Version 2.3 Release Notes

The major change is that pixel format renderers were generalized for the use of high quality color spaces (16 bit per component).

Include Files

First of all, there are changes in include file names. The "capacity" numbers were removed from the names and now agg_pixfmt_rgb.h contains formats such as agg::pixfmt_rgb24 and agg::pixfmt_rgb48. Below if the table of changes.

Old file	New File
#include "agg_gray8.h"	#include "agg_color_gray.h"
#include "agg_color_rgba8.h"	#include "agg_color_rgba.h"
#include "agg_pixfmt_rgb24.h"	#include "agg_pixfmt_rgb.h"
#include "agg_pixfmt_rgb555.h"	#include "agg_pixfmt_rgb_packed.h"
#include "agg_pixfmt_rgb565.h"	#include "agg_pixfmt_rgb_packed.h"
#include "agg_pixfmt_rgba32.h"	#include "agg_pixfmt_rgba.h"
#include "agg_pixfmt_gray8.h"	#include "agg_pixfmt_gray.h"
#include "agg_span_gouraud_rgba8.h"	#include "agg_span_gouraud_rgba.h"
#include "agg_span_image_filter_rgb24.h"	#include "agg_span_image_filter_rgb.h"
#include "agg_span_image_filter_rgba32.h"	#include "agg_span_image_filter_rgba.h"
#include "agg_span_pattern_rgba32.h"	#include "agg_span_pattern_rgba.h"

#include "agg_span_pattern_rgb24.h"	#include "agg_span_pattern_rgb.h"
#include "agg_span_gouraud_rgba8.h"	#include "agg_span_gouraud_rgba.h"
#include "agg_span_gouraud_gray8.h"	#include "agg_span_gouraud_gray.h"
#include "agg_span_pattern_filter_rgba32.h"	#include "agg_span_pattern_filter_rgba.h"
#include "agg_span_pattern_filter_rgb24.h"	#include "agg_span_pattern_filter_rgb.h"

Color formats supported by agg_pixfmt_rgb.h:

pixfmt_rgb24	pixfmt_rgb24_pre	pixfmt_rgb24_gamma
pixfmt_bgr24	pixfmt_bgr24_pre	pixfmt_bgr24_gamma
pixfmt_rgb48	pixfmt_rgb48_pre	pixfmt_rgb48_gamma
pixfmt_bgr48	pixfmt_bgr48_pre	pixfmt_bgr48_gamma

Color formats supported by agg_pixfmt_rgba.h:

pixfmt_rgba32	pixfmt_rgba32_pre	pixfmt_rgba32_plain
pixfmt_argb32	pixfmt_argb32_pre	pixfmt_argb32_plain
pixfmt_abgr32	pixfmt_abgr32_pre	pixfmt_abgr32_plain
pixfmt_bgra32	pixfmt_bgra32_pre	pixfmt_bgra32_plain
pixfmt_rgba64	pixfmt_rgba64_pre	
pixfmt_argb64	pixfmt_argb64_pre	
pixfmt_abgr64	pixfmt_abgr64_pre	
pixfmt_bgra64	pixfmt_bgra64_pre	

In agg_pixfmt_rgb_packed.h added exotic formats, 32 bits per ptixel, such as pixfmt_rgbAAA. Suffix "AAA" means "10 bits per component" (0xA, corresponds to r10g10b10", "BBA" means "r11g11b10".

Color formats supported by agg_pixfmt_rgb_packed.h:

pixfmt_rgb555	pixfmt_rgb555_pre	pixfmt_rgb555_gamma
pixfmt_rgb565	pixfmt_rgb565_pre	pixfmt_rgb565_gamma
pixfmt_rgbAAA	pixfmt_rgbAAA_pre	pixfmt_rgbAAA_gamma
pixfmt_bgrAAA	pixfmt_bgrAAA_pre	pixfmt_bgrAAA_gamma
pixfmt_rgbBBA	pixfmt_rgbBBA_pre	pixfmt_rgbBBA_gamma
pixfmt_bgrABB	pixfmt_bgrABB_pre	pixfmt_bgrABB_gamma

File agg_pixfmt_gray.h supports 8- and 16-bit grayscale buffers.

Colors

Now agg_color_rgba.h contains the following definitions: agg::rgba, agg::rgba8, agg::rgba16.

Former agg_gray8.h is renamed to agg_color_gray.h. It's not very logical (gray isn't a color), but it's done for the sake of consistency. It contains agg::gray8 and agg::gray16.

The conversion policy is that any color type must have constructors with agg::rgba (double r,g,b,a in range 0...1) and agg::rgba8 (integer, in range 0...255). I also removed constructors from packed integer, that is, instead of former `agg::rgba8(v, agg::rgba8::rgb)` use functions rgb8_packed(v), bgr8_packed(v), and argb8_packed(v).

Component Orders

It was a mistake to call the RGB and RGBA orders like agg::order_bgr24, because the order of components doesn't depend on the capacity. So, please remove the numeric values from the order names where they are used, i.e., agg::order_bgr24 → agg::order_bgr, etc.

The high capacity colors introduce a problem of byte order on different platforms, that is, Big-Endian/Little-Endian. But handling different byte orders on-the-fly would be too expensive, so that, if you use high capacity colors, you have to take care of the output byte order if it's different from what is used in your hardware.

Gradients

The main change in gradients is that now you can use arrays of colors of any size. Before it was hardcoded 256 colors. As the result, you now cannot just declare a static color array and parametrize the gradient type with it. For example:

Before:

```
typedef agg::span_gradient<agg::rgba8,
                           interpolator_type,
                           gradient_func_type,
                           const agg::rgba8*,
                           span_allocator_type> span_gradient_type;
```

Now:

```
typedef agg::pod_auto_array<agg::rgba8, 256> color_array_type;
typedef agg::span_gradient<agg::rgba8,
                           interpolator_type,
                           gradient_func_type,
                           color_array_type,
                           span_allocator_type> span_gradient_type;
```

You can use any available container as a color array, the gradient template requires just two things to be defined:

```
unsigned size();
const value_type& operator [] (unsigned i);
```

So that, you can use any available container or adaptor, such as: agg::pod_array_adaptor, agg::pod_auto_array, agg::pod_array, agg::pod_deque, or any standard one, such as std::vector.

The same is appicable to agg::span_gradient_alpha.

Gouraud Shaders

The only change in Gouraud shaders is declarations. Just change

agg::span_gouraud_rgba8<>
to
agg::span_gouraud_rgba<agg::rgba8>

You can also use agg::rgba16. Similarly change

agg::span_gouraud_gray8<>
to
agg::span_gouraud_gray<agg::gray8>

Of course, agg::rgba16 and agg::gray16 can be used too.

Image transformers

All image transformers are also generalized:

Before:

```
    typedef agg::span_image_filter_rgb24_bilinear<agg::order_bgr24,
                                                  interpolator_type> span_gen_type;
```

Now:

```
    typedef agg::span_image_filter_rgb_bilinear<agg::rgba8,
                                                agg::order_bgr,
                                                interpolator_type> span_gen_type;
```

Pattern fill and transformers

Patterns are changed in a similar way, plus there's a new possibility to have a wrap function, like in GDI+.

Before:

```
    typedef agg::remainder_auto_pow2 remainder_type;
    typedef agg::span_pattern_filter_rgba32_bilinear<agg::order_bgra32,
                                                     interpolator_type,
                                                     remainder_type,
                                                     remainder_type> span_gen_type;
```

Now:

```
    typedef agg::wrap_mode_repeat_auto_pow2  wrap_type_x;
    typedef agg::wrap_mode_reflect_auto_pow2 wrap_type_y;
    typedef agg::span_pattern_filter_rgba_bilinear<agg::rgba8,
```

```
                                                    agg::order_bgra,
                                                    interpolator_type,
                                                    wrap_type_x,
                                                    wrap_type_y> span_gen_type;
```

The same is about the simple pattern fill (without transformations):

Before:

```
    typedef agg::span_pattern_rgba32<agg::order_rgba32> span_gen_type;
```

Now:

```
    typedef agg::wrap_mode_reflect_auto_pow2 wrap_x_type;
    typedef agg::wrap_mode_reflect_auto_pow2 wrap_y_type;
    typedef agg::span_pattern_rgba<agg::rgba8,
                                   agg::order_rgba,
                                   wrap_x_type,
                                   wrap_y_type> span_gen_type;
```

You can use the following wrap mode adaptors:

```
agg::wrap_mode_repeat            // Arbitrary size (slow)
agg::wrap_mode_repeat_pow2       // Size truncated to the nearest power of 2 (fastest)
agg::wrap_mode_repeat_auto_pow2  // Automatically choosen
agg::wrap_mode_reflect
agg::wrap_mode_reflect_pow2
agg::wrap_mode_reflect_auto_pow2
```

Gray scale imeges support

Added files agg_span_image_filter_gray.h and agg_span_pattern_filter_gray.h to transform 8- and 16-bit gray scale images. The declarations are very similar to the RGB version.