yuv显卡转换rgb

首先要感谢ffmpeg, 如果没有它，所有做电脑视频----包括PC, 嵌入式, DV/DC,DVD机等公司(也包括我们公司）, 一大半得关门。没有它，一些中小公司将无法研发这些编解码器, 没有了这些技术基础，产品将无从谈起；没有它， PC上常见的Mplayer, KMPlayer, 暴风影音等都不会存在！所以，在很多情况下，做视频软件，ffmpeg是软件的底层库，是基础平台。

        其次要感谢显卡的超强功能。显卡越来越强大，有些显卡的GPU能力甚至要超过了PC上CPU的计算能力-----是不是主次颠倒了？正是有了这些显卡，逼真高效的游戏才成为了可能。

       现在我要说的，就是ffmpeg的解码后，多路高效显示的一点点小技巧。

       微软推出DirectShow时，可以说是天生为流媒体开发而制作的, 因为它提供的功能太强大了：多种格式视频显示、视频音频同步、视频合成、视频分离等等令人激动的功能。如果是单路或几路视频显示，当然用DirectShow是最好的选择，但是，如果要显示的视频路非常多，例如25路，使用它就会发现资源占用率极高，一路视频显示，不算解码，要4个线程！并且图像合成时CPU占有率极高----总而言之，DirectShow不适合多路视频的专业监控。

       DirectDraw是我发现的在Windows平台下最佳的解决方案，唯一的缺点就是，你需要做一些视频图像的处理，这需要更强的专业知识和更多的开发时间。事实上，从某种程度上来说，你就是在开发一款mini型的DirectShow COM.

      不必多说，转入正文。

       ffmpeg解码出来后，一般会生成YV12格式，在2005年以后出产的显卡，它可以直接放到显存中直接显示的---当然，这并不是绝对的，有些显卡，例如明基一些笔记本就不支持YV12。这种做法显然是最高效的，中间没有转换格式，数据量也是最小的。可是，有时我们需要对视频做一些特殊处理，例如，在视频上放一些文字，显示一些时间等，这种情况下，因为在DirectX提供的YUV表面上是无法得到HDC句柄的，如果直接操作YUV数据, 那非常的麻烦－－你自已要完成提供画线，字体合成之类工作，也就是说，你不能使用Win32 API, 要自已写类似的API Function. 实际上有个很简单的办法，那就是利用显卡自已的格式转换功能！

      显卡一般支持YUV格式直接转到RGB24/RGB32。至于显卡支持具体的格式，请用DirectX Caps来查询就知道了。要实现上述功能，其实是很简单，创建主表面－－>创建RGB从表面---->创建YV12从表面，然后将YV12数据复制到 YV12表面， Blt到RGB表面（在这一步中显卡自动完成YV12到RGB的转换), 然后取RGB表面的HDC, 就可以利用TextOut， FillRect, Line之类的Win32 API来绘图写字了，最后，将RGB表面Blt到主表面，这个过程就算是结束了。

       需要说明的是，这个过程只用到显卡的运算能力，没有用到CPU，所以CPU占有率不会提高，但对于显卡来说，要占用显卡的GPU和显存的带宽。显卡的性能就显得比较重要了。

      以下是实现代码：(由于商业原因，一些细节代码被取消了，但整体技术实现流程是完整的)

bool CVideoDraw::InitDirectX(HWND hWnd, int nWidth, int nHeight) {  
if (DirectDrawCreateEx(NULL, (LPVOID\*)&m\_lpDD, IID\_IDirectDraw7, NULL) != DD\_OK)  
   return false;  
    
if (m\_lpDD->SetCooperativeLevel(hWnd, DDSCL\_NORMAL) != DD\_OK){  
   ClearDirectX();  
   return false;  
}

ZeroMemory(&m\_ddsd, sizeof(m\_ddsd));  
m\_ddsd.dwSize = sizeof(m\_ddsd);  
m\_ddsd.dwFlags = DDSD\_CAPS;  
m\_ddsd.ddsCaps.dwCaps = DDSCAPS\_PRIMARYSURFACE;  
if (m\_lpDD->CreateSurface(&m\_ddsd,& m\_lpDDSPrimary, NULL) != DD\_OK){  
   ClearDirectX();  
   return false;  
}

LPDIRECTDRAWCLIPPER pcClipper;  
if( m\_lpDD->CreateClipper(0,& pcClipper, NULL) != DD\_OK) {  
   ClearDirectX();  
   return false;  
}

if( pcClipper->SetHWnd(0, m\_hWnd) != DD\_OK) {  
   ClearDirectX();  
   return false;  
}

if( m\_lpDDSPrimary->SetClipper(pcClipper) != DD\_OK) {  
   ClearDirectX();  
   return false;  
}

pcClipper->Release();

// 创建离屏表面对象  
ZeroMemory(&m\_ddsd, sizeof(m\_ddsd));  
m\_ddsd.dwSize = sizeof(m\_ddsd);  
m\_ddsd.ddsCaps.dwCaps = DDSCAPS\_OFFSCREENPLAIN|DDSCAPS\_VIDEOMEMORY ; // DDSCAPS\_VIDEOMEMORY; //DDSCAPS\_OVERLAY DDSCAPS\_OFFSCREENPLAIN;  
m\_ddsd.dwFlags = DDSD\_CAPS | DDSD\_HEIGHT | DDSD\_WIDTH | DDSD\_PIXELFORMAT;  
m\_ddsd.dwWidth = nWidth;  
m\_ddsd.dwHeight = nHeight;  
m\_ddsd.ddpfPixelFormat.dwSize = sizeof(DDPIXELFORMAT);  
m\_ddsd.ddpfPixelFormat.dwFlags = DDPF\_FOURCC | DDPF\_YUV ;  
m\_ddsd.ddpfPixelFormat.dwFourCC = MAKEFOURCC('Y','V','1','2');  
m\_ddsd.ddpfPixelFormat.dwYUVBitCount = 8;  
if (m\_lpDD->CreateSurface(&m\_ddsd,& m\_lpDDSOffScrYUV, NULL) != DD\_OK) {  
   ClearDirectX();  
   return false;  
}

ZeroMemory(&m\_ddsd, sizeof(m\_ddsd));  
m\_ddsd.dwSize = sizeof(m\_ddsd);  
m\_ddsd.ddsCaps.dwCaps = DDSCAPS\_OFFSCREENPLAIN|DDSCAPS\_VIDEOMEMORY; // DDSCAPS\_VIDEOMEMORY; //DDSCAPS\_OVERLAY DDSCAPS\_OFFSCREENPLAIN;  
m\_ddsd.dwFlags = DDSD\_CAPS | DDSD\_HEIGHT | DDSD\_WIDTH;  
m\_ddsd.dwWidth = nWidth;  
m\_ddsd.dwHeight = nHeight;  
if (m\_lpDD->CreateSurface(&m\_ddsd,& m\_lpDDSOffScrRGB, NULL) != DD\_OK) {  
   ClearDirectX();  
   return false;  
}

return true;  
}

void CVideoDraw::ClearDirectX() {  
if( m\_lpDD != NULL ){  
   SAFE\_RELEASES(m\_lpDDSOffScrRGB);  
   SAFE\_RELEASES(m\_lpDDSOffScrYUV);  
   SAFE\_RELEASES(m\_lpDDSPrimary);  
   SAFE\_RELEASES(m\_lpDD);  
}  
}

bool CVideoDraw::Draw(LPBYTE pBuffer, int nWidth, int nHeight, int nSec) {  
HRESULT ddRval;  
RECT rctDest;    // 目标区域  
RECT rctSour;    // 源区域

for(int nTry=0; nTry < 5; nTry++){  
   ddRval = m\_lpDDSOffScrYUV->Lock(NULL,&m\_ddsd, DDLOCK\_WAIT|DDLOCK\_WRITEONLY, NULL);  
   if( ddRval == DDERR\_SURFACELOST ) {  
    ddRval = m\_lpDDSOffScrYUV->Restore();  
   }  
   if( ddRval == DD\_OK ){  
    break;  
   }  
}

if( ddRval != DD\_OK ) return false;

int i=0;  
LPBYTE lpSurf = (LPBYTE)m\_ddsd.lpSurface;  
LPBYTE lpY = (LPBYTE)pBuffer;  
LPBYTE lpV = (LPBYTE)(pBuffer + nWidth \* nHeight);  
LPBYTE lpU = (LPBYTE)(pBuffer + nWidth \* nHeight \* 5 / 4);

int nOffset = 0;  
int value1 = 0 ;  
int value2 = 0 ;  
int value3 = 0 ;  
int value4 = 0 ;

lpY += nOffset;  
for(i=0; i<m\_ddsd.dwHeight; i++)  
{  
   memcpy(lpSurf, lpY, m\_ddsd.dwWidth);  
   lpY += nWidth;  
   lpSurf += m\_ddsd.lPitch;  
}  
  
value1 = m\_ddsd.dwHeight/2;  
value2 = m\_ddsd.dwWidth / 2;  
value3 = nWidth / 2;  
value4 = m\_ddsd.lPitch / 2;

for(i=0; i<value1; i++)  
{  
   memcpy(lpSurf, lpU, value2);  
   lpU += value3;  
   lpSurf += value4;  
}

for(i=0; i<value1; i++)  
{  
   memcpy(lpSurf, lpV, value2);  
   lpV += value3;  
   lpSurf += value4;  
}  
m\_lpDDSOffScrYUV->Unlock(NULL);

SetRect(&rctDest, 0,0, nWidth, nHeight);  
ddRval = m\_lpDDSOffScrRGB->Blt(&rctDest, m\_lpDDSOffScrYUV, NULL, DDBLT\_WAIT, NULL);  
if( ddRval != DD\_OK)  
   return false;

ddRval = m\_lpDDSOffScrRGB->Lock(NULL,&m\_ddsd, DDLOCK\_WAIT|DDLOCK\_WRITEONLY, NULL);  
if( ddRval == DDERR\_SURFACELOST )  
   ddRval = m\_lpDDSOffScrRGB->Restore();  
    
if( ddRval != DD\_OK )  
   return false;

HDC hdc = NULL;  
m\_lpDDSOffScrRGB->GetDC(&hdc);  
if( hdc )  
{  
   TCHAR szText[64];  
   int thh = nSec/3600;  
   int tmm = (nSec%3600)/60;  
   int tss = nSec%60;  
   wsprintf(szText, \_T("%02d:%02d:%02d"), thh, tmm, tss);  
   SetBkMode(hdc, TRANSPARENT);  
   ::SetTextColor(hdc, RGB(255,0,0));  
   TextOut(hdc, 1, 1, szText, wcslen(szText));  
   m\_lpDDSOffScrRGB->ReleaseDC(hdc);  
}  
m\_lpDDSOffScrRGB->Unlock(NULL);

// Blt到主表面上  
rctSour.left = 0;  
rctSour.top = 0;  
rctSour.right = m\_ddsd.dwWidth;  
rctSour.bottom = m\_ddsd.dwHeight;  
rctDest = m\_rcDraw;  
::ClientToScreen(m\_hWnd, (LPPOINT)&rctDest.left);  
::ClientToScreen(m\_hWnd, (LPPOINT)&rctDest.right);

ddRval = m\_lpDDSPrimary->Blt(&rctDest, m\_lpDDSOffScrRGB, NULL, DDBLT\_WAIT, NULL);  
  
return (ddRval==DD\_OK);  
}