//DetecBlackFrame.h

//Detection Black Frame

//By kiok ahn

//2009.06.03

#ifndef \_DETECTION\_BLACK\_FRAME\_H\_INCLUDE\_

#define \_DETECTION\_BLACK\_FRAME\_H\_INCLUDE\_

#include "stdafx.h"

#include "TTypeDefine.h"

#include "CommonDataApi.h"

typedef struct tagRGB

{

BYTE R;

BYTE B;

BYTE G;

} RGBCOMP, \*LPRGBCOMP;

class CRGBCompoment : public tagRGB

{

public:

CRGBCompoment(){R=0;G=0;B=0;};

CRGBCompoment(const CRGBCompoment& op){R=op.R;G=op.G;B=op.B;};

CRGBCompoment(const tagRGB& op){R=op.R;G=op.G;B=op.B;};

CRGBCompoment(const BYTE& r, const BYTE& g, const BYTE& b ){R=r;G=g;B=b;};

virtual ~CRGBCompoment(){};

CRGBCompoment& operator^(const CRGBCompoment& op)

{

// BYTE\* pItor = (LPBYTE)&op;

// R = R^(\*pItor++);

// G = G^(\*pItor++);

// B = B^(\*pItor);

R = R^op.R;

G = G^op.G;

B = B^op.B;

return \*this;

};

CRGBCompoment& operator^(const RGBCOMP& op)

{

// BYTE\* pItor = (LPBYTE)&op;

// R = R^(\*pItor++);

// G = G^(\*pItor++);

// B = B^(\*pItor);

R = R^op.R;

G = G^op.G;

B = B^op.B;

return \*this;

};

bool operator==(const CRGBCompoment& op)

{

if( R == op.R && G == op.G && B == op.B ) return true;

return false;

};

bool operator==(const RGBCOMP& op)

{

if( R == op.R && G == op.G && B == op.B ) return true;

return false;

};

bool operator!=(const CRGBCompoment& op)

{

if( R == op.R && G == op.G && B == op.B ) return false;

return true;

};

bool operator!=(const RGBCOMP& op)

{

if( R == op.R && G == op.G && B == op.B ) return false;

return true;

};

};

class CDetectionBlackFrame

{

public:

CDetectionBlackFrame(void);

~CDetectionBlackFrame(void);

public:

HRESULT CheckStream ( LPAVDATA lpAVData );

void SetCallback ( LPFN\_AVRESULT lpCallback, LPVOID lpUserData );

HRESULT FinishedStream( void );

HRESULT SetOption ( LPSTR\_MAP lpmapOption );

protected:

HRESULT SendErrorFrame(LPAVDATA lpAVData, const int& nTargetFrame);

HRESULT SendAlramFrame(LPAVDATA lpAVData, const int& nTargetFrame);

protected:

UINT m\_nStartFrame;

UINT m\_nEndFrame ;

DOUBLE m\_dStartTime ;

HINSTANCE m\_hCommonDLL ;

UINT m\_nCurFrame ;

HCDVIDEO m\_hCVideo ;

time\_t m\_tSystemStart;

time\_t m\_tSystemEnd ;

CVIDEOAPI m\_VideoApi ;

// CRGBCompoment m\_BeforeRGB ;

CRGBCompoment m\_rgbBlack ;

protected:

void LoadCommonDLL(void);

private:

LPFN\_AVRESULT m\_lpCallback ;

LPVOID m\_lpUserData ;

AVRESULT m\_AVResult ;

DWORD m\_dwDetecedFlag ;

protected:

BOOL m\_bInitFlag ;

//UINT m\_nStopProcessFrame;

BYTE m\_byteNoiseLevel ;

UINT m\_nNoneDetectStart ; //% about duration

UINT m\_nNoneDetectEnd ; //% about duration

INT64 m\_nAlramTime ; // 지속시간

BOOL m\_bAlram ;

BOOL m\_bDetectFalg ;

**BYTE\* m\_pGrayBuffer ;**

**INT m\_nSizeOfGray ;**

};

#endif//\_DETECTION\_BLACK\_FRAME\_H\_INCLUDE\_