**用AGG实现高质量图形输出（一）**

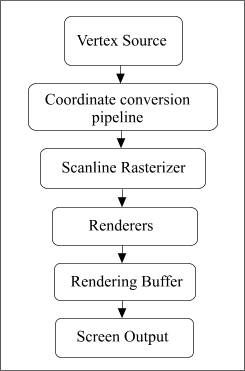
AGG是一个开源、高效的跨平台2D图形库。AGG的功能与GDI+的功能非常类似，但提供了比GDI+更灵活的编程接口，其产生的图形的质量也非常高(自称超过GDI+)



**使用前AGG的准备工作**

1. 下载AGG库，它的家在<http://www.antigrain.com>，目前最高版本是AGG2.5
2. 解压，后面以[AGG]表示AGG的解压目录.
3. 把[AGG]\include加入到include搜索目录中
4. 把[AGG]\src里所有cpp加入到项目中（或者用makefile一起编译）
5. 另外，AGG还有一些其它组件，用到时也要把它们(都是些.h和.cpp文件)加入项目:
   * 如果要用AGG的控件和窗体，要加入[AGG]\src\ctrl\\*.cpp和[AGG]\src\platform\<OS>\\*.cpp，头文件在[AGG]\include\ctrl和[AGG]\include\platform里
   * 如果要用到TrueType字体显示，要加入[AGG]\font\_win32\_tt目录下的源码和头文件。利用freetype库，则是[AGG]\font\_freetype目录。
   * 如果要用到Generic Polygon Clipper库（一个区域剪裁计算库），加入[AGG]\gpc目录下的源码和头文件。

**AGG图形显示原理见下图：**



其中：

1. **Vertex Source** 顶点源，里面存放了一堆2D顶点以及对应的命令，如"MoveTo"、"LineTo"等。
2. **Coordinate conversion pipeline** 坐标转换管道，它可以变换Vertex Source中的顶点，比如矩阵变换，轮廓提取，转换为虚线等。
3. **Scanline Rasterizer** 把顶点数据（矢量数据）转换成一组水平扫描线，扫描线由一组线段(Span)组成，线段(Span)包含了起始位置、长度和覆盖率（可以理解为透明度）信息。AGG的抗锯齿（Anti-Aliasing）功能也是在这时引入的。
4. **Renderers** 渲染器，渲染扫描线(Scanline)中的线段(Span)，最简单的就是为Span提供单一颜色，复杂的有多种颜色(如渐变)、使用图像数据、Pattern等。
5. **Rendering Buffer** 用于存放像素点阵数据的内存块，这里是最终形成的图像数据。

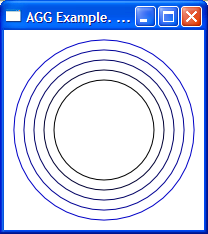
**要理解AGG的工作原理，先看一段代码：**

1. #include "agg\_basics.h"
2. #include "agg\_rendering\_buffer.h"
3. #include "agg\_rasterizer\_scanline\_aa.h"
4. #include "agg\_scanline\_u.h"
5. #include "agg\_renderer\_scanline.h"
6. #include "agg\_pixfmt\_rgb.h"
7. #include "platform/agg\_platform\_support.h"
8. #include "agg\_ellipse.h"
9. #include "agg\_conv\_contour.h"
10. #include "agg\_conv\_stroke.h"
12. class the\_application : public agg::platform\_support
13. {
14. public:
15. the\_application(agg::pix\_format\_e format, bool flip\_y) :
16. agg::platform\_support(format, flip\_y)
17. {
18. }
20. virtual void on\_draw()
21. {
22. //Rendering Buffer
23. agg::rendering\_buffer &rbuf = rbuf\_window();
24. agg::pixfmt\_bgr24 pixf(rbuf);
26. // Renderers
27. typedef agg::renderer\_base<agg::pixfmt\_bgr24> renderer\_base\_type;
28. renderer\_base\_type renb(pixf);
30. typedef agg::renderer\_scanline\_aa\_solid<renderer\_base\_type> renderer\_scanline\_type;
31. renderer\_scanline\_type rensl(renb);
33. // Vertex Source
34. agg::ellipse ell(100,100,50,50);
36. // Coordinate conversion pipeline
37. typedef agg::conv\_contour<agg::ellipse> ell\_cc\_type;
38. ell\_cc\_type ccell(ell);
40. typedef agg::conv\_stroke<ell\_cc\_type> ell\_cc\_cs\_type;
41. ell\_cc\_cs\_type csccell(ccell);
43. // Scanline Rasterizer
44. agg::rasterizer\_scanline\_aa<> ras;
45. agg::scanline\_u8 sl;
47. // Draw
48. renb.clear(agg::rgba8(255,255,255));
49. for(int i=0; i<5; i++)
50. {
51. ccell.width(i\*20);
52. ras.add\_path(csccell);
53. rensl.color( agg::rgba8(0,0,i\*50));
54. agg::render\_scanlines(ras,sl,rensl);
55. }
56. }
57. };
59. int agg\_main(int argc, char\* argv[])
60. {
61. the\_application app(agg::pix\_format\_bgr24, false);
62. app.caption("AGG Example. Anti-Aliasing Demo");
64. if(app.init(600, 400, agg::window\_resize))
65. {
66. return app.run();
67. }
68. return -1;
69. }

编译这段代码的方法是（以VC为例）：

1. 新建空白GUI项目（就是有WinMain的项目）
2. 把[AGG]\src里所有\*.cpp加入到项目中
3. 把[AGG]\src\platform\Win32\\*.cpp加入到项目中
4. Ctrl+C/Ctrl+V 上面的代码
5. 编译！

显示效果：



我们先不管agg\_main及agg::platform\_support的问题，实际上agg::platform\_support只是AGG给我们方便显示AGG图形用的，真正应用时几乎不会用到（后面会讲到怎样把AGG图形画到HDC上）。

现在我们只需要知道这个框架可以生成一个窗体，当窗体重画时会调用virtual void on\_draw()就行了。

现在直接从on\_draw()开始看

1. 通过rbuf\_window()方法得到一个agg::rendering\_buffer，它就是“**Rendering Buffer**”，是一块用于存放图像的内存块。通过pixfmt\_bgr24包装，我们就可以以像素为单位存取图像。
2. agg::renderer\_base和agg::renderer\_scanline\_aa\_solid都属于"**渲染器Renderer**"。renderer\_base为底层渲染器，它支撑起所有的高层渲染器。这里的renderer\_scanline\_aa\_solid就是一个高层渲染器。
3. agg::ellipse是“**顶点源Vertex Source**”，这个顶点源呈现的是一个圆形。
4. agg::conv\_contour和agg::conv\_stroke作为“**坐标转换管道Coordinate conversion pipeline**”，conv\_contour扩展轮廓线，conv\_stroke只显示轮廓线（如果没有conv\_stroke就会显示实心圆，可以去掉试试）。
5. agg::rasterizer\_scanline\_aa<>就是“**Scanline Rasterizer**”啦。
6. agg::render\_scanlines函数执行这个AGG工作流程。

《[待续](http://www.cppprog.com/2009/0821/150.html)》

**用AGG实现高质量图形输出（二）**

本文上接《用AGG实现高质量图形输出（一）》，分别介绍了AGG显示流程中的各个环节。

[上次](http://www.cppprog.com/2009/0816/146.html)讲了AGG的显示原理并举了一个简单的例子，这一篇文章开始讲AGG工作流程里的每个环节。为了方便对照，再放一次AGG显示流程 图

