```
#include "mvc2api.h"
#include "CMXFParserModule.h"
#include "mvc2api_securitymanager.h"

#pragma warning(disable : 4996)

#include <cstdio>
#include <cstdlib>
#include <cstring>

using namespace mvc2;

/// XLQ:
#define MM_LINUX
#define MVC2API_NETWORK_ONLY
#define IMB_IP_ADDRESS "10.7.75.1"
/// XLQ

const char* bin2hex(unsigned char* bin_buf,
    unsigned int bin_len,
    char* str_buf,
    unsigned int str_len)
{
    if ( bin_buf == 0
        || str_buf == 0
        || ((bin_len * 2) + 1) > str_len )
        return 0;

//#ifdef CONFIG_RANDOM_UUID
//  const char* use_random_uuid  = getenv("KM_USE_RANDOM_UUID");
//  if ( use_random_uuid != 0 && use_random_uuid[0] != 0 && use_random_uuid[0] != '0' )
//    return bin2hex_rand(bin_buf, bin_len, str_buf, str_len);
//#endif

char* p = str_buf;

for ( unsigned int i = 0; i < bin_len; i++ )
{
    *p = (bin_buf[i] >> 4) & 0x0f;
    *p += *p < 10 ? 0x30 : 0x61 - 10;
    p++;

    *p = bin_buf[i] & 0x0f;
    *p += *p < 10 ? 0x30 : 0x61 - 10;
    p++;
}

*p = '\0';
return str_buf;
```

```
}

TMmRc TransferAudio_CT(MvcDecoder& dec,
    const unsigned char *audioDataBuffer,
    unsigned long &audioDataLength,
    unsigned int &m_uPlaintextOffset,
    unsigned int &m_uSourceLength,
    bool &m_bHmacFlag,
    char *m_cKeyID)
{
    DataBuffer dataBuffer;
    TMmRc ret = MMRC_Ok;

    ret = dec.getDataBuffer(dataBuffer, 36000 * 10);
    if (MM_IS_ERROR(ret))
    {
        printf("could not get audio databuffer -> abort\n");
        return(ret);
    }

    uint32_t readbytes = 0;

    /// XLQ:从IV段、CV段这32个字节的后面PlainText Offset段的开始取值，到E(V)的最后一个字节！！！
    if(true == m_bHmacFlag)
    {
        memcpy( dataBuffer.getBufferAddress(),
            (audioDataBuffer + 32),
            (audioDataLength - 32 - 56) );

        readbytes = audioDataLength - 32 - 56;

        dataBuffer.setMicValue( (audioDataBuffer + (audioDataLength - 56) ), 56);
    }
    else
    {
        memcpy( dataBuffer.getBufferAddress(),
            (audioDataBuffer + 32),
            (audioDataLength - 32) );

        readbytes = audioDataLength - 32;
    }

    dataBuffer.setDecryptionSize(m_uPlaintextOffset, m_uSourceLength);

    char keyId[64] = "";
    bin2hex((unsigned char *)m_cKeyID, 16, keyId, 64);
    //printf("%s", keyId);

    mvc2::UuidValue keyid(keyId);
    //mvc2::UuidValue keyid = "25091308b27ed5bf19daec4a1b32a6be";
```

```
    dataBuffer.setKeyId(keyid, audioDataBuffer, audioDataBuffer + 16);

    //dataBuffer.setKeyIndex(1, audioDataBuffer, audioDataBuffer + 16);

    uint32_t padding = (16 - (readbytes & 0x0f)) & 0x0f;
    uint8_t *buf = dataBuffer.getBufferAddress();
    for (uint32_t i = 0; i < padding; i++)
    {
        buf[readbytes + i] = 0;
    }

    dataBuffer.send(readbytes + padding);

    return(ret);
}

TMmRc TransferVideo_CT(MvcDecoder& dec,
    const unsigned char *videoDataBuffer,
    unsigned long &videoDataLength,
    unsigned int &m_uPlaintextOffset,
    unsigned int &m_uSourceLength,
    bool &m_bHmacFlag,
    char *m_cKeyID)
{
    DataBuffer dataBuffer;
    TMmRc ret = MMRC_Ok;

    ret = dec.getDataBuffer(dataBuffer, 2 * 1024 * 1024);
    if (MM_IS_ERROR(ret))
    {
        printf("could not get databuffer -> abort\n");
        return(ret);
    }

    uint32_t readbytes = 0;

    /// XLQ:从IV段、CV段这32个字节的后面PlainText Offset段的开始取值，到E(V)的最后一个字节！！！
    if(true == m_bHmacFlag)
    {
        memcpy( dataBuffer.getBufferAddress(),
            (videoDataBuffer + 32),
            (videoDataLength - 32 - 56) );

        readbytes = videoDataLength - 32 - 56;

        dataBuffer.setMicValue( (videoDataBuffer + (videoDataLength - 56) ), 56);
    }
    else
    {
```

```
        memcpy( dataBuffer.getBufferAddress(),
            (videoDataBuffer + 32),
            (videoDataLength - 32) );

        readbytes = videoDataLength - 32;
    }

    dataBuffer.setDecryptionSize(m_uPlaintextOffset, m_uSourceLength);

    char keyId[64] = "";
    bin2hex((unsigned char *)m_cKeyID, 16, keyId, 64);
    //printf("%s", keyId);

    mvc2::UuidValue keyid(keyId);
    //mvc2::UuidValue keyid = "007203b983345b84bcb0c928e8bab01b";

    dataBuffer.setKeyId(keyid, videoDataBuffer, videoDataBuffer + 16);

    //dataBuffer.setKeyIndex(0, videoDataBuffer, videoDataBuffer + 16);

    uint32_t padding = (16 - (readbytes & 0x0f)) & 0x0f;
    uint8_t *buf = dataBuffer.getBufferAddress();
    for (uint32_t i = 0; i < padding; i++)
    {
        buf[readbytes + i] = 0;
    }
    //  datbuf.setUserData(framecount);
    //printf("sending pic %d\n",framecount);
    //printf("readbytes %d\n", readbytes);
    //printf("padding %d\n", padding);
    dataBuffer.send(readbytes + padding);
    //fclose(infile);

    //datbuf.wait(100);

    return(ret);
}


TMmRc TransferVideo_PT(MvcDecoder& dec, char *videoDataBuffer, unsigned long &videoDataLength)
{
    DataBuffer dataBuffer;
    TMmRc ret = MMRC_Ok;

    ret = dec.getDataBuffer(dataBuffer, 2 * 1024 * 1024);
    if (MM_IS_ERROR(ret))
    {
        printf("could not get databuffer -> abort\n");
        return(ret);
    }
```

```
    memcpy(dataBuffer.getBufferAddress(), videoDataBuffer, videoDataLength);
    uint32_t readbytes = videoDataLength;

    // copy one frame here (as an example we fill the buffer with 1's)
    //uint32_t readbytes = static_cast<uint32_t>(fread(datbuf.getBufferAddress(),1,(size_t)(datbuf.getFreeSize()),infile));
    //int readbytes = fread(datbuf.getBufferAddress(), 1, 1301870, infile);
    //printf("datbuf.getFreeSize() %d\n", datbuf.getFreeSize());
    //printf("readbytes1 %d\n", readbytes);
    uint32_t padding = (16 - (readbytes & 0x0f)) & 0x0f;
    uint8_t *buf = dataBuffer.getBufferAddress();
    for (uint32_t i = 0; i < padding; i++)
    {
        buf[readbytes + i] = 0;
    }
    //  datbuf.setUserData(framecount);
    //printf("sending pic %d\n",framecount);
    //printf("readbytes %d\n", readbytes);
    //printf("padding %d\n", padding);
    dataBuffer.send(readbytes + padding);
    //fclose(infile);

    //datbuf.wait(100);

    return(ret);
}

TMmRc TransferVideo_PT(MvcDecoder& dec,
    MvcDecoder& dec_right,
    char *videoDataBuffer,
    unsigned long &videoDataLength,
    char *videoDataBuffer_right,
    unsigned long &videoDataLength_right)
{
    DataBuffer dataBuffer;
    TMmRc ret = MMRC_Ok;

    ret = dec.getDataBuffer(dataBuffer, 2 * 1024 * 1024);
    if (MM_IS_ERROR(ret))
    {
        printf("could not get databuffer -> abort\n");
        return(ret);
    }

    memcpy(dataBuffer.getBufferAddress(), videoDataBuffer, videoDataLength);
    uint32_t readbytes = videoDataLength;

    // copy one frame here (as an example we fill the buffer with 1's)
    //uint32_t readbytes = static_cast<uint32_t>(fread(datbuf.getBufferAddress(),1,(size_t)(datbuf.getFreeSize()),infile));
    //int readbytes = fread(datbuf.getBufferAddress(), 1, 1301870, infile);
```

```
    //printf("datbuf.getFreeSize() %d\n", datbuf.getFreeSize());
    //printf("readbytes1 %d\n", readbytes);
    uint32_t padding = (16 - (readbytes & 0x0f)) & 0x0f;
    uint8_t *buf = dataBuffer.getBufferAddress();
    for (uint32_t i = 0; i < padding; i++)
    {
        buf[readbytes + i] = 0;
    }
    //  datbuf.setUserData(framecount);
    //printf("sending pic %d\n",framecount);
    //printf("readbytes %d\n", readbytes);
    //printf("padding %d\n", padding);
    dataBuffer.send(readbytes + padding);
    //fclose(infile);

    //datbuf.wait(100);


    /// XLQ:右眼
    DataBuffer dataBuffer_right;
    TMmRc ret_right = MMRC_Ok;

    ret_right = dec_right.getDataBuffer(dataBuffer_right, 2 * 1024 * 1024);
    if (MM_IS_ERROR(ret_right))
    {
        printf("could not get databuffer -> abort\n");
        return(ret_right);
    }

    memcpy(dataBuffer_right.getBufferAddress(), videoDataBuffer_right, videoDataLength_right);
    uint32_t readbytes_right = videoDataLength_right;

    // copy one frame here (as an example we fill the buffer with 1's)
    //uint32_t readbytes = static_cast<uint32_t>(fread(datbuf.getBufferAddress(),1,(size_t)(datbuf.getFreeSize()),infile));
    //int readbytes = fread(datbuf.getBufferAddress(), 1, 1301870, infile);
    //printf("datbuf.getFreeSize() %d\n", datbuf.getFreeSize());
    //printf("readbytes1 %d\n", readbytes);
    uint32_t padding_right = (16 - (readbytes_right & 0x0f)) & 0x0f;
    uint8_t *buf_right = dataBuffer_right.getBufferAddress();
    for (uint32_t i_right = 0; i_right < padding_right; i_right++)
    {
        buf_right[readbytes_right + i_right] = 0;
    }
    //  datbuf.setUserData(framecount);
    //printf("sending pic %d\n",framecount);
    //printf("readbytes %d\n", readbytes);
    //printf("padding %d\n", padding);
    dataBuffer_right.send(readbytes_right + padding_right);
    //fclose(infile);
```

```
    //datbuf.wait(100);

    return(ret);
}

TMmRc TransferAudio_PT(MvcDecoder& dec, char *audioDataBuffer, unsigned long &audioDataLength)
{
    DataBuffer dataBuffer;
    TMmRc ret = MMRC_Ok;

    ret = dec.getDataBuffer(dataBuffer, 36000 * 10);
    if (MM_IS_ERROR(ret))
    {
        printf("could not get audio databuffer -> abort\n");
        return(ret);
    }

    memcpy(dataBuffer.getBufferAddress(), audioDataBuffer, audioDataLength);
    uint32_t readbytes = audioDataLength;

    uint32_t padding = ( 16 - (readbytes & 0x0f) ) & 0x0f;
    uint8_t *buf = dataBuffer.getBufferAddress();
    for (uint32_t i = 0; i < padding; i++)
    {
        buf[readbytes + i] = 0;
    }
    //  datbuf.setUserData(framecount);
    //printf("sending pic %d\n",framecount);
    //printf("readbytes %d\n", readbytes);
    //printf("padding %d\n", padding);
    dataBuffer.send(readbytes + padding);

    //datbuf.send(audiosDataLength);
    return(ret);
}

#if 1

/// XLQ:播放明文3D的例子。
int main(int argc, char **argv)
{
    printf("这是一个播放明文3D影片的例子\n");

    TMmRc ret;

    /// XLQ:连接板卡。
    // use first card example
    MvcDevice mvcdevice;
    /// XLQ:
#ifndef MVC2API_NETWORK_ONLY
```

```
    if (! ( mvcdevice = MvcDeviceIterator().getIndex(0) ) )
#else
    if (! ( mvcdevice = MvcNetDeviceIterator(IMB_IP_ADDRESS).getIndex(0) ) )
#endif
        /// XLQ
    {
        printf("MVC card not found\n");
        exit(0);
    }
    else
    {
        printf("MVC card is found!\n");
    }


    /// XLQ:视频左眼部分
    Jpeg2kDecoder j2kdecLeft(&ret, mvcdevice);
    if (MM_IS_ERROR(ret))
    {
        printf("failed to create Jpeg2k left video decoder: %d\n",ret);
        exit(0);
    }

    j2kdecLeft.setFrameRate(24.00);                    // set frame rate, so we don't need to set timestamps anymore


    /// XLQ:视频右眼部分
    Jpeg2kDecoder j2kdecRight(&ret, j2kdecLeft);
    if (MM_IS_ERROR(ret))
    {
        printf("failed to create Jpeg2k right video decoder: %d\n",ret);
        exit(0);
    }

    j2kdecRight.setFrameRate(24.00);                   // set frame rate, so we don't need to set timestamps anymore


    /// XLQ:创建左眼VideoOutput
    // create the video output
    //VideoOutput videoout(&ret, mvcdevice, VideoOutput::VideoProperty_Default);
    VideoOutput videoOutLeft(&ret, mvcdevice, VideoOutput::VideoProperty_Dual_HDTV);
    if (MM_IS_ERROR(ret))
    {
        printf("failed to create left video output: %d\n",ret);
        exit(0);
    }

    // setting a video mode (optional)
    //if (MM_IS_ERROR(ret = videoout.setVideoMode(VideoMode::Mode_1920_1080_2400_p)))
    if (MM_IS_ERROR(ret = videoOutLeft.setVideoMode(VideoMode::Mode_2048_1080_2400_p)))
```

```
    //if (MM_IS_ERROR(ret = videoout.setVideoMode(VideoMode::Mode_2048_1080_4800_p)))
    {
        printf("failed to set left video mode: %d\n",ret);
    }

    //printf("getActiveVideoMode():%d\n", videoout.getActiveVideoMode());


    /// XLQ:创建右眼VideoOutput!
    VideoOutput videoOutRight(&ret, videoOutLeft);
    if (MM_IS_ERROR(ret))
    {
        printf("failed to create right video right output: %d\n",ret);
        exit(0);
    }

    // setting a video mode (optional)
    //if (MM_IS_ERROR(ret = videoout.setVideoMode(VideoMode::Mode_1920_1080_2400_p)))
    if (MM_IS_ERROR(ret = videoOutRight.setVideoMode(VideoMode::Mode_2048_1080_2400_p)))
    {
        printf("failed to set right video mode: %d\n",ret);
    }

    //printf("video right getActiveVideoMode():%d\n", videoout_right.getActiveVideoMode());


    //if (MM_IS_ERROR(ret = videoout_right.setOutputActivity(VideoOutput::Activity_Primary_Secondary_as_Overlay)))
    //{
    //    printf("failed to video right setOutputActivity: %d\n",ret);
    //}

    //    // connect decoder with video output
    //if (MM_IS_ERROR(ret = j2kdec_right.connectOutput(videoout)))
    //{
    //    printf("failed to connect decoder with video output: %d\n",ret);
    //    exit(0);
    //}

    // connect decoder with video output
    if (MM_IS_ERROR(ret = j2kdecLeft.connectOutput(videoOutLeft)))
    {
        printf("failed to connect left decoder with left video output: %d\n",ret);
        exit(0);
    }

    if (MM_IS_ERROR(ret = j2kdecRight.connectOutput(videoOutRight)))
    {
        printf("failed to connect right decoder with right video output: %d\n",ret);
        exit(0);
    }
```

```
//// XLQ:字幕部分
//SubtitleDecoder subtitleDec(&ret, mvcdevice);
//if (MM_IS_ERROR(ret))
//{
//    printf("failed to create subtitle decoder: %d\n", ret);
//    exit(0);
//}

//// connect decoder with video output
//if (MM_IS_ERROR(ret = subtitleDec.connectOutput(videooutLeft)))
//{
//    printf("failed to connect decoder with video output: %d\n",ret);
//    exit(0);
//}


/// XLQ:音频部分
PCMDecoder pcmDec(&ret, mvcdevice, 24, 6);
if (MM_IS_ERROR(ret))
{
    printf("failed to create pcm audio decoder: %d\n", ret);
    exit(0);
}

AudioOutput audioout(&ret, mvcdevice, 6);
if (MM_IS_ERROR(ret))
{
    printf("failed to create audio output: %d\n",ret);
    exit(0);
}

if (MM_IS_ERROR(ret = audioout.setOutputFrequency(48000)))
{
    printf("failed to set audio output frequency: %d\n",ret);
}


// connect decoder with video output
if (MM_IS_ERROR(ret = pcmDec.connectOutput(audioout)))
{
    printf("failed to connect decoder with audio output: %d\n",ret);
    exit(0);
}
///


/// XLQ:播放控制部分
PlaybackControl playctrl(&ret, mvcdevice);
if (MM_IS_ERROR(ret))
```

```
{
    printf("failed to create playback control: %d\n",ret);
    exit(0);
}

// connect decoder with playback
ret = playctrl.connect(j2kdecLeft);
if (MM_IS_ERROR(ret))
{
    printf("failed to connect playback control with left video decoder: %d\n",ret);
    exit(0);
}

ret = playctrl.connect(j2kdecRight);
if (MM_IS_ERROR(ret))
{
    printf("failed to connect playback control with right video decoder: %d\n",ret);
    exit(0);
}

// connect decoder with playback
//ret = playctrl.connect(subtitleDec);
//if (MM_IS_ERROR(ret))
//{
//    printf("failed to connect playback control with subtilte decoder: %d\n",ret);
//    exit(0);
//}

/// XLQ:
ret = playctrl.connect(pcmDec);
if (MM_IS_ERROR(ret))
{
    printf("failed to connect playback control with audio decoder: %d\n",ret);
    exit(0);
}
///

char *videoDataBufferLeft = new char[2 * 1024 * 1024];
unsigned long videoDataLengthLeft = 0;
char *videoDataBufferRight = new char[2 * 1024 * 1024];
unsigned long videoDataLengthRight = 0;
char *audioDataBuffer = new char[36000 * 10];
unsigned long audioDataLength = 0;

CMXFParserModule cmxfParserModule;
unsigned long vidoeFrameSum = 0;
unsigned long audioFrameSum = 0;
cmxfParserModule.Init3DVideoParser("D:\\LIXIAOLONG_3D\\LIXIAOLONG_JPEG3D\\LIXIAOLONG_JPEG3D_01.mxf", vidoeFrameSum);
cmxfParserModule.InitAudioParser("D:\\LIXIAOLONG_3D\\LIXIAOLONG_JPEG3D\\LIXIAOLONG_JPEG3D_audio_01.mxf", audioFrameSum);
```

```
    double m_dAspectRatio = 0;
    unsigned long m_uWidthSize = 0;
    unsigned long m_uHeightSize = 0;
    unsigned long m_uFrameRate = 0;
    bool m_bHmacFlag1 = false;
    bool m_bCryptoFlag1 = false;
    char m_cKeyID1[16] = "";
    cmxfParserModule.Get3DVideoInfo(m_dAspectRatio,
        m_uWidthSize,
        m_uHeightSize,
        m_uFrameRate,
        m_bHmacFlag1,
        m_bCryptoFlag1,
        m_cKeyID1);

    if(true == m_bCryptoFlag1)
    {
        /// XLQ:播放的是密文。
        return -1;
    }

    unsigned long m_uSamplingRate = 0;
    unsigned long m_uChannelCount = 0;
    unsigned long m_uBitsPerSample = 0;
    bool m_bHmacFlag2 = false;
    bool m_bCryptoFlag2 = false;
    char m_cKeyID2[16] = "";
    cmxfParserModule.GetAudioInfo(m_uSamplingRate,
        m_uChannelCount,
        m_uBitsPerSample,
        m_bHmacFlag2,
        m_bCryptoFlag2,
        m_cKeyID2);

    if(true == m_bCryptoFlag2)
    {
        /// XLQ:播放的是密文。
        return -1;
    }


//    /// XLQ:将字幕数据作为缓冲提前载入
//    uint8_t *data = new uint8_t[10 * 1024 * 1024];
//    uint32_t dataSize = 0;
//
//    FILE *subtitleXmlFile = fopen("D:\\dieying3_xyz_sub\\a00ccb11-62d4-46b1-bd05-c14b86bea9d7\\dieying3_chinese_subtitle.xml", "rb");
//    fread(data, 1, 10244, subtitleXmlFile);
//    dataSize = 10244;
//    fclose(subtitleXmlFile);
//
```

```
//    uint32_t resourceId = 0;
//    ret = subtitleDec.sendSubtitleFile(data, dataSize, &resourceId);
//    if (MM_IS_ERROR(ret))
//    {
//        printf("failed to sendSubtitleFile: %d\n", ret);
//        exit(0);
//    }
//    else
//    {
//        printf("resourceId=%d\n", resourceId);
//    }
//
//    //// XLQ:载入字体文件
//#if 1
//
//    FILE *subtitleFontFile = fopen("D:\\dieying3_xyz_sub\\a00ccb11-62d4-46b1-bd05-c14b86bea9d7\\simhei-C.ttf", "rb");
//    fread(data, 1, 54904, subtitleFontFile);
//    dataSize = 54904;
//    fclose(subtitleFontFile);
//
//    ret = subtitleDec.sendOverlayElement("simhei-C.ttf", data, dataSize, resourceId);
//    if (MM_IS_ERROR(ret))
//    {
//        printf("failed to sendOverlayElement: %d\n", ret);
//        exit(0);
//    }
//
//#else
//
//    OverlayElementDataBuffer element;
//    subtitleDec.getDataBuffer(element);
//    if (element)
//    {
//        element.setElementName("simhei-C.ttf", resourceId);
//        FILE *fontfile = fopen("D:\\dieying3_xyz_sub\\a00ccb11-62d4-46b1-bd05-c14b86bea9d7\\simhei-C.ttf","rb");
//        if (fontfile)
//        {
//            ret = element.send(fread(element.getBufferAddress(),1,element.getFreeSize(),fontfile));
//            if (MM_IS_ERROR(ret))
//            {
//                printf("failed to element.send: %d\n", ret);
//                exit(0);
//            }
//            fclose(fontfile);
//        }
//        else
//        {
//            printf("could not open font: %s\n","D:\\dieying3_xyz_sub\\a00ccb11-62d4-46b1-bd05-c14b86bea9d7\\simhei-C.ttf");
//        }
//    }
```

```
//
//#endif
//
    //ret = subtitleDec.enableSubtitles(0, SubtitleDecoder::Render_Soft_Shadows);
    //if (MM_IS_ERROR(ret))
    //{
    //    printf("failed to enableSubtitles: %d\n", ret);
    //    exit(0);
    //}


    for (int i = 0; i < 20; i++)                        // 20 picture preload before run
    {
        unsigned int m_uPlaintextOffsetLeft = 0;
        unsigned int m_uSourceLengthLeft = 0;
        unsigned int m_uPlaintextOffsetRight = 0;
        unsigned int m_uSourceLengthRight = 0;

        cmxfParserModule.Get3DFrameData(i,
            videoDataBufferLeft,
            videoDataLengthLeft,
            m_uPlaintextOffsetLeft,
            m_uSourceLengthLeft,
            videoDataBufferRight,
            videoDataLengthRight,
            m_uPlaintextOffsetRight,
            m_uSourceLengthRight);

        ret = TransferVideo_PT(j2kdecLeft,
            videoDataBufferLeft,
            videoDataLengthLeft);
        if (ret)
        {
            exit(0);
        }

        ret = TransferVideo_PT(j2kdecRight,
            videoDataBufferRight,
            videoDataLengthRight);
        if (ret)
        {
            exit(0);
        }

        unsigned int m_uPlaintextOffsetAudio = 0;
        unsigned int m_uSourceLengthAudio = 0;

        cmxfParserModule.GetAudioFrameData(i,
            audioDataBuffer,
            audioDataLength,
```

```
            m_uPlaintextOffsetAudio,
            m_uSourceLengthAudio);

        ret = TransferAudio_PT(pcmDec, audioDataBuffer, audioDataLength);
        if (ret)
        {
            exit(0);
        }

    }


    if (ret == MMRC_Ok)
    {

        playctrl.run();


        for (int i = 0; i < vidoeFrameSum; i++)
        {
            unsigned int m_uPlaintextOffsetLeft = 0;
            unsigned int m_uSourceLengthLeft = 0;
            unsigned int m_uPlaintextOffsetRight = 0;
            unsigned int m_uSourceLengthRight = 0;

            cmxfParserModule.Get3DFrameData(i,
                videoDataBufferLeft,
                videoDataLengthLeft,
                m_uPlaintextOffsetLeft,
                m_uSourceLengthLeft,
                videoDataBufferRight,
                videoDataLengthRight,
                m_uPlaintextOffsetRight,
                m_uSourceLengthRight);

            ret = TransferVideo_PT(j2kdecLeft,
                videoDataBufferLeft,
                videoDataLengthLeft);
            if (ret)
            {
                exit(0);
            }

            ret = TransferVideo_PT(j2kdecRight,
                videoDataBufferRight,
                videoDataLengthRight);
            if (ret)
            {
                exit(0);
            }
```

Here we get double frames from .mxf file,so there must be difference between the left eye image and the right

```
            unsigned int m_uPlaintextOffsetAudio = 0;
            unsigned int m_uSourceLengthAudio = 0;

            cmxfParserModule.GetAudioFrameData(i,
                audioDataBuffer,
                audioDataLength,
                m_uPlaintextOffsetAudio,
                m_uSourceLengthAudio);

            ret = TransferAudio_PT(pcmDec, audioDataBuffer, audioDataLength);
            if (ret)
            {
                exit(0);
            }
        }

        j2kdecLeft.setEndOfStream();
        j2kdecRight.setEndOfStream();

        /// XLQ:
        pcmDec.setEndOfStream();
        /// XLQ
    }

    if (ret != MMRC_Ok)
    {
        //printf("picture transfer failed (%s)\n",filename);
        printf("picture transfer failed\n");
    }

    playctrl.waitForEndOfStream();

    printf("End of stream reached\n");


    if(NULL != audioDataBuffer)
    {
        delete[] audioDataBuffer;
        audioDataBuffer = NULL;
        audioDataLength = 0;
    }

    if(NULL != videoDataBufferLeft)
    {
        delete[] videoDataBufferLeft;
        videoDataBufferLeft = NULL;
        videoDataLengthLeft = 0;
    }
```

```
    if(NULL != videoDataBufferRight)
    {
        delete[] videoDataBufferRight;
        videoDataBufferRight = NULL;
        videoDataLengthRight = 0;
    }

    //if(NULL != data)
    //{
    //    delete[] data;
    //    data = NULL;
    //    dataSize = 0;
    //}

    return 0;
}

#else

#endif
```