐变彩条图像之后，程序会打印出彩条信号的颜色信息，如下图所示。

函数原型

gen_rgb24_rgbgradient_bar ()是用于生成渐变彩条图的函数，该函数的原型如下。

```cpp
/**
 * Generate Picture contains Color Bar Changing from source color
 * to destination color in RGB24 Format
 *
 * @param width             the width of picture.
 * @param height  the height of picture.
 * @param barnum         the number of Bars in the picture.
 * @param src_r              Red component of source color.
 * @param src_g             Green component of source color.
 * @param src_b             Blue component of source color.
 * @param dst_r             Red component of destination color.
 * @param dst_g             Green component of destination color.
 * @param dst_b             Blue component of destination color.
 * @return 0 if finished, -1 if there are errors.
 */
int gen_rgb24_rgbgradient_bar(int width, int height,int barnum,
         unsigned char src_r,unsigned char src_g,unsigned char src_b,
         unsigned char dst_r,unsigned char dst_g,unsigned char dst_b)
```

简单解释每个参数的含义：

width：图像宽

height：图像高

barnum：彩条数量

src_r：左侧颜色R分量

src_g：左侧颜色G分量

src_b：左侧颜色B分量

dst_r：右侧颜色R分量

dst_g：右侧颜色G分量

dst_b：右侧颜色B分量

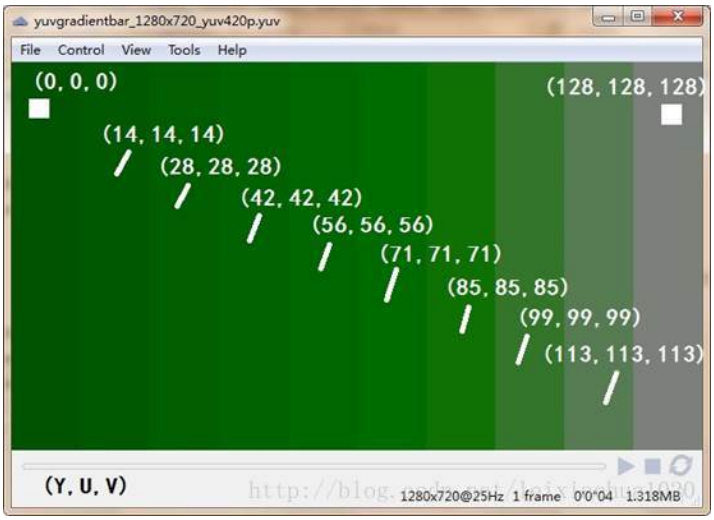
如果函数成功运行的话，会生成一个名称为“rgbgradientbar_%dx%d_rgb24.rgb”的RGB24格式的文件（其中%dx%d代表了图像的宽和高）。

例如，生成分辨率为1280x720的上文中的渐变彩条图的代码如下。

```cpp
gen_rgb24_rgbgradient_bar(1280,720,10,255,0,0,0,0,255);
```

YUV渐变彩条图

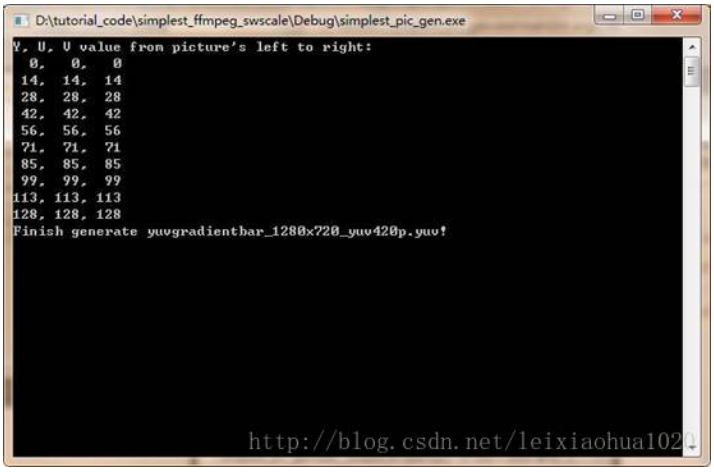
下面这张图是一张YUV渐变彩条图的示例。这张图的分辨率是1280x720，一共包含了10个彩条，像素格式是YUV420P，YUV颜色从绿色（YUV分别取值为0，0，0）逐渐变换为灰色（YUV分别取值为128，128，128）。



每个彩条的YUV取值如下所列：

```
0,  0,  0
14,  14,  14
28,  28,  28
42,  42,  42
56,  56,  56
71,  71,  71
85,  85,  85
99,  99,  99
113, 113, 113
128, 128, 128
```

在生成渐变彩条图像之后，程序会打印出彩条信号的颜色信息，如下图所示。



函数原型

gen_yuv420p_yuvgradient_bar()是用于生成渐变彩条图的函数，该函数的原型如下。

```cpp
/**
 * Generate Picture contains Color Bar Changing from source color
 * to destination color in YUV420P Format
 *
 * @param width             the width of picture.
 * @param height  the height of picture.
 * @param barnum         the number of Bars in the picture.
 * @param src_y             Luma component of source color.
 * @param src_u             U component of source color.
 * @param src_v             V component of source color.
 * @param dst_y             Luma component of destination color.
 * @param dst_u             U component of destination color.
 * @param dst_v             V component of destination color.
 * @return 0 if finished, -1 if there are errors.
 */
int gen_yuv420p_yuvgradient_bar(int width, int height,int barnum,
         unsigned char src_y,unsigned char src_u,unsigned char src_v,
         unsigned char dst_y,unsigned char dst_u,unsigned char dst_v)
```

简单解释每个参数的含义：

width：图像宽

height：图像高

barnum：彩条数量

src_y：左侧颜色Y分量

src_u：左侧颜色U分量

src_v：左侧颜色V分量

dst_y：右侧颜色Y分量

dst_u：右侧颜色U分量

dst_v：右侧颜色V分量

如果函数成功运行的话，会生成一个名称为“yuvgradientbar_%dx%d_yuv420p.yuv”的YUV420P格式的文件（其中%dx%d代表了图像的宽和高）。
例如，生成分辨率为1280x720的上文中的渐变彩条图的代码如下。

```cpp
gen_yuv420p_yuvgradient_bar(1280,720,10,0,0,0,128,128,128);
```

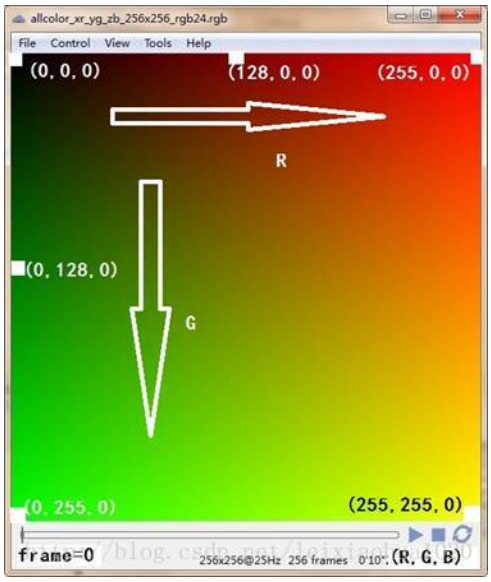
颜色视频

RGB颜色视频

“RGB颜色视频”不是一幅图像，而是一段视频文件。这个视频中包含了RGB24中的所有颜色。通过这个视频，可以了解RGB各个分量对颜色的影响。下面简单记录一下这个视频的规则：

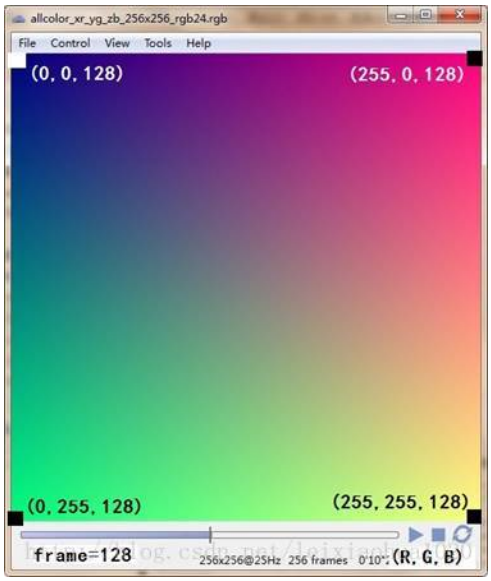
- 视频的宽为256，高为256，视频的帧数为256
- 最左边的像素的R分量取值为0，从左至右每个像素的R分量的取值依次加1
- 最上面的像素的G分量取值为0，从上至下每个像素的G分量的取值依次加1
- 第1帧的所有像素的B分量取值为0，每增加一帧该帧像素的B分量的取值依次加1

所以可以理解为一个坐标系，原点在视频的左上角，X轴对应R分量，Y轴对应G分量，Z轴（时间轴）对应B分量。
该视频的第0帧如下图所示。

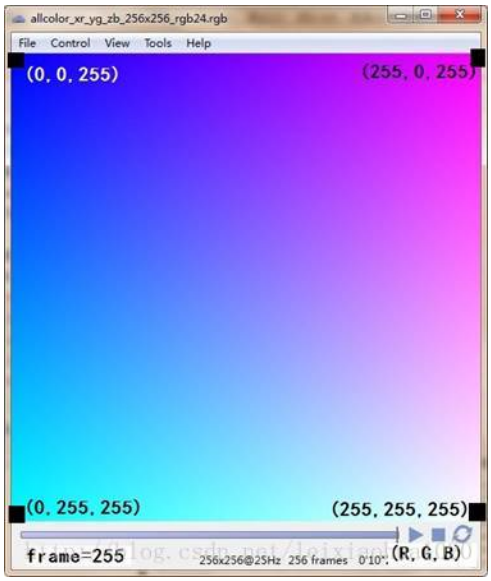


从图中可以看出，左上角为黑色（R，G，B取值0，0，0）；右上角为红色（R，G，B取值0，0，255）；左下角为绿色（R，G，B取值0，255，0）；右下角为黄色（R，G，B取值255，255，0）。

该视频的第128帧如下图所示。



可以看出当蓝色分量增加至128的时候，颜色发生了较大的变化。
该视频的第255帧如下图所示。



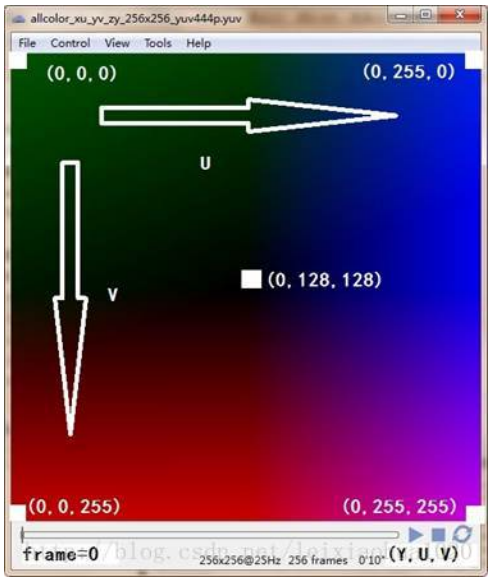
从图中可以看出，左上角为蓝色（R，G，B取值0，0，255）；右上角为品色（R，G，B取值255，0，255）；左下角为青色（R，G，B取值0，255，255）；右下角为白色（R，G，B取值255，255，255）。

YUV颜色视频

“RGB颜色视频”中包含了YUV444中的所有颜色。通过这个视频，可以了解YUV各个分量对颜色的影响。下面简单记录一下这个视频的规则：

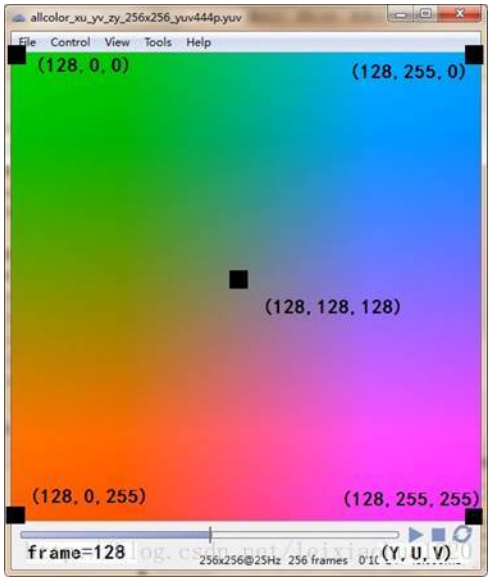
- 视频的宽为256，高为256，视频的帧数为256
- 最左边的像素的U分量取值为0，从左至右每个像素的U分量的取值依次加1
- 最上面的像素的V分量取值为0，从上至下每个像素的V分量的取值依次加1
- 第1帧的所有像素的Y分量取值为0，每增加一帧该帧像素的Y分量的取值依次加1

所以可以理解为一个坐标系，原点在视频的左上角，X轴对应U分量，Y轴对应V分量，Z轴（时间轴）对应Y分量。
该视频的第0帧如下图所示。

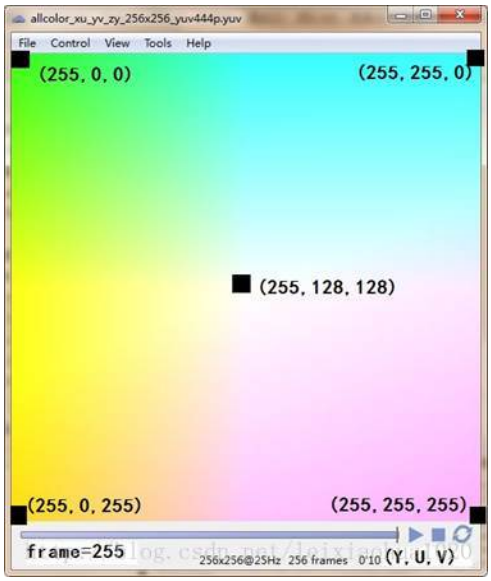


从图中可以看出，左上角颜色偏绿（Y，U，V取值0，0，0）；右上角颜色偏蓝（Y，U，V取值0，0，255）；左下角颜色偏红（Y，U，V取值0，255，0）；右下角颜色偏品色（Y，U，V取值255，255，0）。而正中央是黑色（Y，U，V取值0，128，128）。
在这个地方可能很多人会有疑问，认为Y，U，V取值为0，0，0的时候按理说应该是黑色。实际上U，V是加了偏置的分量，而偏置量就是128。所以“纯正”的黑色实际上对应的是Y，U，V取值为0，128，128的颜色。

该视频的第128帧如下图所示。



可以看出随着Y分量的增加，颜色发生了一些变化。
该视频的第255帧如下图所示。



可以看出，尽管Y分量从0增长到255，但是实际上色调变化不大，只是亮度变化很大。这是因为U，V分量存储了色度信息，Y分量存储了亮度信息。

函数原型

gen_allcolor_video()是用于生成渐变彩条图的函数，该函数的原型如下。

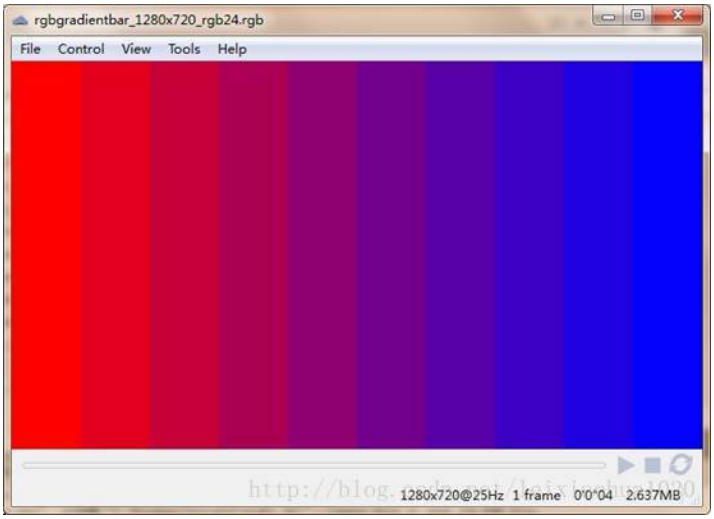
```cpp
/**
 * Generate a video in 256x256 and has 256 frames that contains all the colors.
 * Each color is shown in 1 pixel. They are mapped as follows:
 * In RGB24
 * R component's value is increasing with the growth of width (X-axis);
 * G component's value is increasing with the growth of height (Y-axis);
 * B component's value is increasing with the growth of frame number (Z-axis).
 * In YUV444P
 * U component's value is increasing with the growth of width (X-axis);
 * V component's value is increasing with the growth of height (Y-axis);
 * Y component's value is increasing with the growth of frame number (Z-axis).
 *
 * This function now support to draw YUV444P/RGB24 format pixel.
 *
 * @return 0 if finished, -1 if there are errors.
 */
int gen_allcolor_video();
```

该函数没有参数，直接调用即可生成上述视频。

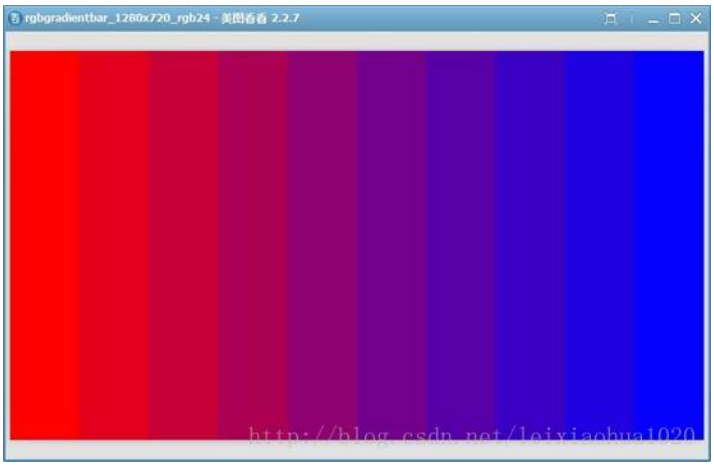
工具函数：RGB24转BMP

本工具除了可以生成测试图片外，还提供了一个简单的工具函数：RGB24转BMP。经过转换后，原本只能用专用的RGB/YUV播放器查看的像素数据，就可以直接拿图片浏览器查看了。

例如输入的RGB24像素数据如下所示。



而输出的BMP图片如下所示。



RGB24转换BMP有以下2个关键点：

（1） 在RGB数据前面加上文件头

（2） 把RGB24数据中的“R”和“B”位置互换（因为BMP中的RGB24实际的存储方式是bgrbgrbgr…）。

源代码

```cpp
/**
 * 最简单的测试图片生成工具
 * Simplest Pic Gen
 *
 * 雷霄骅 Lei Xiaohua
 * leixiaohua1020@126.com
 * 中国传媒大学/数字电视技术
 * Communication University of China / Digital TV Technology
 * http://blog.csdn.net/leixiaohua1020
 *
 * 本程序可以生成多种RGB/YUV格式的测试图像。包括：
 * 灰阶图         [YUV420P]
 * 彩条图         [RGB24]
 * 彩色条纹图     [RGB24]
 * RGB渐变彩条图 [RGB24]
 * YUV渐变彩条图 [YUV420P]
 * 颜色视频       [RGB24][YUV444P]
 *
 * This software can generate several picture that used for
 * test:
 * Gray Bar Picture         [YUV420P]
 * Color Bar Picture        [RGB24]
 * Color Stripe Picture     [RGB24]
 * RGB Gradient Bar Picture [RGB24]
 * YUV Gradient Bar Picture [YUV420P]
 * All Color Video          [RGB24][YUV444P]
 *
 */

#include <stdio.h>
#include <malloc.h>


/**
 * Generate Picture contains Stripe in RGB24 Format
 *
 * @param width     the width of picture.
 * @param height    the height of picture.
 * @param r         Red component of stripe
 * @param g         Green component of stripe
 * @param b         Blue component of stripe
 * @return 0 if finished, -1 if there are errors.
 */
int gen_rgb24_stripe(int width, int height,
    unsigned char r,unsigned char g,unsigned char b){

    unsigned char *data=NULL;
    char filename[100]={0};
    FILE *fp=NULL;
    int i=0,j=0;

    //Check
    if(width<=0||height<=0){
        printf("Error: Width, Height cannot be 0 or negative number!\n");
        printf("Default Param is used.\n");
        width=640;
        height=480;
    }

```

```
    data=(unsigned char *)malloc(width*height*3);

    sprintf(filename,"rgbstripe_%dx%d_rgb24.rgb",width,height);
    if((fp=fopen(filename,"wb+"))==NULL){
        printf("Error: Cannot create file!");
        return -1;
    }

    for(j=0;j<height;j++){
        for(i=0;i<width;i++){
            if(i%2!=0){
                data[(j*width+i)*3+0]=r;
                data[(j*width+i)*3+1]=g;
                data[(j*width+i)*3+2]=b;
            }else{//White
                data[(j*width+i)*3+0]=255;
                data[(j*width+i)*3+1]=255;
                data[(j*width+i)*3+2]=255;
            }
        }
    }
    fwrite(data,width*height*3,1,fp);
    fclose(fp);
    free(data);
    printf("Finish generate %s!\n",filename);
    return 0;
}


/**
 * Generate Picture contains Gray Bar changing from Black to White in YUV420P Format
 *
 * @param width     the width of picture.
 * @param height    the height of picture.
 * @param barnum    the number of Bars in the picture.
 * @param ymin      the minimum value of luminance.
 * @param ymax      the maximum value of luminance.
 * @return 0 if finished, -1 if there are errors.
 */
int gen_yuv420p_graybar(int width, int height,int barnum,unsigned char ymin,unsigned char ymax){

    int barwidth;
    float lum_inc;
    unsigned char lum_temp;
    int uv_width,uv_height;
    FILE *fp=NULL;
    unsigned char *data_y=NULL;
    unsigned char *data_u=NULL;
    unsigned char *data_v=NULL;
    int t=0,i=0,j=0;
    char filename[100]={0};

    //Check
    if(width<=0||height<=0||barnum<=0){
        printf("Error: Width, Height or Bar Number cannot be 0 or negative number!\n");
        printf("Default Param is used.\n");
        width=640;
        height=480;
        barnum=10;
    }
    if(width%barnum!=0){
        printf("Warning: Width cannot be divided by Bar Number without remainder!\n");
    }
    barwidth=width/barnum;
    lum_inc=((float)(ymax-ymin))/((float)(barnum-1));
    uv_width=width/2;
    uv_height=height/2;

    data_y=(unsigned char *)malloc(width*height);
    data_u=(unsigned char *)malloc(uv_width*uv_height);
    data_v=(unsigned char *)malloc(uv_width*uv_height);

    sprintf(filename,"graybar_%dx%d_yuv420p.yuv",width,height);
    if((fp=fopen(filename,"wb+"))==NULL){
        printf("Error: Cannot create file!");
        return -1;
    }

    //Output Info
    printf("Y, U, V value from picture's left to right:\n");
    for(t=0;t<(width/barwidth);t++){
        lum_temp=ymin+(char)(t*lum_inc);
        printf("%3d, 128, 128\n",lum_temp);
    }
    //Gen Data
    for(j=0;j<height;j++){
        for(i=0;i<width;i++){
            t=i/barwidth;
            lum_temp=ymin+(char)(t*lum_inc);
            data_y[j*width+i]=lum_temp;
        }
```

```
    }
    for(j=0;j<uv_height;j++){
        for(i=0;i<uv_width;i++){
            data_u[j*uv_width+i]=128;
        }
    }
    for(j=0;j<uv_height;j++){
        for(i=0;i<uv_width;i++){
            data_v[j*uv_width+i]=128;
        }
    }
    fwrite(data_y,width*height,1,fp);
    fwrite(data_u,uv_width*uv_height,1,fp);
    fwrite(data_v,uv_width*uv_height,1,fp);
    fclose(fp);
    free(data_y);
    free(data_u);
    free(data_v);
    printf("Finish generate %s!\n",filename);
}

/**
 * Generate Picture contains standard Color Bar in RGB24 Format
 *
 * @param width     the width of picture.
 * @param height    the height of picture.
 * @return 0 if finished, -1 if there are errors.
 */
int gen_rgb24_colorbar(int width, int height){

    unsigned char *data=NULL;
    int barwidth;
    char filename[100]={0};
    FILE *fp=NULL;
    int i=0,j=0;
    int lum;
    float r_coeff=0.299,g_coeff=0.587,b_coeff=0.114;

    //Check
    if(width<=0||height<=0){
        printf("Error: Width, Height cannot be 0 or negative number!\n");
        printf("Default Param is used.\n");
        width=640;
        height=480;
    }
    if(width%8!=0)
        printf("Warning: Width cannot be divided by Bar Number without remainder!\n");

    data=(unsigned char *)malloc(width*height*3);
    barwidth=width/8;

    sprintf(filename,"colorbar_%dx%d_rgb24.rgb",width,height);
    if((fp=fopen(filename,"wb+"))==NULL){
        printf("Error: Cannot create file!");
        return -1;
    }

    printf("Luminance (Y) component value of colors from left to right:\n");
    lum=r_coeff*255.0+g_coeff*255.0+b_coeff*255.0;
    printf("[White]  \tR,G,B=255,255,255\t Y=%.3f*R+%.3f*G+%.3f*B=%3d\n",
        r_coeff,g_coeff,b_coeff,lum);
    lum=r_coeff*255.0+g_coeff*255.0+b_coeff*0.0;
    printf("[Yellow] \tR,G,B=255,255,  0\t Y=%.3f*R+%.3f*G+%.3f*B=%3d\n",
        r_coeff,g_coeff,b_coeff,lum);
    lum=r_coeff*0.0+g_coeff*255.0+b_coeff*255.0;
    printf("[Cyan]   \tR,G,B=  0,255,255\t Y=%.3f*R+%.3f*G+%.3f*B=%3d\n",
        r_coeff,g_coeff,b_coeff,lum);
    lum=r_coeff*0.0+g_coeff*255.0+b_coeff*0.0;
    printf("[Green]  \tR,G,B=  0,255,  0\t Y=%.3f*R+%.3f*G+%.3f*B=%3d\n",
        r_coeff,g_coeff,b_coeff,lum);
    lum=r_coeff*255.0+g_coeff*0.0+b_coeff*255.0;
    printf("[Magenta]\tR,G,B=255,  0,255\t Y=%.3f*R+%.3f*G+%.3f*B=%3d\n",
        r_coeff,g_coeff,b_coeff,lum);
    lum=r_coeff*255.0+g_coeff*0.0+b_coeff*0.0;
    printf("[Red]    \tR,G,B=255,  0,  0\t Y=%.3f*R+%.3f*G+%.3f*B=%3d\n",
        r_coeff,g_coeff,b_coeff,lum);
    lum=r_coeff*0.0+g_coeff*0.0+b_coeff*255.0;
    printf("[Blue]   \tR,G,B=  0,  0,255\t Y=%.3f*R+%.3f*G+%.3f*B=%3d\n",
        r_coeff,g_coeff,b_coeff,lum);
    lum=r_coeff*0.0+g_coeff*0.0+b_coeff*0.0;
    printf("[Black]  \tR,G,B=  0,  0,  0\t Y=%.3f*R+%.3f*G+%.3f*B=%3d\n",
        r_coeff,g_coeff,b_coeff,lum);

    for(j=0;j<height;j++){
        for(i=0;i<width;i++){
            int barnum=i/barwidth;
            switch(barnum){
            case 0:{
                data[(j*width+i)*3+0]=255;
                data[(j*width+i)*3+1]=255;
                data[(j*width+i)*3+2]=255;
```

```
                break;
                   }
            case 1:{
                data[(j*width+i)*3+0]=255;
                data[(j*width+i)*3+1]=255;
                data[(j*width+i)*3+2]=0;
                break;
                   }
            case 2:{
                data[(j*width+i)*3+0]=0;
                data[(j*width+i)*3+1]=255;
                data[(j*width+i)*3+2]=255;
                break;
                   }
            case 3:{
                data[(j*width+i)*3+0]=0;
                data[(j*width+i)*3+1]=255;
                data[(j*width+i)*3+2]=0;
                break;
                   }
            case 4:{
                data[(j*width+i)*3+0]=255;
                data[(j*width+i)*3+1]=0;
                data[(j*width+i)*3+2]=255;
                break;
                   }
            case 5:{
                data[(j*width+i)*3+0]=255;
                data[(j*width+i)*3+1]=0;
                data[(j*width+i)*3+2]=0;
                break;
                   }
            case 6:{
                data[(j*width+i)*3+0]=0;
                data[(j*width+i)*3+1]=0;
                data[(j*width+i)*3+2]=255;

                break;
                   }
            case 7:{
                data[(j*width+i)*3+0]=0;
                data[(j*width+i)*3+1]=0;
                data[(j*width+i)*3+2]=0;
                break;
                   }
            }

        }
    }
    fwrite(data,width*height*3,1,fp);
    fclose(fp);
    free(data);
    printf("Finish generate %s!\n",filename);
}

/**
 * Generate Picture contains Color Bar Changing from source color
 * to destination color in RGB24 Format
 *
 * @param width     the width of picture.
 * @param height    the height of picture.
 * @param barnum    the number of Bars in the picture.
 * @param src_r     Red component of source color.
 * @param src_g     Green component of source color.
 * @param src_b     Blue component of source color.
 * @param dst_r     Red component of destination color.
 * @param dst_g     Green component of destination color.
 * @param dst_b     Blue component of destination color.
 * @return 0 if finished, -1 if there are errors.
 */
int gen_rgb24_rgbgradient_bar(int width, int height,int barnum,
    unsigned char src_r,unsigned char src_g,unsigned char src_b,
    unsigned char dst_r,unsigned char dst_g,unsigned char dst_b){

    unsigned char *data=NULL;
    int barwidth;
    float r_inc,g_inc,b_inc;
    unsigned char r_temp,g_temp,b_temp;
    char filename[100]={0};
    FILE *fp=NULL;
    int t=0,i=0,j=0;

    //Check
    if(width<=0||height<=0||barnum<=0){
        printf("Error: Width, Height or Bar Number cannot be 0 or negative number!\n");
        printf("Default Param is used.\n");
        width=640;
        height=480;
    }
    if(width%barnum!=0)
        printf("Warning: Width cannot be divided by Bar Number without remainder!\n");
```

```
    data=(unsigned char *)malloc(width*height*3);
    barwidth=width/barnum;
    r_inc=((float)(dst_r-src_r))/((float)(barnum-1));
    g_inc=((float)(dst_g-src_g))/((float)(barnum-1));
    b_inc=((float)(dst_b-src_b))/((float)(barnum-1));

    sprintf(filename,"rgbgradientbar_%dx%d_rgb24.rgb",width,height);
    if((fp=fopen(filename,"wb+"))==NULL){
        printf("Error: Cannot create file!");
        return -1;
    }

    //Output Info
    printf("R, G, B value from picture's left to right:\n");
    for(t=0;t<(width/barwidth);t++){
        r_temp=src_r+(char)(t*r_inc);
        g_temp=src_g+(char)(t*g_inc);
        b_temp=src_b+(char)(t*b_inc);
        printf("%3d, %3d, %3d\n",r_temp,g_temp,b_temp);
    }

    for(j=0;j<height;j++){
        for(i=0;i<width;i++){
            t=i/barwidth;
            r_temp=src_r+(char)(t*r_inc);
            g_temp=src_g+(char)(t*g_inc);
            b_temp=src_b+(char)(t*b_inc);
            data[(j*width+i)*3+0]=r_temp;
            data[(j*width+i)*3+1]=g_temp;
            data[(j*width+i)*3+2]=b_temp;
        }
    }
    fwrite(data,width*height*3,1,fp);
    fclose(fp);
    free(data);
    printf("Finish generate %s!\n",filename);
    return 0;
}

/**
 * Generate Picture contains Color Bar Changing from source color
 * to destination color in YUV420P Format
 *
 * @param width     the width of picture.
 * @param height    the height of picture.
 * @param barnum    the number of Bars in the picture.
 * @param src_y     Luma component of source color.
 * @param src_u     U component of source color.
 * @param src_v     V component of source color.
 * @param dst_y     Luma component of destination color.
 * @param dst_u     U component of destination color.
 * @param dst_v     V component of destination color.
 * @return 0 if finished, -1 if there are errors.
 */
int gen_yuv420p_yuvgradient_bar(int width, int height,int barnum,
    unsigned char src_y,unsigned char src_u,unsigned char src_v,
    unsigned char dst_y,unsigned char dst_u,unsigned char dst_v){

    int uv_width,uv_height;
    unsigned char *data_y=NULL;
    unsigned char *data_u=NULL;
    unsigned char *data_v=NULL;
    FILE *fp=NULL;
    int barwidth,uv_barwidth;
    float y_inc,u_inc,v_inc=0;
    unsigned char y_temp,u_temp,v_temp=0;
    char filename[100]={0};
    int t=0,i=0,j=0;
    //Check
    if(width<=0||height<=0||barnum<=0){
        printf("Error: Width, Height or Bar Number cannot be 0 or negative number!\n");
        printf("Default Param is used.\n");
        width=640;
        height=480;
    }
    if(width%barnum!=0)
        printf("Warning: Width cannot be divided by Bar Number without remainder!\n");

    uv_width=width/2;
    uv_height=height/2;
    data_y=(unsigned char *)malloc(width*height);
    data_u=(unsigned char *)malloc(uv_width*uv_height);
    data_v=(unsigned char *)malloc(uv_width*uv_height);
    barwidth=width/barnum;
    uv_barwidth=barwidth/(width/uv_width);
    y_inc=((float)(dst_y-src_y))/((float)(barnum-1));
    u_inc=((float)(dst_u-src_u))/((float)(barnum-1));
    v_inc=((float)(dst_v-src_v))/((float)(barnum-1));

```

```
    sprintf(filename,"yuvgradientbar_%dx%d_yuv420p.yuv",width,height);
    if((fp=fopen(filename,"wb+"))==NULL){
        printf("Error: Cannot create file!");
        return -1;
    }

    //Output Info
    printf("Y, U, V value from picture's left to right:\n");
    for(t=0;t<(width/barwidth);t++){
        y_temp=src_y+(char)(t*y_inc);
        u_temp=src_u+(char)(t*u_inc);
        v_temp=src_v+(char)(t*v_inc);
        printf("%3d, %3d, %3d\n",y_temp,u_temp,v_temp);
    }

    //Gen Data
    for(j=0;j<height;j++){
        for(i=0;i<width;i++){
            t=i/barwidth;
            y_temp=src_y+(char)(t*y_inc);
            data_y[j*width+i]=y_temp;
        }
    }
    for(j=0;j<uv_height;j++){
        for(i=0;i<uv_width;i++){
            t=i/uv_barwidth;
            u_temp=src_u+(char)(t*u_inc);
            data_u[j*uv_width+i]=u_temp;
        }
    }
    for(j=0;j<uv_height;j++){
        for(i=0;i<uv_width;i++){
            t=i/uv_barwidth;
            v_temp=src_v+(char)(t*v_inc);
            data_v[j*uv_width+i]=v_temp;
        }
    }
    fwrite(data_y,width*height,1,fp);
    fwrite(data_u,uv_width*uv_height,1,fp);
    fwrite(data_v,uv_width*uv_height,1,fp);
    fclose(fp);
    free(data_y);
    free(data_u);
    free(data_v);
    printf("Finish generate %s!\n",filename);
    return 0;
}

/**
 * Convert RGB24 format to BMP format
 *
 * @param rgb24path     path of input RGB24 file.
 * @param bmppath       path of output BMP file
 * @param width         the width of picture.
 * @param height        the height of picture.
 * @return 0 if finished, -1 if there are errors.
 */
int rgb24_to_bmp(char *rgb24path,char *bmppath,int width,int height)
{
    typedef struct
    {
        long imageSize;
        long blank;
        long startPosition;
    }BmpHead;

    typedef struct
    {
        long  Length;
        long  width;
        long  height;
        unsigned short  colorPlane;
        unsigned short  bitColor;
        long  zipFormat;
        long  realSize;
        long  xPels;
        long  yPels;
        long  colorUse;
        long  colorImportant;
    }InfoHead;

    int i=0,j=0;
    BmpHead m_BMPHeader={0};
    InfoHead  m_BMPInfoHeader={0};
    char bfType[2]={'B','M'};
    int header_size=sizeof(bfType)+sizeof(BmpHead)+sizeof(InfoHead);
    unsigned char *rgb24_buffer=NULL;
    FILE *fp_rgb24=NULL,*fp_bmp=NULL;

    if((fp_rgb24=fopen(rgb24path,"rb"))==NULL){
        printf("Error: Cannot open input RGB24 file.\n");
        return -1;
```

```
        return -1;
    }
    if((fp_bmp=fopen(bmppath,"wb"))==NULL){
        printf("Error: Cannot open output BMP file.\n");
        return -1;
    }

    rgb24_buffer=(unsigned char *)malloc(width*height*3);
    fread(rgb24_buffer,1,width*height*3,fp_rgb24);

    m_BMPHeader.imageSize=3*width*height+header_size;
    m_BMPHeader.startPosition=header_size;

    m_BMPInfoHeader.Length=sizeof(InfoHead);
    m_BMPInfoHeader.width=width;
    //BMP storage pixel data in opposite direction of Y-axis (from bottom to top).
    m_BMPInfoHeader.height=-height;
    m_BMPInfoHeader.colorPlane=1;
    m_BMPInfoHeader.bitColor=24;
    m_BMPInfoHeader.realSize=3*width*height;

    fwrite(bfType,1,sizeof(bfType),fp_bmp);
    fwrite(&m_BMPHeader,1,sizeof(m_BMPHeader),fp_bmp);
    fwrite(&m_BMPInfoHeader,1,sizeof(m_BMPInfoHeader),fp_bmp);

    //BMP save R1|G1|B1,R2|G2|B2 as B1|G1|R1,B2|G2|R2
    //It saves pixel data in Little Endian
    //So we change 'R' and 'B'
    for(j =0;j<height;j++){
        for(i=0;i<width;i++){
            char temp=rgb24_buffer[(j*width+i)*3+2];
            rgb24_buffer[(j*width+i)*3+2]=rgb24_buffer[(j*width+i)*3+0];
            rgb24_buffer[(j*width+i)*3+0]=temp;
        }
    }
    fwrite(rgb24_buffer,3*width*height,1,fp_bmp);
    fclose(fp_rgb24);
    fclose(fp_bmp);
    free(rgb24_buffer);
    printf("Finish generate %s!\n",bmppath);
    return 0;
}



/**
 * Generate a video in 256x256 and has 256 frames that contains all the colors.
 * Each color is shown in 1 pixel. They are mapped as follows:
 * In RGB24
 * R component's value is increasing with the growth of width (X-axis);
 * G component's value is increasing with the growth of height (Y-axis);
 * B component's value is increasing with the growth of frame number (Z-axis).
 * In YUV444P
 * U component's value is increasing with the growth of width (X-axis);
 * V component's value is increasing with the growth of height (Y-axis);
 * Y component's value is increasing with the growth of frame number (Z-axis).
 *
 * This function now support to draw YUV444P/RGB24 format pixel.
 *
 * @return 0 if finished, -1 if there are errors.
 */
int gen_allcolor_video(){

    unsigned char *data=NULL;
    char filename[100]={0};
    FILE *fp=NULL;
    int width=256,height=256,frames=256;
    int i=0,j=0,k=0;

    //From left to right (width, X-axis),R increasing from 0 to255
    //From Top to bottom (height, Y-axis),G increasing from 0 to255
    //From 0 to 255 frames (time, Z-axis),B increasing from 0 to255
    data=(unsigned char *)malloc(width*height*3);
    sprintf(filename,"allcolor_xr_yg_zb_%dx%d_rgb24.rgb",width,height);
    if((fp=fopen(filename,"wb+"))==NULL){
        printf("Error: Cannot create file!");
        return -1;
    }
    for(k=0;k<frames;k++){
        for(j=0;j<height;j++){
            for(i=0;i<width;i++){
                data[(j*width+i)*3+0]=i;
                data[(j*width+i)*3+1]=j;
                data[(j*width+i)*3+2]=k;
            }
        }
        fwrite(data,width*height*3,1,fp);
        printf("Finish generate frame %d!\n",k);
    }
    fclose(fp);
    free(data);
```

```
    printf("Finish generate %s!\n",filename);

    //From left to right (width, X-axis),U increasing from 0 to255
    //From Top to bottom (height, Y-axis),V increasing from 0 to255
    //From 0 to 255 frames (time, Z-axis),Y increasing from 0 to255
    data=(unsigned char *)malloc(width*height);
    sprintf(filename,"allcolor_xu_yv_zy_%dx%d_yuv444p.yuv",width,height);
    if((fp=fopen(filename,"wb+"))==NULL){
        printf("Error: Cannot create file!");
        return -1;
    }
    for(k=0;k<frames;k++){
        for(j=0;j<height;j++){//Y
            for(i=0;i<width;i++){
                data[j*width+i]=k;
            }
        }
        fwrite(data,width*height,1,fp);
        for(j=0;j<height;j++){//U
            for(i=0;i<width;i++){
                data[j*width+i]=i;
            }
        }
        fwrite(data,width*height,1,fp);
        for(j=0;j<height;j++){//V
            for(i=0;i<width;i++){
                data[j*width+i]=j;
            }
        }
        fwrite(data,width*height,1,fp);
        printf("Finish generate frame %d!\n",k);
    }
    fclose(fp);
    free(data);
    printf("Finish generate %s!\n",filename);

    return 0;
}



int main(int argc, char* argv[])
{
    //All picture's resolution is 1280x720
    //Gray Bar, from 16 to 235
    gen_yuv420p_graybar(1280,720,10,16,235);
    //Color Bar
    gen_rgb24_colorbar(1280,720);
    //10 bars, RGB changed from 255,0,0 to 0,0,255
    gen_rgb24_rgbgradient_bar(1280,720,10,255,0,0,0,0,255);
    //10 bars, RGB changed from 0,0,0 to 128,128,128
    gen_yuv420p_yuvgradient_bar(1280,720,10,0,0,0,128,128,128);
    //RGB24 to BMP
    rgb24_to_bmp("colorbar_1280x720_rgb24.rgb","colorbar_1280x720_rgb24.bmp",1280,720);
    //Red stripe
    gen_rgb24_stripe(1280,720,255,0,0);
    //Gen color video
    gen_allcolor_video();
    return 0;
}
```

运行结果

程序运行完后，会生成上文中叙述的几种测试图。

下载

Simplest FFmpeg Swscale

项目主页

SourceForge： https://sourceforge.net/projects/simplestffmpegswscale/

Github： https://github.com/leixiaohua1020/simplest_ffmpeg_swscale

开源中国： http://git.oschina.net/leixiaohua1020/simplest_ffmpeg_swscale

CDSN下载地址： http://download.csdn.net/detail/leixiaohua1020/8292175

本教程是最简单的基于FFmpeg的libswscale进行像素处理的教程。它包含了两个工程：
simplest_ffmpeg_swscale: 最简单的libswscale的教程。
simplest_pic_gen: 生成各种测试图片的工具。

更新-1.1 (2015.2.13)==========

这次考虑到了跨平台的要求，调整了源代码，保证了该项目代码可以在个平台上编译通过。

CSDN下载地址： http://download.csdn.net/detail/leixiaohua1020/8445671

SourceForge上已经更新。

文章标签： 测试图 灰度图 彩条信号 渐变图 颜色

个人分类： 我的开源项目 FFMPEG

所属专栏： FFmpeg