**4K video player**

A portable video player based on ffmpeg for windows platform.

It is a simple and light weight player implemention without SDL or other third party library. player core codes are written in C language, KISS and easy to read.

It directly access to video and audio rendering device on specific platforms. on win32 playform, using gdi & direct3d api implements video rendering

Currently, this player can smoothly playback many video files, with low cpu and memory usage (compared with ffplay of offical ffmpeg), high performance, compatibility and stability, and the audio/video also synchronized very well.

testplayer

==========

testplayer is a simple test player for fanplayer.dll

hot-keys for testplayer:

ctrl+1 - open file with single video player mode

ctrl+2 - open files with reat video player mode

ctrl+F - step seek forward

ctrl+B - step seek backward

ctrl+right - volume up

ctrl+left - volume down

Ctrl + K - Get video FPS

Ctrl + P - stop  
left mouse click - pause / play

To adjust the play position of the video, we can simply click on the bottom of the progress bar in the video player.

###### Play / Pause / Stop function

###### Play

Ex) player\_play(m\_ffPlayer);

void player\_play(void \*hplayer)

- Parameters:

`hplayer` - Pointer to the player object returned by player\_open

- Purpose: Start playing

- Return value: This function does not have a return value.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

If vidoe codec is AV1, in fanplyer.dll, we call player\_send message function

player\_send\_message(player->cmnvars.winmsg, MSG\_AV1\_CODEC, 0)  
  
Then In PlayerDlg.cpp:

BOOL CplayerDlg::PreTranslateMessage(MSG \*pMsg)

{

if (pMsg->message == MSG\_FANPLAYER) {

switch (pMsg->wParam) {

case MSG\_AV1\_CODEC:

OnStop();

AfxMessageBox(\_T("This video encoded in AV01 format which is not supported"), MB\_OK);

break;

}

}

}

If video is 4K, in fanplayer.dll we call player\_send message function

player\_send\_message(player->cmnvars.winmsg, MSG\_NOT\_4K, 0);  
  
Then In PlayerDlg.cpp:

BOOL CplayerDlg::PreTranslateMessage(MSG \*pMsg)

{

if (pMsg->message == MSG\_FANPLAYER) {

switch (pMsg->wParam) {

case MSG\_NOT\_4K:

OnStop();

AfxMessageBox(\_T("This video is not 4K video. Check this video again"), MB\_OK);

break;

}

}

}

###### Pause

Ex) player\_pause(m\_ffPlayer);

void player\_pause(void \*hplayer)

- Parameters:

`hplayer` - Pointer to the player object returned by player\_open

- Purpose: pause playing

- Return value: This function does not have a return value.

###### Stop

Ex) player\_close(m\_ffPlayer);

void player\_close(void \*hplayer)

- Parameters:

`hplayer` - Pointer to the player object returned by player\_open

- Purpose: close the player

- Return value: This function does not have a return value.

###### Play position event (in 1 second unit, callback) provided

Ex)

LONGLONG total = 1, pos = 0;

player\_getparam(m\_ffPlayer, PARAM\_MEDIA\_DURATION, &total);

player\_getparam(m\_ffPlayer, PARAM\_MEDIA\_POSITION, &pos);

total: total length of video

pos: current player position of video

void player\_getparam(void \*hplayer, int id, void \*param)

- Parameters:

`hplayer` - Pointer to the player object returned by player\_open

`id` - the parameter id

`param` - the parameter pointer

- Purpose: get player parameters

- Return value: This function does not have a return value.

###### YUV420 frame output (callback) function

Ex) m\_ffPlayer = player\_open(m\_filePath, GetSafeHwnd(), params);

void\* player\_open(char \*file, void \*win, PLAYER\_INIT\_PARAMS \*params)

- Parameters:

`file` - file path

`win` - The win32 platform passes in the window handle

`params` - player initialization parameters

- Purpose: creates a single video player object

- Return value: void\* pointer type, pointing to the player object

Work flow

1. void\* player\_open(char \*file, void \*win, PLAYER\_INIT\_PARAMS \*params)

2. static void\* video\_decode\_thread\_proc(void \*param)

3. pthread\_create(&mainplayer->front\_player->vdecode\_thread, NULL, video\_decode\_thread\_proc, mainplayer->front\_player);

4. static void\* video\_decode\_thread\_proc(void \*param)

5. void render\_video(void \*hrender, AVFrame \*video)

6. render\_setup\_srcrect(render, &lockedpic, &srcpic);

7. static void render\_setup\_srcrect(RENDER \*render, AVFrame \*video, AVFrame \*srcpic)

static void render\_setup\_srcrect(RENDER \*render, AVFrame \*video, AVFrame \*srcpic)

- Parameters:

- `render`: A pointer to a `RENDER` struct that contains information about the video rendering configuration.

- `video`: A pointer to an `AVFrame` struct that contains information about the video frame that will be rendered.

- `srcpic`: A pointer to an `AVFrame` struct that will be used for rendering the video frame.

- Purpose: This function sets up the `srcpic` frame by copying over information from the `video` frame and modifying some of the data pointers and linesize values based on the cropping that has been defined in the `render` struct. In this function, the data pointers for the Y, U, and V components of the `srcpic` frame are adjusted based on the cropping that is defined in the `render` struct. Specifically, the data pointer for the Y component is shifted to the correct starting location based on the top and left crop values, while the data pointers for the U and V components are adjusted to account for the fact that they are half the width and height of the Y component. This allows the `srcpic` frame to represent a cropped version of the original video frame in YUV 4:2:0 format.

- Return value: This function does not have a return value.

###### Volume control function

Ex)

int volume;

player\_getparam(m\_ffPlayer, PARAM\_AUDIO\_VOLUME, &volume);

Volume = volume\_value;

player\_setparam(m\_ffPlayer, PARAM\_AUDIO\_VOLUME, &volume);

void player\_setparam(void \*hplayer, int id, void \*param)

- Parameters:

`hplayer` - Pointer to the player object returned by player\_open

`id` - the parameter id

`param` - the parameter pointer

- Purpose: set player parameters

- Return value: This function does not have a return value.

###### Audio Sync function – Play back video (separate file) based on front video audio

void\* playBackwordVideo(char\* rearvideo\_path, char\* frontvideo\_path, void\* win, PLAYER\_INIT\_PARAMS\* params)

- Parameters:

`rearvideo\_path` - rear video file path(no audio)

`frontvideo\_path` - front video file path

`win` - The win32 platform passes in the window handle

`params` - player initialization parameters

- Purpose: creates a rear video player object

- Return value: void\* pointer type, pointing to the player object

PS. This functions similar to player\_open function.

###### SEEK function (10 seconds before/after)

Ex) player\_seek(m\_ffPlayer, 5000, SEEK\_STEP\_FORWARD);

void player\_seek(void \*hplayer, int64\_t ms, int type)

- Parameters:

`hplayer` - Pointer to the player object returned by player\_open

`ms` - the specified position in milliseconds

`type` - the specified type, 0 / SEEK\_STEP\_FORWARD / SEEK\_STEP\_BACKWARD

- Purpose: Jump to the specified position

If type is 0, it is a normal seek operation, and ms specifies the seek position, in milliseconds

If type is SEEK\_STEP\_FORWARD, it will step forward ms milliseconds

If type is SEEK\_STEP\_FORWARD, it will step backward ms milliseconds

- Return value: This function does not have a return value.

###### Get FPS function

Ex)

float frame\_rate = 0;

player\_getparam(m\_ffPlayer, PARAM\_FRAMERATE\_VALUE, &frame\_rate);