

原 ffdshow 源代码分析 4：位图覆盖滤镜（滤镜部分Filter）

2013年10月25日 00:15:42 阅读数：5026

=====

ffdshow源代码分析系列文章列表：

[ffdshow 源代码分析 1：整体结构](#)

[ffdshow 源代码分析 2：位图覆盖滤镜（对话框部分Dialog）](#)

[ffdshow 源代码分析 3：位图覆盖滤镜（设置部分Settings）](#)

[ffdshow 源代码分析 4：位图覆盖滤镜（滤镜部分Filter）](#)

[ffdshow 源代码分析 5：位图覆盖滤镜（总结）](#)

[ffdshow 源代码分析 6：对解码器的dll的封装（libavcodec）](#)

[ffdshow 源代码分析 7：libavcodec视频解码器类（TvideoCodecLibavcodec）](#)

[ffdshow 源代码分析 8：视频解码器类（TvideoCodecDec）](#)

[ffdshow 源代码分析 9：编解码器有关类的总结](#)

=====



第一篇文章介绍了ffdshow的位图覆盖滤镜的对话框（Dialog）部分：[ffdshow 源代码分析2：位图覆盖滤镜（对话框部分Dialog）](#)

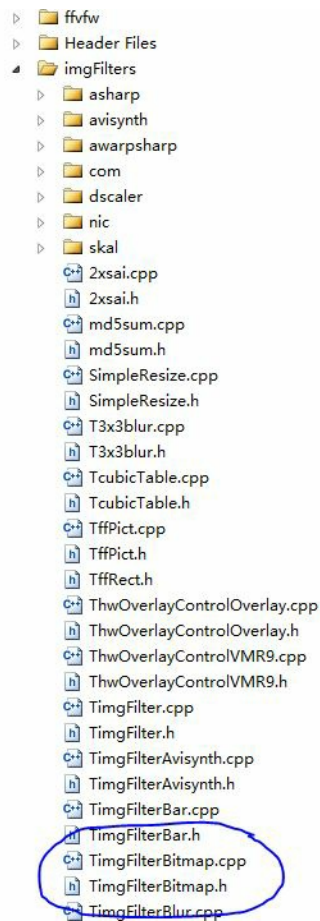
第二篇文章介绍了ffdshow的位图覆盖滤镜的设置（Settings）部分：[ffdshow 源代码分析 3：位图覆盖滤镜（设置部分Settings）](#)

此外还有一个滤镜部分（Filter）。这三个部分就可以组成一个ffdshow的滤镜功能了。本文就来简介一下ffdshow的滤镜部分。

滤镜部分（Filter）

ffdshow的滤镜的滤镜部分(怎么感觉名字有点重复 = =,算了先这么叫吧)的功能主要用于完成具体的图像处理功能。具体到位图覆盖滤镜的话，就是用于把图片覆盖到视频上面，他是ffdshow滤镜的核心。

与位图覆盖（Bitmap）滤镜的滤镜处理有关的类位于imgFilters目录下的TimFilterBitmap.h和TimFilterBitmap.cpp文件中。



先来看看TimgFilterBitmap.h

这里要注意一下，该类的名字叫TimgFilterBitmap。它的声明方式确实比较奇怪：DECLARE_FILTER(TimgFilterBitmap, public, TimgFilter)

可以看出DECLARE_FILTER是一个宏，具体这个宏的内部代码就先不查看了，否则会感觉很混乱，暂且留下一个小小的谜团。在这里只要知道这是声明了一个滤镜类就可以了。

其实TimgFilterBitmap的核心函数不多，就一个，那就是process()，具体的处理功能都是在这个函数里面实现的。

```

1.  /*
2.  *雷霄骅
3.  *leixiaohua1020@126.com
4.  *中国传媒大学/数字电视技术
5.  */
6.
7.  #ifndef TIMGFILTERBITMAP_H_
8.  #define TIMGFILTERBITMAP_H_
9.  //叠加一张位图
10. #include "TimgFilter.h"
11. #include "Tfont.h"
12.
13. struct TffPict;
14. struct TbitmapSettings;
15. //特别的声明方式 ==
16. DECLARE_FILTER(TimgFilterBitmap, public, TimgFilter)
17. private:
18. //图像
19. TffPict *bitmap;
20. //内存
21. Tbuffer bitmapbuf;
22. char_t oldflnm[MAX_PATH];
23. typedef void (*Tblendplane)(const TcspInfo &cspInfo, const unsigned int dx[3], const unsigned int dy[3], unsigned char *dst[3], const
t stride_t dststride[3], const unsigned char *src[3], const stride_t srcstride[3], int strength, int invstrength);
24. //注意 这个类有一个实例, 名字叫w
25. class TrenderedSubtitleLineBitmap : public TrenderedSubtitleWordBase
26. {
27. public:
28. TrenderedSubtitleLineBitmap(void): TrenderedSubtitleWordBase(false) {}
29. TffPict *pict;
30. const TbitmapSettings *cfg;
31. //叠加
32. Tblendplane blend;
33. //打印
34. virtual void print(int startx, int starty /* not used */, unsigned int dx[3], int dy1[3], unsigned char *dstLn[3], const stride_
t stride[3], const unsigned char *bmp[3], const unsigned char *msk[3], REFERENCE_TIME rtStart = REFTIME_INVALID) const;
35. } w;
36. TrenderedSubtitleLine l;
37. //是TrenderedSubtitleLine的一个vector
38. TrenderedSubtitleLines ls;
39. int oldmode;
40. //几种叠加方式
41. template<class _mm> static void blend(const TcspInfo &cspInfo, const unsigned int dx[3], const unsigned int dy[3], unsigned char *ds
t[3], const stride_t dststride[3], const unsigned char *src[3], const stride_t srcstride[3], int strength, int invstrength);
42. template<class _mm> static void add(const TcspInfo &cspInfo, const unsigned int dx[3], const unsigned int dy[3], unsigned char *dst[
3], const stride_t dststride[3], const unsigned char *src[3], const stride_t srcstride[3], int strength, int invstrength);
43. template<class _mm> static void darken(const TcspInfo &cspInfo, const unsigned int dx[3], const unsigned int dy[3], unsigned char *d
st[3], const stride_t dststride[3], const unsigned char *src[3], const stride_t srcstride[3], int strength, int invstrength);
44. template<class _mm> static void lighten(const TcspInfo &cspInfo, const unsigned int dx[3], const unsigned int dy[3], unsigned char *
dst[3], const stride_t dststride[3], const unsigned char *src[3], const stride_t srcstride[3], int strength, int invstrength);
45. template<class _mm> static void softlight(const TcspInfo &cspInfo, const unsigned int dx[3], const unsigned int dy[3], unsigned char
 *dst[3], const stride_t dststride[3], const unsigned char *src[3], const stride_t srcstride[3], int strength, int invstrength);
46. template<class _mm> static void exclusion(const TcspInfo &cspInfo, const unsigned int dx[3], const unsigned int dy[3], unsigned char
 *dst[3], const stride_t dststride[3], const unsigned char *src[3], const stride_t srcstride[3], int strength, int invstrength);
47. //获取叠加方式
48. template<class _mm> static Tblendplane getBlend(int mode);
49. protected:
50. virtual bool is(const TffPictBase &pict, const TfilterSettingsVideo *cfg);
51. virtual uint64_t getSupportedInputColorspaces(const TfilterSettingsVideo *cfg) const
52. {
53. return FF_CSPPS_MASK_YUV_PLANAR;
54. }
55. public:
56. TimgFilterBitmap(IffdshowBase *Ideci, Tfilters *IpARENT);
57. virtual ~TimgFilterBitmap();
58. //核心函数 (Filter配置信息队列, 图像, 配置信息)
59. virtual HRESULT process(TfilterQueue::iterator it, TffPict &pict, const TfilterSettingsVideo *cfg0);
60. };
61.
62. #endif

```

再看看TimgFilterBitmap.cpp

这个文件本身代码量是比较大的, 只是其他部分我都还没有仔细分析, 确实没那没多时间。。。在这里仅简要分析一下最核心的函数process()。正是这个函数真正实现了滤镜的功能。在这个位图叠加滤镜中, process()实现了位图在视频上面的叠加功能。

```

1. //核心函数 (Filter配置信息队列, 图像, 配置信息)
2. HRESULT TimgFilterBitmap::process(TfilterQueue::iterator it, TffPict &pict, const TfilterSettingsVideo *cfg0)
3. {
4.     //都有这一句= =
5.     if (is(pict, cfg0)) {
6.         //Bitmap的配置信息
7.         const TbitmapSettings *cfg = (const TbitmapSettings*)cfg0;
8.         init(pict, cfg->full, cfg->half);
9.         unsigned char *dst[4];
10.        bool cspChanged = getCurNext(FF_CSPS_MASK_YUV_PLANAR, pict, cfg->full, COPYMODE_DEF, dst);
11.        //处理
12.        if (!bitmap || cspChanged || strcmp(oldflnm, cfg->flnm) != 0) {
13.            ff_strncpy(oldflnm, cfg->flnm, countof(oldflnm));
14.            if (bitmap) {
15.                delete bitmap;
16.            }
17.            //新建一张图
18.            //通过cfg->flnm路径
19.            //载入bitmapbuf
20.            bitmap = new TffPict(csp2, cfg->flnm, bitmapbuf, deci);
21.            //3个颜色分量?
22.            for (int i = 0; i < 3; i++) {
23.                w.dx[i] = bitmap->rectFull.dx >> bitmap->cspInfo.shiftX[i];
24.                w.dy[i] = bitmap->rectFull.dy >> bitmap->cspInfo.shiftY[i];
25.                w.bmp[i] = bitmap->data[i];
26.                w.bmpmskstride[i] = bitmap->stride[i];
27.            }
28.            w.dxChar = w.dx[0];
29.            w.dyChar = w.dy[0];
30.        }
31.
32.        if (bitmap->rectFull.dx != 0) {
33.            if (oldmode != cfg->mode)
34.                if (Tconfig::cpu_flags & FF_CPU_SSE2) {
35.                    //获取叠加方式 (SSE2)
36.                    //在cfg的mode里
37.                    w.blend = getBlend<Tsse2>(oldmode = cfg->mode);
38.                } else {
39.                    //获取叠加方式 (MMX)
40.                    w.blend = getBlend<Tmmx>(oldmode = cfg->mode);
41.                }
42.            //输出到屏幕上的设置
43.            TprintPrefs prefs(deci, NULL);
44.            //各种参数
45.            prefs.dx = dx2[0];
46.            prefs.dy = dy2[0];
47.            prefs.xpos = cfg->posx;
48.            prefs.ypos = cfg->posy;
49.            //模式不同的话
50.            if (cfg->posmode == 1) {
51.                prefs.xpos *= -1;
52.                prefs.ypos *= -1;
53.            }
54.            prefs.align = cfg->align;
55.            prefs.linespacing = 100;
56.            prefs.csp = pict.csp;
57.            w.pict = &pict;
58.            w.cfg = cfg;
59.            //打印, 需要用到TprintPrefs
60.            ls.print(prefs, dst, stride2);
61.        }
62.    }
63.    //最后都是这一句?
64.    return parent->processSample(++it, pict);
65. }

```

版权声明：本文为博主原创文章，未经博主允许不得转载。 <https://blog.csdn.net/leixiaohua1020/article/details/13006213>

文章标签： [ffdshow](#) [滤镜](#) [位图叠加](#) [directshow](#) [源代码](#)

个人分类： [ffdshow](#)

所属专栏： [开源多媒体项目源代码分析](#)

此PDF由spygg生成, 请尊重原作者版权!!!

我的邮箱: liushidc@163.com