Subject: AGG and Upp Draw integration...
Posted by fudadmin on Thu, 13 Apr 2006 05:17:23 GMT
View Forum Message <> Reply to Message

quote from AGG...
and just some thoughts...

```
//The image buffers
        // are not displayed directly, they should be copied to or
        // combined somehow with the rbuf_window(). rbuf_window() is
        // the only buffer that can be actually displayed.
        rendering_buffer& rbuf_window()         { return m_rbuf_window; }
        rendering_buffer& rbuf_img(unsigned idx) { return m_rbuf_img[idx]; }

//=======

        void copy_img_to_window(unsigned idx)
        {
           if(idx < max_images && rbuf_img(idx).buf())
           {
              rbuf_window().copy_from(rbuf_img(idx));
           }
        }

        //------------------------------------------------------------------
        void copy_window_to_img(unsigned idx)
        {
           if(idx < max_images)
           {
              create_img(idx, rbuf_window().width(), rbuf_window().height());
              rbuf_img(idx).copy_from(rbuf_window());
           }
        }

        //------------------------------------------------------------------
        void copy_img_to_img(unsigned idx_to, unsigned idx_from)
        {
           if(idx_from < max_images &&
              idx_to < max_images &&
              rbuf_img(idx_from).buf())
           {
              create_img(idx_to,
                         rbuf_img(idx_from).width(),
                         rbuf_img(idx_from).height());
              rbuf_img(idx_to).copy_from(rbuf_img(idx_from));
           }
        }
```

If I understand correctly, one simple approach could be:
1. to use some of agg image manipulation functions even outside Ultimate's Draw on one image as
2. agg's so called "rendering buffer" and simply ...draw that image! Just a question of pixel formats...

More difficult would be to bind graphic objects with events...
and 1/3 pixel precision...

Subject: Re: AGG and Upp Draw integration...
Posted by mirek on Thu, 13 Apr 2006 12:19:41 GMT
View Forum Message <> Reply to Message

fudadmin wrote on Thu, 13 April 2006 01:17quote from AGG...
and just some thoughts...

```
//The image buffers
        // are not displayed directly, they should be copied to or
        // combined somehow with the rbuf_window(). rbuf_window() is
        // the only buffer that can be actually displayed.
        rendering_buffer& rbuf_window()         { return m_rbuf_window; }
        rendering_buffer& rbuf_img(unsigned idx) { return m_rbuf_img[idx]; }

//=======

        void copy_img_to_window(unsigned idx)
        {
           if(idx < max_images && rbuf_img(idx).buf())
           {
              rbuf_window().copy_from(rbuf_img(idx));
           }
        }

        //----------------------------------------------------------------
        void copy_window_to_img(unsigned idx)
        {
           if(idx < max_images)
           {
              create_img(idx, rbuf_window().width(), rbuf_window().height());
              rbuf_img(idx).copy_from(rbuf_window());
           }
        }

        //----------------------------------------------------------------
        void copy_img_to_img(unsigned idx_to, unsigned idx_from)
        {
```

```
        if(idx_from < max_images &&
           idx_to < max_images &&
           rbuf_img(idx_from).buf())
        {
            create_img(idx_to,
                       rbuf_img(idx_from).width(),
                       rbuf_img(idx_from).height());
            rbuf_img(idx_to).copy_from(rbuf_img(idx_from));
        }
    }
```

If I understand correctly, one simple approach could be:
1. to use some of agg image manipulation functions even outside Ultimate's Draw on one image as
2. agg's so called "rendering buffer" and simply ...draw that image! Just a question of pixel formats...

More difficult would be to bind graphic objects with events...
and 1/3 pixel precision...

Sure, in fact this is what will happen in output to the screen, but this does not solve Drawing/printing...

The scenartio that has to be supported:

Image is drawn at certain size to Drawing, then whole drawing is resized and drawn elsewhere (printer).

Therefore virtualization has to support storing drawing ops into Drawing and also printing somehow.

Mirek

Subject: Re: AGG and Upp Draw integration...
Posted by mirek on Thu, 13 Apr 2006 12:21:36 GMT
View Forum Message <> Reply to Message

P.S.: See all implementation details coming into play in reference/Reports...

Mirek

Subject: Re: AGG and Upp Draw integration...
Posted by mirek on Mon, 15 May 2006 20:33:55 GMT
View Forum Message <> Reply to Message

Ok, so "rendering buffer" == "ImageBuffer"

Anyway, not exactly hard thing to do, but lot of lines will be needed to provide "drawing recording" code... (basically, in one of modes, all drawing operations will have instead performed be stored into String to be drawn by DrawData later, after possible rescaling).

Mirek

Subject: Re: AGG and Upp Draw integration...
Posted by fudadmin on Wed, 07 Jun 2006 01:46:23 GMT
View Forum Message <> Reply to Message

I have got agg working to upp ImageBuffer. That means, I guess (I haven't tested all the examples), it is possible to use all power of agg with Ultimate++!
One interesting thing that if I use one more separate buffer (not ImageBuffer directly atached to agg::rendering_buffer) with memset and then memcpy it to ImageBuffer, animation looks significantly better (less shearing) in my eyes... Any ideas why? (I tried to slow down things with GuiSleep but that doesn't help...)

P.S. Some more thoughts...
The second thing is that if to adapt more things from upp to agg then 2/3 of agg would be possible to throw away... New Image and things are superb!
The third thing, I can't believe my eyes but it looks that it's possible to have even 1/256 pixel accuracy (at least visual) only with upp! E.g. during my experiments I've made a smooth horizontal line going from 1 pixel to 2 pixels width (256 length)...

Subject: Re: AGG and Upp Draw integration...
Posted by mirek on Wed, 07 Jun 2006 11:17:41 GMT
View Forum Message <> Reply to Message

fudadmin wrote on Tue, 06 June 2006 21:46I have got agg working to upp ImageBuffer. That means, I guess (I haven't tested all the examples), it is possible to use all power of agg with Ultimate++!

Is it possible to upload it to uppbox? (I hope you have now adapted AGG as package...) With example?

Quote:
One interesting thing that if I use one more separate buffer (not ImageBuffer directly atached to

agg::rendering_buffer) with memset and then memcpy it to ImageBuffer, animation looks significantly better (less shearing) in my eyes... Any ideas why? (I tried to slow down things with GuiSleep but that doesn't help...)

Hard to say what is happening based on such short description...

Quote:
The second thing is that if to adapt more things from upp to agg then 2/3 of agg would be possible to throw away... New Image and things are superb!
The third thing, I can't believe my eyes but it looks that it's possible to have even 1/256 pixel accuracy (at least visual) only with upp! E.g. during my experiments I've made a smooth horizontal line going from 1 pixel to 2 pixels width (256 length)...

I am a bit confused about above statements. Well, new U++ Image is good improvement, however, what is so important about AGG is vector drawing - and there is nothing new in U++ in this area (actually, some less often used functions are still unimplemented).

What do you mean by drawing line with 1/256 pixel accuracy? Have you implemented your own drawing routines that work on ImageBuffer?

Mirek

Subject: Re: AGG and Upp Draw integration...
Posted by fudadmin on Wed, 07 Jun 2006 12:22:46 GMT
View Forum Message <> Reply to Message

I'll post in a few days after I clean and make my code more elegant. Also, there are more updates for agg as of 5 May 2006. At the moment, I have made a bit of a mess of agg files...
And my working test looks like this:
(1/256 accuracy is a result of using some alpha values...)

```
//#define AGG_RGB24
//#define AGG_BGRA32
//#define AGG_RGBA32
//#define AGG_ARGB32
//#define AGG_ABGR32
//#define AGG_RGB565
//#define AGG_RGB555
//#define AGG_GRAY8
//#define AGG_BGR24

#define AGG_RGBA32

#include <agg_main/agg_main.h>
```

```
#include <nImage/Image.h>
#include <nImage/Raster.h>

// AGG rendering buffer class - pointers to each row.


//====================1/256 accuracy!=================
void DrawLine(ImageBuffer& b, Point p1, Point p2, Color color, int alpha = 255)
{
int angle=0;
int dx=p2.x-p1.x;
int dy=p2.y-p1.y;

   angle = dx/dy;

 RGBA ba;
 RGBA ba1;
 ba.a = alpha;
 ba.b = color.GetB();
 ba.r = color.GetR();
 ba.g = color.GetG();

 ba1.a=255;
 ba1.b = 0;//color.GetB();
 ba1.r = 255;//255;//color.GetR();
 ba1.g = 0;//color.GetG();

// RGBA *t = b[y] + x;
// RGBA *t = b[10];
int shading=1; //255/(2*angle);
//int steps;
 int y=21;
 for(int x=0; x<dx; x++){
//   b[y][x]=ba;
 }

 y=1;
 for(int x=0; x<dx; x++) {
     ba.a= (x%2)+x/2;
      b[y-1][x]=ba;

     ba.a =255-ba.a%2;
//     b[y][x]=ba;

//     ba.a= 255-ba.a%2;
//      b[y+1][x]=ba;
  }
```

```
 y=1;
 for(int x=0; x<dx; x++) {
     ba.a= x%2+x;
//   b[y][x]=ba;
     ba.a =255-ba.a%2;
//   b[y+1][x]=ba;

  }
 y=24;
 for(int x=0; x<dx; x++){
  ba.a=255;
//   b[y][x]=ba;
 }


//  Fill(t, t+5*i , ba);
//  Fill(t, t+5*i-1 , ba2);
 // t += b.GetSize().cx;
//  t+=b.GetSize().cx;
//  *t++=ba;
}

//Mirek's...
void DrawRect(ImageBuffer& b, int x, int y, int cx, int cy, Color color, int alpha = 255)
{
 RGBA ba;
 ba.a = alpha;
 ba.b = color.GetB();
 ba.r = color.GetR();
 ba.g = color.GetG();
 RGBA *t = b[y] + x;
 for(int i = 0; i < cy; i++) {
  Fill(t, t + cx, ba);
  t += b.GetSize().cx;
 }
}


//====================================
//====================================
//====================================
class App : public TopWindow {
private:
 SliderCtrl slider;
 double m_x[3];
 double m_y[3];
 double m_dx;
 double m_dy;
```

```
 int    m_idx;

 agg::int8u *tbuf;

 agg::rasterizer_scanline_aa<> m_ras;
 agg::scanline_p8            m_sl_p8;
 agg::scanline_bin           m_sl_bin;


 int frame_width;
 int frame_height;

 double m_gamma_value;

public:
 virtual void Paint(Draw& draw);
 void DrawAgg();
 void test();
 typedef App CLASSNAME;

 typedef agg::renderer_base<pixfmt> renderer_base;
 typedef agg::renderer_scanline_aa_solid<renderer_base> renderer_aa;
 typedef agg::renderer_scanline_bin_solid<renderer_base> renderer_bin;

 void App::SetGamma();


 App();

 ~App() {
// ; delete [] m_points;
 delete [] tbuf;
 }

};

void App::DrawAgg()
{
 ImageBuffer rgbbuf(frame_width, frame_height);
 RGBA *rgba = rgbbuf[0];

 agg::rendering_buffer rbuf(tbuf, frame_width, frame_height, frame_width*4);

 pixfmt pixf(rbuf);
 renderer_base rb(pixf);

 renderer_aa ren_aa(rb);
```

```
  agg::path_storage path;

  path.move_to(m_x[0], m_y[0]);
  path.line_to(m_x[1], m_y[1]);
  path.line_to(m_x[2], m_y[2]);
  path.close_polygon();

  ren_aa.color(agg::rgba(0.7, 0.5, 0.1, 0.7));

  m_ras.gamma(agg::gamma_power( m_gamma_value* 2.0));
  m_ras.add_path(path);
  agg::render_scanlines(m_ras, m_sl_p8, ren_aa);

 memsetex(rgbbuf, &tbuf, sizeof(RGBA), 1000);
// buf= rbuf.buf();
// memcpy(buf, rbuf.buf(), 10*3*6);
}



void App::Paint(Draw& w)
{
 Size sz = GetSize();
 w.DrawRect(GetSize(), SLtGray);

 ImageBuffer b(256, 256);  //make space (in buffer)
// DrawRect(b, 0, 0, 100, 100, SBlue);  //drawToBuffer
// DrawRect(b, 0, 0, 20, 20, Red);
 DrawRect(b, 0, 0, 25, 255, Red);
 DrawLine(b, Point(0,0), Point(255,1), Black);

 Image img = b;
// img.Paint(w, 0, 0);  //image has the buffer! paint it (in fact both!) to window
//=====================================================
 ImageBuffer rgbbuf(frame_width, frame_height);
// RGBA *rgba = rgbbuf();

 memset(rgbbuf, 255, frame_width * frame_height * 4);

// DrawRect(rgbbuf, 0, 0, frame_width, frame_height, SWhite);
 unsigned char * buf= (unsigned char *)rgbbuf[0];

 agg::rendering_buffer rbuf(buf, frame_width, frame_height, frame_width*4); //veikia!!!
// agg::rendering_buffer rbuf(tbuf, frame_width, frame_height, frame_width*4); //veikia!!!

// agg::rendering_buffer rbuf; //  (???, frame_width, frame_height, frame_width*4);
```

```
 pixfmt pixf(rbuf);
 renderer_base rb(pixf);

 renderer_aa ren_aa(rb);

  agg::path_storage path;

  path.move_to(m_x[0], m_y[0]);
  path.line_to(m_x[1], m_y[1]);
  path.line_to(m_x[2], m_y[2]);
  path.close_polygon();

  ren_aa.color(agg::rgba(0.7, 0.5, 0.1, 0.7));

  m_ras.gamma(agg::gamma_power( m_gamma_value*2.0));
  m_ras.add_path(path);
  agg::render_scanlines(m_ras, m_sl_p8, ren_aa);

// memsetex(rgbbuf, &tbuf, sizeof(RGBA), 0); //possible to use for moving?
// memcpy(rgbbuf, rbuf.buf(), frame_width*frame_height*4);
// memcpy(rgbbuf, tbuf, frame_width*frame_height*4);

 Image img1 = rgbbuf;
 img1.Paint(w, 50, 50);
 GuiSleep(500);
// img.Paint(w, 100, 50);

// memset(tbuf, 255, frame_width * frame_height * 4);

// img.Paint(w, 250, 70, 100);
// img.Paint(w, 300, 250, 100);
//new buffer
// ImageBuffer b1(60, 60);

//// DrawAgg();

// img.Paint(w, 300, 300);

}


void App::SetGamma()
{
 m_gamma_value=(double)~slider/100;
 int delta = ~slider;
  m_x[0] = 100 + 120-delta; m_y[0] = 60-delta;
  m_x[1] = 369 + 120-delta; m_y[1] = 170-delta;
  m_x[2] = 143 + 120-delta; m_y[2] = 310-delta;
```

```
// memset(tbuf, 255, frame_width * frame_height * 4);
 Refresh();
}


App::App()
{
 frame_width=500;
 frame_height=400;
 slider.SetRect(600,300,20,100);
 Add(slider);
 slider.MinMax(0,100);
 slider.SetData(50);
 m_gamma_value=1.0;

 slider.WhenAction=THISBACK(SetGamma);
// ImageBuffer b(100, 100);
// DrawLine(b, Point(0,0), Point(20,20), Green);
  m_x[0] = 100 + 120; m_y[0] = 60;
  m_x[1] = 369 + 120; m_y[1] = 170;
  m_x[2] = 143 + 120; m_y[2] = 310;

 tbuf = new agg::int8u[frame_width*frame_height*4];

 memset(tbuf, 255, frame_width * frame_height * 4);

// DrawAgg();
  BackPaint();
}


GUI_APP_MAIN
{
// the_application     app(pix_format, flip_y, num_points);

// app.init(start_width, start_height, agg::window_resize | agg::window_keep_aspect_ratio)

 App().Title("AGG Drawing with U++...").Run();

}

/*
Ok, so "rendering buffer" == "ImageBuffer" Wink

Anyway, not exactly hard thing to do, but lot of lines will be needed to provide "drawing recording"
code...
(basically, in one of modes,
```

all drawing operations will have instead performed:
be stored into String
to be drawn by DrawData later, after possible rescaling).

Mirek

```
//from agg line:
 unsigned char* buffer = new unsigned char[frame_width * frame_height * 3];

 memset(buffer, 255, frame_width * frame_height * 3);

 agg::rendering_buffer rbuf(buffer,
       frame_width,
       frame_height,
       frame_width * 3);

 unsigned i;
 for(i = 0; i < rbuf.height()/2; ++i)
 {
  // Get the pointer to the beginning of the i-th row (Y-coordinate)
  // and shift it to the i-th position, that is, X-coordinate.
  //---------------
  unsigned char* ptr = rbuf.row(i) + i * 3;

  // PutPixel, very sophisticated, huh? :)
  //-------------
  *ptr++ = 127; // R
  *ptr++ = 200; // G
  *ptr++ = 98;  // B
 }


*/
```

Subject: Re: AGG and Upp Draw integration...
Posted by forlano on Wed, 07 Jun 2006 17:28:35 GMT
View Forum Message <> Reply to Message

fudadmin wrote on Wed, 07 June 2006 03:46I have got agg working to upp ImageBuffer. That means, I guess (I haven't tested all the examples), it is possible to use all power of agg with Ultimate++!
One interesting thing that if I use one more separate buffer (not ImageBuffer directly atached to agg::rendering_buffer) with memset and then memcpy it to ImageBuffer, animation looks significantly better (less shearing) in my eyes... Any ideas why? (I tried to slow down things with GuiSleep but that doesn't help...)

P.S. Some more thoughts...
The second thing is that if to adapt more things from upp to agg then 2/3 of agg would be possible to throw away... New Image and things are superb!
The third thing, I can't believe my eyes but it looks that it's possible to have even 1/256 pixel accuracy (at least visual) only with upp! E.g. during my experiments I've made a smooth horizontal line going from 1 pixel to 2 pixels width (256 length)...

I've not really understood what it is happening with the new upp Draw routine. But after your post I was on the AGG site and remained impressed by its capability and quality. So if AGG is being to be integrated in upp++ I think a new era will begin. Please post as soon as possibile your new achievement

Luigi

Subject: Re: AGG and Upp Draw integration...
Posted by fudadmin on Fri, 09 Jun 2006 21:17:37 GMT
View Forum Message <> Reply to Message

fudadmin wrote on Wed, 07 June 2006 02:46
P.S. Some more thoughts...
The second thing is that if to adapt more things from upp to agg then 2/3 of agg would be possible to throw away... New Image and things are superb!

Or... the other way round - to throw away some things from upp. Because quite a lot of things duplicate each other, IMO.

Subject: Re: AGG and Upp Draw integration...
Posted by fudadmin on Sat, 10 Jun 2006 01:17:28 GMT
View Forum Message <> Reply to Message

forlano wrote on Wed, 07 June 2006 18:28

I've not really understood what it is happening with the new upp Draw routine. But after your post I was on the AGG site and remained impressed by its capability and quality. So if AGG is being to be integrated in upp++ I think a new era will begin. Please post as soon as possibile your new achievement

Luigi

if you want "really" to be impressed download and try the things from here:
http://www.arilect.com/upp/forum/index.php?t=msg&th=296& amp;start=0&

(I'm surprised you haven't tried)

Subject: Re: AGG and Upp Draw integration...
Posted by mirek on Sun, 11 Jun 2006 19:26:18 GMT
View Forum Message <> Reply to Message

fudadmin wrote on Fri, 09 June 2006 17:17fudadmin wrote on Wed, 07 June 2006 02:46
P.S. Some more thoughts...
The second thing is that if to adapt more things from upp to agg then 2/3 of agg would be possible to throw away... New Image and things are superb!

Or... the other way round - to throw away some things from upp. Because quite a lot of things duplicate each other, IMO.

Hm, like what?

MIrek

Subject: Re: AGG and Upp Draw integration...
Posted by fudadmin on Sun, 11 Jun 2006 22:21:24 GMT
View Forum Message <> Reply to Message

Maybe that duplication is really needed for upp Ctlrs, I don't know. But isn't upp Draw and related classes just a low quality agg (in terms of graphics not programming)(like bin rasterizers)? Rectangle, size, clipping management... resizing, blending... Then DrawLine, DrawRectangle, DrawPolygon and similar.
But... don't pay too much attention to this...

Subject: Re: AGG and Upp Draw integration...
Posted by mirek on Mon, 12 Jun 2006 00:01:51 GMT
View Forum Message <> Reply to Message

fudadmin wrote on Sun, 11 June 2006 18:21Maybe that duplication is really needed for upp Ctlrs, I don't know. But isn't upp Draw and related classes just a low quality agg (in terms of graphics not programming)(like bin rasterizers)? Rectangle, size, clipping management... resizing, blending... Then DrawLine, DrawRectangle, DrawPolygon and similar.
But... don't pay too much attention to this...

Keep in mind one important thing - these operations are hardware accelerated and more importantly, are implemented by printer drivers.

Using AGG for printing as the only possibility would result in rendering each page as bitmap - 50MB / page.

Mirek

Subject: Re: AGG and Upp Draw integration...
Posted by fudadmin on Mon, 12 Jun 2006 01:22:38 GMT
View Forum Message <> Reply to Message

luzr wrote on Mon, 12 June 2006 01:01fudadmin wrote on Sun, 11 June 2006 18:21Maybe that duplication is really needed for upp Ctlrs, I don't know. But isn't upp Draw and related classes just a low quality agg (in terms of graphics not programming)(like bin rasterizers)? Rectangle, size, clipping management... resizing, blending... Then DrawLine, DrawRectangle, DrawPolygon and similar.
But... don't pay too much attention to this...

Keep in mind one important thing - these operations are hardware accelerated and more importantly, are implemented by printer drivers.

Using AGG for printing as the only possibility would result in rendering each page as bitmap - 50MB / page.

Mirek

Yes, I've forgotten that. It's time for hardware producers to support agg... .

Subject: Re: AGG and Upp Draw integration...
Posted by fudadmin on Mon, 12 Jun 2006 02:05:44 GMT
View Forum Message <> Reply to Message

Quote:
Sure, in fact this is what will happen in output to the screen, but this does not solve Drawing/printing...

The scenartio that has to be supported:

Image is drawn at certain size to Drawing, then whole drawing is resized and drawn elsewhere (printer).

Therefore virtualization has to support storing drawing ops into Drawing and also printing somehow.

I have my aggImg working with DrawingDraw and then it gets resized nicely with Reports.
How do you imagine the whole interface?
Something like w.DrawAggLine or w.DrawLine(....., "agg-line")?

Subject: Re: AGG and Upp Draw integration...
Posted by mirek on Mon, 12 Jun 2006 05:47:39 GMT
View Forum Message <> Reply to Message

I believe that AGG (and other SW renderers) should be supported via DrawData interface.

Something like:

....Draw& w;....

AGGDraw agg(w, x, y, cx, cy);
agg.DrawLine(...
...

There are two alternatives (to investigate):

- AGGDraw simply stores all drawing operations to String and its destructor performs DrawData (which paints the content using registred AGG routine)

- slightly more efficient approach would detect whether target is Drawing and performed storing just in that case; otherwise it would create Image and draw content

I would start with the first one. I believe that costs to record drawing first and replay later are negligible compared to actual drawing operations.

BTW, note that DrawData performs "RLE encoding" of resulting bitmap drawing which I believe should greatly reduce printer bandwidth in most common cases.

Mirek

P.S.: Please consider this as preliminary and do not be angry if I change mind completely. This is too important to accept very first version, we need some experimenting before final decision...

Subject: Re: AGG and Upp Draw integration...
Posted by fudadmin on Tue, 13 Jun 2006 11:48:29 GMT
View Forum Message <> Reply to Message

fudadmin wrote on Mon, 12 June 2006 02:22Quote:
Using AGG for printing as the only possibility would result in rendering each page as bitmap - 50MB / page.

Mirek

Yes, I've forgotten that. It's time for hardware producers to support agg... .

there it goes:
http://www.khronos.org/openvg/

Subject: Re: AGG and Upp Draw integration...
Posted by fudadmin on Tue, 13 Jun 2006 19:07:13 GMT
View Forum Message <> Reply to Message

luzr wrote on Mon, 12 June 2006 06:47I believe that AGG (and other SW renderers) should be supported via DrawData interface.

...

I believe that costs to record drawing first and replay later are negligible compared to actual drawing operations.

BTW, note that DrawData performs "RLE encoding" of resulting bitmap drawing which I believe should greatly reduce printer bandwidth in most common cases.

Mirek

After that topic: http://www.arilect.com/upp/forum/index.php?t=msg&&th
=1098&goto=3689#msg_3689
I agree...

Do I understandand correctly that the line in DrawDataOp:

Image m = dd->Render(ccy); //render some lines (rectangle of image)

implies on having not the whole aggImg but only some piece of it generated (or lines rendered) "on demand"?

Subject: Re: AGG and Upp Draw integration...
Posted by mirek on Wed, 14 Jun 2006 18:52:33 GMT
View Forum Message <> Reply to Message

fudadmin wrote on Tue, 13 June 2006 15:07luzr wrote on Mon, 12 June 2006 06:47I believe that AGG (and other SW renderers) should be supported via DrawData interface.

...

I believe that costs to record drawing first and replay later are negligible compared to actual

drawing operations.

BTW, note that DrawData performs "RLE encoding" of resulting bitmap drawing which I believe should greatly reduce printer bandwidth in most common cases.

Mirek

After that topic: http://www.arilect.com/upp/forum/index.php?t=msg&&th=1098&goto=3689#msg_3689
I agree...

Do I understandand correctly that the line in DrawDataOp:

```
Image m = dd->Render(ccy);  //render some lines (rectangle of image)
```

implies on having not the whole aggImg but only some piece of it generated (or lines rendered) "on demand"?

Yes, it creates the "band" of the target width, but ccy height. DataDrawer should return consequentive (it that the right word? bands, starting with y = 0.

BTW, term "band" is adapted from Windows printing terminology and I believe that the very original meaning comes from those old matrix printers - when printing graphics, you need to send similars strips to the printer, covering the height of ink band.

Mirek

Subject: Re: AGG and Upp Draw integration...
Posted by forlano on Sun, 18 Jun 2006 12:03:49 GMT
View Forum Message <> Reply to Message

fudadmin wrote on Sat, 10 June 2006 03:17
if you want "really" to be impressed download and try the things from here:
http://www.arilect.com/upp/forum/index.php?t=msg&th=296& amp; amp;start=0&
(I'm surprised you haven't tried)

Today I've downloaded once more the agg port. I was able to compile it with MSC8. It's very nice.
So you was able to port all the test cases of the agg library. Congratulations.
I had a look to the source code and seems to me not at all immediate. In fact in the past I was used only to hilight 1 pixel at time in some hires device.
I hope that the integration of agg with upp will produce some easy_to_use interface.

Luigi

Subject: Re: AGG and Upp Draw integration...
Posted by mirek on Mon, 16 Oct 2006 15:54:49 GMT
View Forum Message <> Reply to Message

Any news on AGG front? (I guess chances to have it for next major release are dim now).

Mirek

Subject: Re: AGG and Upp Draw integration...
Posted by Novo on Thu, 04 Jan 2007 04:33:38 GMT
View Forum Message <> Reply to Message

AGG has changed license to the GNU GPL license ...

Does that affect Upp in some way?

Subject: Re: AGG and Upp Draw integration...
Posted by mirek on Thu, 04 Jan 2007 14:24:46 GMT
View Forum Message <> Reply to Message

Really?

Hm, what if I have older version?

OTOH, Tom wanted to have fun with rendering polygons...

Mirek

Subject: Re: AGG and Upp Draw integration...
Posted by Novo on Thu, 04 Jan 2007 15:28:18 GMT
View Forum Message <> Reply to Message

Check this yourself. Look at the bottom-left corner ...

http://www.antigrain.com/

Version 2.4 keeps a previous license. New fixes will be done only to version 2.5 and above.