Colour management bricks on Linux

Kai-Uwe Behrmann
Oyranos project

Open Source Overview

- **Linux / BSD are open source**
  - forking and redistribution is ok
- **Posix (common interfaces for Solaris, BSD, Linux, osX ...)**
  - portable C, C++ API's and Posix extensions
  - most API's are ported to Win32 as well
- **License variety (GPL, LGPL, MIT, propriarity is allowed ...)**
  - GPL requires your derived code to be GPL (Ghostscript)
  - LGPL your changes to the library must be given back before distribution (Cairo, KDE and Gnome libraries)
  - new BSD / MIT very liberal (littleCMS)

Available Tools

- **colour tools are almost collections of utils and libraries**
- **CMM's**
  - littleCMS - the most popular
  - SampleCMS (not distributed)
  - CTL (not distributed, non ICC conform format)
- **profilers**
  - ArgyllCMS
  - Scarse
  - Lprof (with GUI)
- **device dependent (mainly about display profiles)**
  - xcalib
  - Oyranos
  - ArgyllCMS

Programm

- raw conversion
- compositing with vectors
- colour match to screen
- proofing on screen
- preparation for displaying and printing

Camera Raw Conversion

- **demosaicing tools**
  - Dcraw, with support from Dave Coffin
  - libopenraw <http://libopenraw.freedesktop.org>
- **device colours**
  - The Linux profile database for devices other than monitors is about evolving.
  - Let the convertor create a ICC profile and remember the given profile for the image data, or
  - maintain a own device ICC profile database.
- **additional considerations**
  - Allow for saving of the device driver parameters and allow the user to associate with a own selected device ICC profile.

Color Compositing

- **Now that the Image is prepared, how to add to the mix of other elements?**
- **Use a intermediate rendering colour space**
  - Consistent results across output devices.
  - Convert all colour elements to a usually big editing space.
  - The editing colour space and the standard rendering intent can be queried from Oyranos.
  - Ask Oyranos for the monitor profile or query the _ICC_PROFILE Xorg atom directly
  - Composite as usual in Rgb
  - Convert the intermediate rendering from editing colour space to the monitor colour space.
  - Write to the framebuffer, by whatever means of your toolkit..

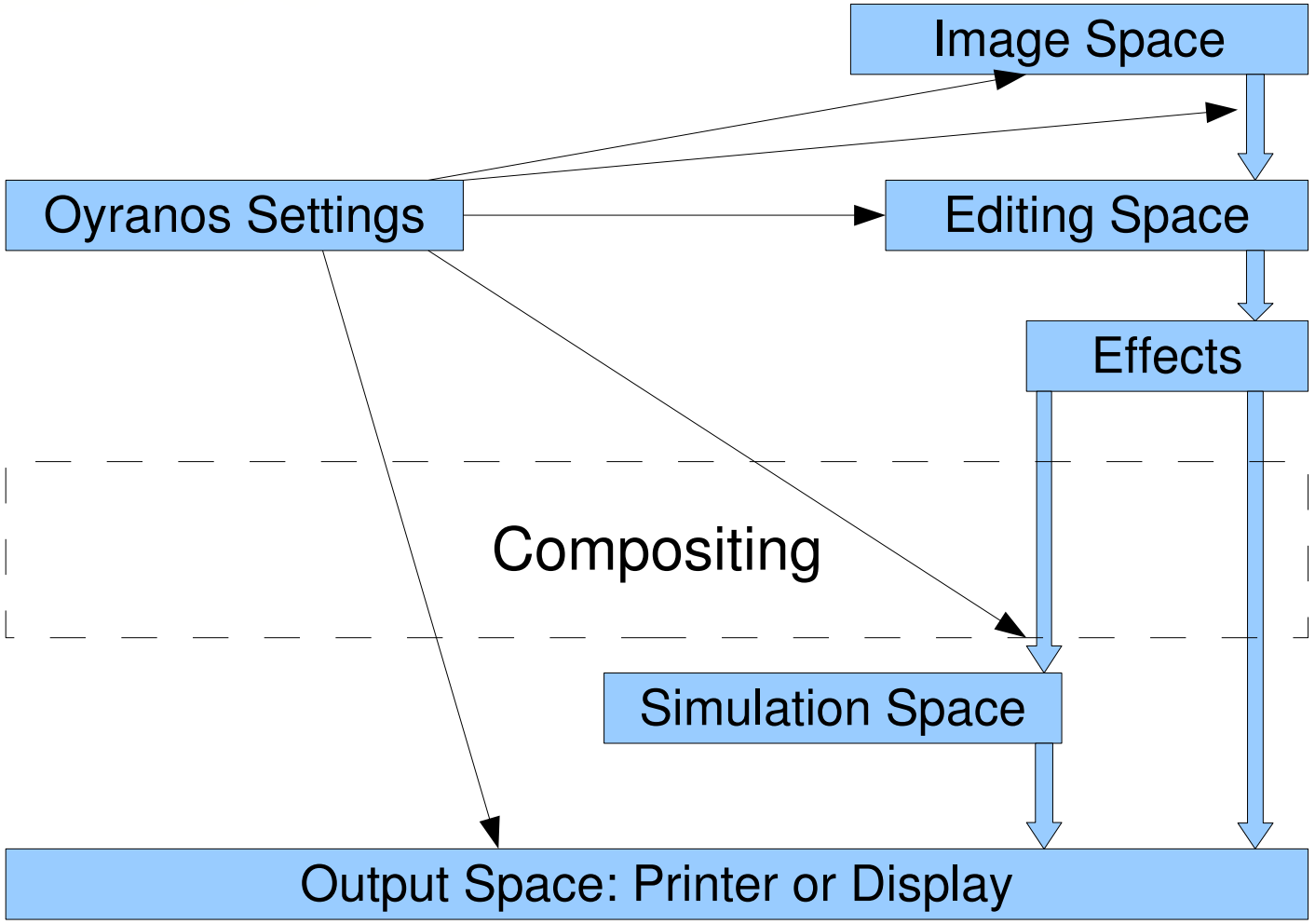
Color Compositing

- **Quick and dirty**
  - Ask Oyranos for the monitor profile or query the _ICC_PROFILE Xorg atom directly
  - The standard rendering intent can be queried from Oyranos.
  - Convert to the output colour space and composite.
  - Risk of artifacts.
  - Write to the framebuffer, by whatever means of your toolkit..

Proofing

- In order to provide a meaningful proof, the application needs to know, which printing queue is selected in advance of doing the proofing.
- CUPS is supposed to deliver the ICC printer profile through a PPD entry. But there is neighter help for parsing the PPD entry nor for fetching a remote ICC profile through CUPS.
- To obtain the printer profile for proofing, a application can only rely on iternal settings. This is similiar to the camera profile.
- Support for device profiles is planed in Oyranos.
- Oyranos provides some user settings for proofing.

Display and Print rendering



I. Sources

- **Code includes and used types**

```
/*  Sources in Oyranos git:  examples/image2pdf.c
 *
 *  Compile: cc -pedantic -Wall -g `oyranos-config --cflags`
    `oyranos-config --ld_x_flags` `pkg-config --cflags cairo`
    `pkg-config --libs cairo` `pkg-config --libs lcms` image2pdf.c
    -o image2pdf
 */

#include <stdio.h>
#include <math.h>
#include <oyranos.h>          /* Oyranos headers */
#include <oyranos_monitor.h>
#include <lcms.h>             /* littleCMS – typical CMM on Linux */
#include <cairo.h>            /* Cairo headers */
#include <cairo-pdf.h>

...

cairo_t * cr = 0;                     /* drawing context */
cairo_surface_t * surface = 0,        /* page surface */
                * image_surf = 0;     /* single image surface */

cmsHPROFILE input, proof, editing, monitor, print;
cmsHTRANSFORM to_output = 0;
```

- **Application target**

II. Sources

```
{
  printf("Merge CamerRAW images into one output image and use\n");
  printf("Oyranos CMS settings to obtain the result.\n");
  printf("\n");
  printf("Usage of the image2pdf example application:");
  printf("  To obtain a PDF (test.pdf):\n");
  printf("    image2pdf imageA.raw imageB.raw\n");
  printf("  To obtain a monitor preview (test.png):\n");
  printf("    image2pdf --monitor imageA.raw imageB.raw\n");
  printf("  To obtain a proofed monitor preview(test_proof.png):\n");
  printf("    image2pdf --monitor --proof imageA.raw imageB.raw\n");
  return 1;
}
```

III. Sources

- **Cairo page layout**

```
double page_w = 210.0,          /* page width in mm */
       page_h = 297.0,
       resolution = 72.0;       /* Cairo PDF surface resolution */

if(strcmp(argv[o],"--monitor") == 0 || strcmp(argv[o],"-m") == 0)
{
  ++o;
  to_moni = 1;
  resolution = 96;
}

pixel_w = page_w / 25.4 * resolution;
pixel_h = page_h / 25.4 * resolution;

/* create a surface to place our images on */
if(to_moni)
  surface = cairo_image_surface_create( CAIRO_FORMAT_ARGB32, pixel_w,pixel_h);
else
  surface = cairo_pdf_surface_create( "test.pdf",
                                      pixel_w,
                                      pixel_h );

cr = cairo_create( surface );
```

- **Passing of Oyranos profiles to littleCMS**

IV. Sources

```
/*  The monitor profile is located in the Xserver. For details see:
 *  http://www.freedesktop.org/wiki/Specifications/icc_profiles_in_x_spec
 */
data = oyGetMonitorProfile( 0, &size, malloc );
if(!size || !data)
{
  profile_name = oyGetDefaultProfileName( oyASSUMED_WEB, malloc );
  data = oyGetProfileBlock( profile_name, &size, malloc );
}
monitor = cmsOpenProfileFromMem( data, size );


/*  The editing profile can be obtained by Oyranos.
 *  It could be used as a blending colour space, if we had full access to the
 *  final document.
 */
profile_name = oyGetDefaultProfileName( oyEDITING_RGB, malloc );
data = oyGetProfileBlock( profile_name, &size, malloc );
editing = cmsOpenProfileFromMem( data, size );
```

- **Passing of Oyranos profiles to littleCMS**

V. Sources

```
/*  The output profile is equal to sRGB, as output profiles are curently not
 *  supported in Cairo.
 */
profile_name = oyGetDefaultProfileName( oyASSUMED_WEB, malloc );
data = oyGetProfileBlock( profile_name, &size, malloc );
print = cmsOpenProfileFromMem( data, size );


/*  The printer profile is not easily available on Linux.
 *  Better maintain a own database for profile to queue profile assignment.
 *  We use here just the Oyranos proofing profile.
 */
profile_name = oyGetDefaultProfileName( oyPROFILE_PROOF, malloc );
data = oyGetProfileBlock( profile_name, &size, malloc );
proof = cmsOpenProfileFromMem( data, size );


/* collect some more parameters */
rendering_intent = oyGetBehaviour( oyBEHAVIOUR_RENDERING_INTENT );
bpc = oyGetBehaviour( oyBEHAVIOUR_RENDERING_BPC );
proof_intent = oyGetBehaviour( oyBEHAVIOUR_RENDERING_INTENT_PROOF ) ?
                        INTENT_ABSOLUTE_COLORIMETRIC :
                        INTENT_RELATIVE_COLORIMETRIC ;
```

- **Profile selection for camera RAW**

VI. Sources

```
/*  1. Embedded profile: We do not know about how to get from DCraw.
 *  2. Exif profile infos: DCraw does not provide as it would make not much
 *     sense. We would have to search for a external exif parser.
 *  3. Device database for a ICC profile: which fits to info[0] + info[1].
 */
image_profile_name = 0;

if(!dcraw_icc_space && image_profile_name)
  dcraw_icc_space = "-o 0"; /* raw */

/*  4. Popup? Does the user want some vote, as there is no profile found
 *     so far? We can ask the Oyranos CMS and decide upon user settings. */
if(!dcraw_icc_space)
switch( oyGetBehaviour (oyBEHAVIOUR_ACTION_UNTAGGED_ASSIGN ))
{ case 0: /* Ignore CM and profiles. Or, in case you cant, say here sRGB. */
          dcraw_icc_space = "-o 1"; /* dcraw linear matrix to gamma sRGB */
          profile_name = oyGetDefaultProfileName( oyASSUMED_WEB, malloc );
          data = oyGetProfileBlock( profile_name, &size, malloc );
          input = cmsOpenProfileFromMem( data, size );
          break;
  case 1: /* 4a. Use the Oyranos standard assumed input Rgb profile.
           *     This does not apply, as the Cairo output is untagged.
                 Thus sRGB is the only option.
           */
  case 2: /* 4b. let the user select a input profile */
          ;/* For simplicity here is no interactive mode provided. */
}
```

VII. Sources

- **Profile selection for camera RAW**

```
/* 4a. Oyranos assumed Rgb profile: our last possibility. */
if(!dcraw_icc_space)
{
  /*  Tell DCraw to do the conversion with the build in matrices.
   */
  dcraw_icc_space = "-o 2"; /* dcraw linear matrix to Adobe Rgb */

  data = createProfile( 1.0, 0.64, 0.33, 0.21, 0.71, 0.15, 0.06,
                        "linearAdobeRGB1998", &size );
  input = cmsOpenProfileFromMem( data, size );
  free( data );
}
```

- **littleCMS colour transform**

```
/* build the colour context with littleCMS */
to_output = cmsCreateProofingTransform( input, TYPE_BGRA_8,
                        to_moni ? monitor : print,
                      /* Cairo uses a Blue Green Red Alpha channel layout */
                        TYPE_BGRA_8,
                        proof,
                        rendering_intent, proof_intent,
                        do_proof ?
                          cmsFLAGS_SOFTPROOFING : 0 |
                         (bpc && rendering_intent == 1) ?
                           cmsFLAGS_WHITEBLACKCOMPENSATION : 0 );
```

VIII. Sources

- **Camera RAW loading**

```
/* decode with camera white balance and pipe half size, 16-bit, be verbose*/
sprintf (command, "PATH=.:$PATH ; dcraw -w -c -h -4 -v %s '%s'\n",
                  dcraw_icc_space, filename );

fp = popen (command, "r");
fscanf (fp, "P6 %d %d %d%c", &w, &h, &depth, &c);

/* create a Cairo image */
image_surf = cairo_image_surface_create( CAIRO_FORMAT_ARGB32, w, h );

/* write our dcraw stream on the Cairo image */
image_data = cairo_image_surface_get_data( image_surf );

/* Cairo uses a Blue Green Red Alpha channel layout */
for(j = 0; j < w*h; ++j)
{
  image_data[j*4+2] = readShort( fp )*255.;
  image_data[j*4+1] = readShort( fp )*255.;
  image_data[j*4+0] = readShort( fp )*255.;
  image_data[j*4+3] = 255;
}
```

- **Colour conversion**

```
/* Convert to the output colour space.
 */
cmsDoTransform( to_output, image_data, image_data, size );
```

- **Draw image(s) and frame(s)**

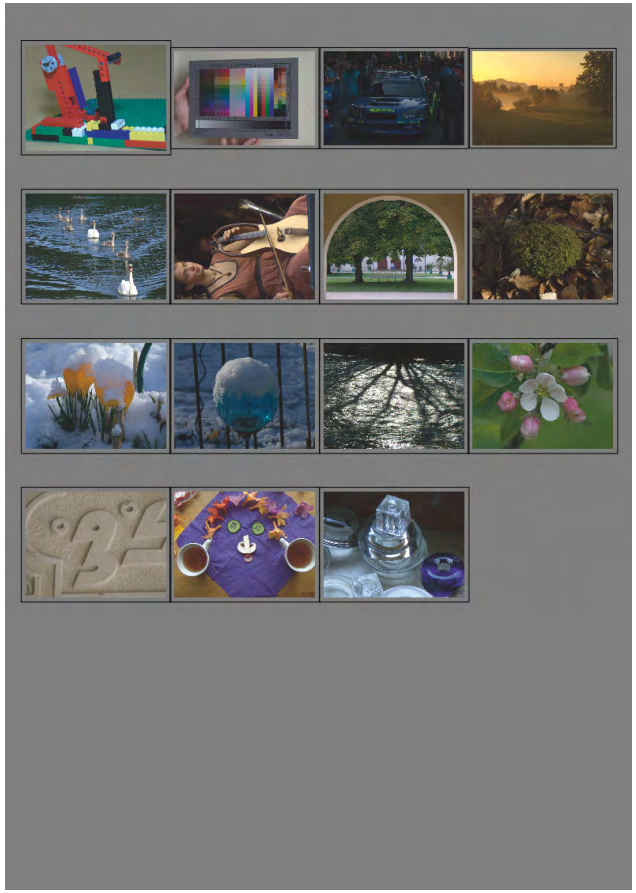
IX. Sources

```
/* draw a frame around the image */
frame = pixel_w/20.0 * scale;
cairo_set_source_rgba( cr, .0, .0, .0, 1.0);
cairo_set_line_width (cr, 1.);
cairo_rectangle( cr, x - frame, y - frame,
                     w*scale + 2*frame, h*scale + 2*frame);
cairo_stroke(cr);

/* draw the image */
cairo_translate( cr, x, y );
cairo_scale( cr, scale, scale );
cairo_set_source_surface( cr, image_surf, 0,0 );
cairo_paint( cr );
cairo_restore( cr );

/* small clean */
cmsCloseProfile( input );
cairo_surface_destroy( image_surf );
fclose( fp );
```

Example



- **Monitor version**
- **Print preview**

Thanks