rxp_player Documentation

Release 0.0.1

roxlu

February 11, 2017

rxp_player is an open source, cross platform C library for playing back .ogg video files that are encoded with theora and vorbis. In the near future we will also add support for Daala and Opus.

Contents:

CHAPTER 1

Getting Started

To compile rxp_player you need to:

- Clone the repository
- Make sure you compiled the dependencies (or use the one we provide)
- Compile the library with the packaged build script

1.1 Building the library

Clone the rxp_player repository from github:

```
git clone git@github.com:roxlu/rxp_player.git
```

1.1.1 Dependencies

rxp_player depends on the follow libraries

- libogg
- libvorbis
- libtheora
- libuv

Furthermore the example that is part of the library, which implements a fully working video player needs a couple of other libraries. We provide precompiled libraries that are tested on Mac/Win/Arch-Linux and will be automatically downloaded upon first build. These libraries are:

- cubeb
- glfw
- glxw
- tinylib

1.1.2 Compiling rxp_player on Mac and Linux

We provide build scripts for both linux, mac and windows. We're using CMake for the build system. By default the CMake file will download the dependencies and necessary test files the first time you execute the scripts. To build execute:

```
cd build
./release.sh
```

This will automatically download a test video and starts the test application. It will also build a *librxp_player.a* file and copies it to the root *install* directory for you compiler and operation system.

1.1.3 Compiling on Windows

@TODO

1.1.4 Libraries to link with on Mac

When you want to link with *librxpplayer* in your application you need to link with the following frameworks and libraries on mac.

- CoreFoundation
- Cocoa
- OpenGL
- IOKit
- CoreVideo
- AudioUnit
- CoreAudio
- AudioToolbox

CHAPTER 2

Programmers Guide

On this page we will describe everything you need to know to create a fully working video player with rxp_player with audio and video output. For the output parts we refer to the glfw example that you can find in *src/examples*.

The simplest video player is one that does not have audio output. Therefore we will start with this first and in the next section we describe how you can add audio output too. Although the API is similar for both video with and without audio, for the user there are some big differences.

A global flow of how to use the player is:

- initialize a player with `rxp_player_init()`
- open a file using `rxp_player_open()`
- start playback with `rxp_player_play()`
- call `rxp_player_update()` repeatedly in your draw loop.
- release all used memory when you receive the *RXP_PLAYER_EVENT_RESET* by calling `rxp_player_clear()` in you *on_event* callback.

Before you can use the player you need to initialize the *rxp_player* context which manages all memory, video, audio etc.. Call `rxp_player_init()` with a pointer to a *rxp_player* struct. All functions of the *rxp_player* library return zero on sucess, < 0 on error, so make sure to check this.

After you've initialized the struct you can open a file by calling `rxp_player_open()` and pass the context struct and the filepath to the .ogg file you want to play.

To start playing call `rxp_player_play()`. But by only calling these functions you're not yet there. You need to tell the *rxp_player* that you want to receive video frames. For this we use a callback, that should accept a pointer to the player and a *rxp_packet*.

The *rxp_packet* holds all the information you need to display a frame. Every time this function is called it means you need to update your screen with the recevied buffers. The *rxp_packet* has a member *img[3]* that contains the *width*, *height*, *stride* and video *data* for each of the video planes. At the time of writing we only support YUV420P video data.

The function below shows a simple example of this:

```
// player is `rxp_player player`
static int setup_player() {

   if (rxp_player_init(&player) < 0) {
     printf("+ Error: cannot init player.\n");
     return -1;
   }
```

```
  if (rxp_player_open(&player, "bunny.ogg") < 0) {
    printf("+ Error: cannot open the ogg file.\n");
    return -2;
  }

  if (rxp_player_play(&player) < 0) {
    printf("+ Error: failed to start playing.\n");
    return -3;
  }

  player.on_video_frame = on_video_frame;
  player.on_event = on_event;

  return 0;
}
```

An important aspect you need to implement is the *on_event* callback where you clear all used memory when the player has finished playing and decoding all frames. The *on_event* function will be called when certain player or decoder events happen. These events are:

RXP_PLAYER_DEC_EVENT_AUDIO_INFO: The decoder decoded some audio frames and the players' members nchannels and samplerate have been set.

RXP_PLAYER_EVENT_PLAY: The scheduler/player has opened the file and decoded the first couple of frames/seconds and the player is ready to start running. This is when you should start the audio stream when the .ogg file has audio samples. You can check this by testing the number of channels, which should be > 0, when the .ogg file has an audio stream.

RXP_PLAYER_EVENT_RESET: Whenever you receive the RXP_PLAYER_EVENT_RESET event it's time to tear down the player and stop the audio stream if it was running. Call `rxp_player_clear()` when you receive this event. This event is fired when either you asked the player to stop by using `rxp_player_stop()` or simply when we're ready decoding video frames or when the audio buffer hasn't got any new samples that can be played.

The function below shows an example that implements an event handler, which also start an audio stream using the **'cubeb'_** library. Note how we clear the used memory where we receive the *RXP_PLAYER_EVENT_RESET* event. When you don't call `rxp_player_clear()` memory will leak.

```
static void on_event(rxp_player* p, int event) {

  if (event == RXP_DEC_EVENT_AUDIO_INFO) {
    printf("+ Received RXP_DEC_EVENT_AUDIO_INFO event.\n");
  }
  else if (event == RXP_PLAYER_EVENT_PLAY) {
    printf("+ Received RXP_PLAYER_EVENT_PLAY event.\n");
    if (p->nchannels > 0) {
      start_audio();
    }
  }
  else if (event == RXP_PLAYER_EVENT_RESET) {
    printf("+ Received RXP_PLAYER_EVENT_RESET event.\n");

    if (rxp_player_clear(p) < 0) {
      printf("+ Failed clearing the player.\n");
    }

    /* check if this is a repeated call to start the audio stream */
    if (audio_ctx) {
      cubeb_stream_stop(audio_stream);
      cubeb_stream_destroy(audio_stream);
      cubeb_destroy(audio_ctx);
      audio_ctx = NULL;
      audio_stream = NULL;
      printf("+ Cleaned up the audio stream.\n");
    }
  }
}
```

CHAPTER 3

API Reference

rxp_player_init (*rxp_player* player*)

Initialize the player and all it's members. Internally this will create a queue for the video plackets, sets up the ringbuffer for audio to default values (won't allocate any bytes for the ringbuffer here), initializes the decoder, scheduler clock etc.. This must be called before you make any other call on the rxp_player, and everytime where you called *rxp_player_clear()*.

Parameters **rxp_player*** – Pointer to the rxp_player

Returns 0 on success, < 0 on error.

rxp_player_clear (*rxp_player* player*)

This function clears all used memory of the *rxp_player*. This function will will deallocate the packet queue, deallocate the scheduler, decoder, clock etc.. After calling *rxp_player_clear()* you can call *rxp_player_init()* again if you want to reload or replay the file.

Parameters **rxp_player*** – Pointer to the rxp_player

Returns 0 on success, < 0 on error.

rxp_player_open (*rxp_player* player*, *char* file*)

This will open the given .ogg file. Make sure that you've called *rxp_player_init()* before calling this function. Also, when you want to re-open the same file after it has been played completely and you already called *rxp_player_clear()* to free internally used memory, you need to call *rxp_player_init()* before calling this function again.

Parameters **rxp_player*** – Pointer to the rxp_player

Returns 0 on success, < 0 on error.

```
// somewhere globally ....
rxp_player player;

// opening a file
{
  if (rxp_player_init(&player) < 0) {
    exit(1);
  }

  if (rxp_player_open(&player, "bunny.ogg") < 0) {
    exit(1);
  }

  if (rxp_player_play(&player) < 0) {
    exit(1);
  }
```

```
      // set callbacks
      player.on_event = on_event
      player.on_video_frame = on_video_frame
    }
```

`rxp_player_play` (*rxp_player* player*)

Start playing the opened file. Make sure that you've called `rxp_player_init()`, `rxp_player_open()` first.

> **Parameters** **`rxp_player*`** – Pointer to the rxp_player
>
> **Returns** 0 on success, < 0 on error.

`rxp_player_update` (*rxp_player* player*)

Make sure to call this function as often as possible as it will check if you need to display a new video frame. And it will make sure that the internally used scheduler will be updated as well so it will continue decoding as needed.

> **Parameters** **`rxp_player*`** – Pointer to the rxp_player
>
> **Returns** 0 on success, < 0 on error.

`rxp_player_pause` (*rxp_player* player*)

Pause the playback. This will change the state of the player and the `rxp_player_update()` will not handle any frames/timings until we continue playing again. To continue playback, call `rxp_player_play()` again.

> **Parameters** **`rxp_player*`** – Pointer to the rxp_player
>
> **Returns** 0 on success, < 0 on error.

`rxp_player_stop` (*rxp_player* player*)

Stop the currently being played player. This will stop everything completely and you'll need to re-initialize the player again if you want to start playing again. This will trigger the *RXP_PLAYER_EVENT_RESET* from where you call `rxp_player_clear()` as described in the programmers guide.

> **Parameters** **`rxp_player*`** – Pointer to the rxp_player
>
> **Returns** 0 on success, < 0 on error.

`rxp_player_is_playing` (*rxp_player* player*)

Check if the video player is playing.

> **Parameters** **`rxp_player*`** – Pointer to the rxp_player
>
> **Returns** 0 when the player is playing, else 1, < 0 on error.

`rxp_player_is_paused` (*rxp_player* player*)

Check if the video player is paused.

> **Parameters** **`rxp_player*`** – Pointer to the rxp_player
>
> **Returns** 0 when the player is paused, else 1, < 0 on error.

Index