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
 }


*/
```