原 最简单的视音频播放示例9：SDL2播放PCM

2014年10月31日 00:23:11 阅读数：24021

==========

最简单的视音频播放示例系列文章列表：

最简单的视音频播放示例1：总述

最简单的视音频播放示例2：GDI播放YUV, RGB

最简单的视音频播放示例3：Direct3D播放YUV，RGB（通过Surface）

最简单的视音频播放示例4：Direct3D播放RGB（通过Texture）

最简单的视音频播放示例5：OpenGL播放RGB/YUV

最简单的视音频播放示例6：OpenGL播放YUV420P（通过Texture，使用Shader）

最简单的视音频播放示例7：SDL2播放RGB/YUV

最简单的视音频播放示例8：DirectSound播放PCM

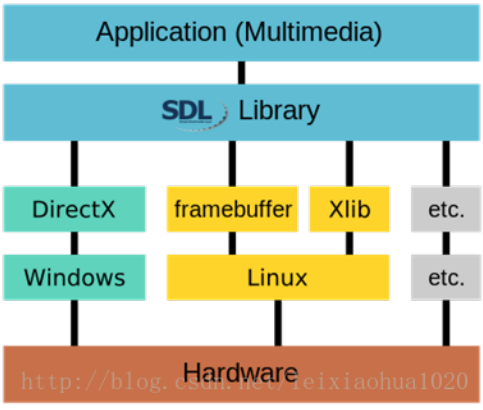
最简单的视音频播放示例9：SDL2播放PCM

==========

本文记录SDL播放音频的技术。在这里使用的版本是SDL2。实际上SDL本身并不提供视音频播放的功能，它只是封装了视音频播放的底层API。在Windows平台下，SDL封装了Direct3D这类的API用于播放视频；封装了DirectSound这类的API用于播放音频。因为SDL的编写目的就是简化视音频播放的开发难度，所以使用SDL播放视频（YUV/RGB）和音频（PCM）数据非常的容易。

SDL简介

SDL(Simple DirectMedia Layer)是一套开放源代码的跨平台多媒体开发库，使用C语言写成。SDL提供了数种控制图像、声音、输出入的函数，让开发者只要用相同或是相似的代码就可以开发出跨多个平台(Linux、Windows、Mac OS X等)的应用软件。目前SDL多用于开发游戏、模拟器、媒体播放器等多媒体应用领域。用下面这张图可以很明确地说明SDL的用途。



SDL实际上并不限于视音频的播放，它将功能分成下列数个子系统（subsystem）：

Video（图像）：图像控制以及线程（thread）和事件管理（event）。

Audio（声音）：声音控制

Joystick（摇杆）：游戏摇杆控制

CD-ROM（光盘驱动器）：光盘媒体控制

Window Management（视窗管理）：与视窗程序设计集成

Event（事件驱动）：处理事件驱动

在Windows下，SDL与DirectX的对应关系如下。

SDL	DirectX
SDL_Video、SDL_Image	DirectDraw、Direct3D
SDL_Audio、SDL_Mixer	DirectSound
SDL_Joystick、SDL_Base	DirectInput
SDL_Net	DirectPlay

注：上文内容在《使用SDL播放视频》的文章中已经介绍，这里再次重复贴一遍。

SDL播放音频的流程

SDL播放音频的流程狠简单，分为以下步骤。

1. **初始化**
   1) 初始化SDL。
   2) 根据参数（SDL_AudioSpec）打开音频设备
2. **循环播放数据**
   1) 播放音频数据。
   2) 延时等待播放完成。

下面详细分析一下上文流程。

1. 初始化

1) 初始化SDL。

使用SDL_Init()初始化SDL。该函数可以确定希望激活的子系统。SDL_Init()函数原型如下：

```cpp
int SDLCALL SDL_Init(Uint32 flags)
```

其中，flags可以取下列值：

SDL_INIT_TIMER：定时器
SDL_INIT_AUDIO：音频
SDL_INIT_VIDEO：视频
SDL_INIT_JOYSTICK：摇杆
SDL_INIT_HAPTIC：触摸屏
SDL_INIT_GAMECONTROLLER：游戏控制器
SDL_INIT_EVENTS：事件
SDL_INIT_NOPARACHUTE：不捕获关键信号（这个不理解）
SDL_INIT_EVERYTHING：包含上述所有选项

有关SDL_Init()有一点需要注意：初始化的时候尽量做到“够用就好”，而不要用SDL_INIT_EVERYTHING。因为有些情况下使用SDL_INIT_EVERYTHING会出现一些不可预知的问题。例如，在MFC应用程序中播放纯音频，如果初始化SDL的时候使用SDL_INIT_EVERYTHING，那么就会出现听不到声音的情况。后来发现，去掉了SDL_INIT_VIDEO之后，问题才得以解决。

2)

根据参数（SDL_AudioSpec）打开音频设备

使用SDL_OpenAudio()打开音频设备。该函数需要传入一个SDL_AudioSpec的结构体。DL_OpenAudio()的原型如下。

```cpp
int SDLCALL SDL_OpenAudio(SDL_AudioSpec * desired,
                                          SDL_AudioSpec * obtained);
```

它的参数是两个SDL_AudioSpec结构体，它们的含义：
desired：期望的参数。
obtained：实际音频设备的参数，一般情况下设置为NULL即可。

SDL_AudioSpec结构体的定义如下。

```cpp
typedef struct SDL_AudioSpec
{
    int freq;                   /**< DSP frequency -- samples per second */
    SDL_AudioFormat format;     /**< Audio data format */
    Uint8 channels;             /**< Number of channels: 1 mono, 2 stereo */
    Uint8 silence;              /**< Audio buffer silence value (calculated) */
    Uint16 samples;             /**< Audio buffer size in samples (power of 2) */
    Uint16 padding;             /**< Necessary for some compile environments */
    Uint32 size;                /**< Audio buffer size in bytes (calculated) */
    SDL_AudioCallback callback;
    void *userdata;
} SDL_AudioSpec;
```

其中包含了关于音频各种参数：
freq：音频数据的采样率。常用的有48000,44100等。
format：音频数据的格式。举例几种格式：

AUDIO_U16SYS：Unsigned 16-bit samples

AUDIO_S16SYS：Signed 16-bit samples

AUDIO_S32SYS：32-bit integer samples

AUDIO_F32SYS：32-bit floating point samples

channels：声道数。例如单声道取值为1，立体声取值为2。
silence：设置静音的值。
samples：音频缓冲区中的采样个数，要求必须是2的n次方。
padding：考虑到兼容性的一个参数。
size：音频缓冲区的大小，以字节为单位。
callback：填充音频缓冲区的回调函数。
userdata：用户自定义的数据。
在这里记录一下填充音频缓冲区的回调函数的作用。当音频设备需要更多数据的时候会调用该回调函数。回调函数的格式要求如下。

```cpp
void (SDLCALL * SDL_AudioCallback) (void *userdata, Uint8 * stream,
                                    int len);
```

回调函数的参数含义如下所示。
userdata：SDL_AudioSpec结构中的用户自定义数据，一般情况下可以不用。
stream：该指针指向需要填充的音频缓冲区。
len：音频缓冲区的大小（以字节为单位）。
在回调函数中可以使用SDL_MixAudio()完成混音等工作。众所周知SDL2和SDL1.x关于视频方面的API差别很大。但是SDL2和SDL1.x关于音频方面的API是一模一样的。唯独在回调函数中，SDL2有一个地方和SDL1.x不一样：SDL2中必须首先使用SDL_memset()将stream中的数据设置为0。

2.

循环播放数据

1)

播放音频数据。

使用SDL_PauseAudio()可以播放音频数据。SDL_PauseAudio()的原型如下。

```cpp
void SDLCALL SDL_PauseAudio(int pause_on)
```

当pause_on设置为0的时候即可开始播放音频数据。设置为1的时候，将会播放静音的值。

2)

延时等待播放完成。

这一步就是延时等待音频播放完毕了。使用像SDL_Delay()这样的延时函数即可。

代码

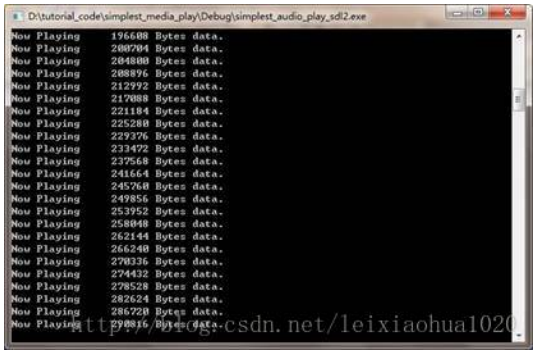
源代码如下所示。

```cpp
/**
 * 最简单的SDL2播放音频的例子（SDL2播放PCM）
 * Simplest Audio Play SDL2 (SDL2 play PCM)
 *
 * 雷霄骅 Lei Xiaohua
 * leixiaohua1020@126.com
 * 中国传媒大学/数字电视技术
 * Communication University of China / Digital TV Technology
 * http://blog.csdn.net/leixiaohua1020
 *
 * 本程序使用SDL2播放PCM音频采样数据。SDL实际上是对底层绘图
 * API（Direct3D，OpenGL）的封装，使用起来明显简单于直接调用底层
 * API。
 *
 * 函数调用步骤如下：
 *
 * [初始化]
 * SDL_Init(): 初始化SDL。
 * SDL_OpenAudio(): 根据参数（存储于SDL_AudioSpec）打开音频设备。
 * SDL_PauseAudio(): 播放音频数据。
 *
 * [循环播放数据]
 * SDL_Delay(): 延时等待播放完成。
 *
 * This software plays PCM raw audio data using SDL2.
 * SDL is a wrapper of low-level API (DirectSound).
 * Use SDL is much easier than directly call these low-level API.
 *
 * The process is shown as follows:
 *
 * [Init]
 * SDL_Init(): Init SDL.
 * SDL_OpenAudio(): Opens the audio device with the desired
 *                  parameters (In SDL_AudioSpec).
 * SDL_PauseAudio(): Play Audio.
 *
 * [Loop to play data]
 * SDL_Delay(): Wait for completetion of playback.
 */

#include <stdio.h>
#include <tchar.h>

extern "C"
{
#include "sdl/SDL.h"
};

//Buffer:
//|-----------|-------------
//chunk-------pos---len-----|
static  Uint8  *audio_chunk;
static  Uint32  audio_len;
static  Uint8  *audio_pos;

/* Audio Callback
 * The audio function callback takes the following parameters:
 * stream: A pointer to the audio buffer to be filled
 * len: The length (in bytes) of the audio buffer
 *
*/
void  fill_audio(void *udata,Uint8 *stream,int len){
    //SDL 2.0
    SDL_memset(stream, 0, len);
    if(audio_len==0)
            return;
    len=(len>audio_len?audio_len:len);

    SDL_MixAudio(stream,audio_pos,len,SDL_MIX_MAXVOLUME);
    audio_pos += len;
    audio_len -= len;
}

int main(int argc, char* argv[])
{
    //Init
    if(SDL_Init(SDL_INIT_AUDIO | SDL_INIT_TIMER)) {
        printf( "Could not initialize SDL - %s\n", SDL_GetError());
        return -1;
    }
    //SDL_AudioSpec
```

```
    SDL_AudioSpec wanted_spec;
    wanted_spec.freq = 44100;
    wanted_spec.format = AUDIO_S16SYS;
    wanted_spec.channels = 2;
    wanted_spec.silence = 0;
    wanted_spec.samples = 1024;
    wanted_spec.callback = fill_audio;

    if (SDL_OpenAudio(&wanted_spec, NULL)<0){
        printf("can't open audio.\n");
        return -1;
    }

    FILE *fp=fopen("../NocturneNo2inEflat_44.1k_s16le.pcm","rb+");
    if(fp==NULL){
        printf("cannot open this file\n");
        return -1;
    }
    int pcm_buffer_size=4096;
    char *pcm_buffer=(char *)malloc(pcm_buffer_size);
    int data_count=0;

    //Play
    SDL_PauseAudio(0);

    while(1){
        if (fread(pcm_buffer, 1, pcm_buffer_size, fp) != pcm_buffer_size){
            // Loop
            fseek(fp, 0, SEEK_SET);
            fread(pcm_buffer, 1, pcm_buffer_size, fp);
            data_count=0;
        }
        printf("Now Playing %10d Bytes data.\n",data_count);
        data_count+=pcm_buffer_size;
        //Set audio buffer (PCM data)
        audio_chunk = (Uint8 *) pcm_buffer;
        //Audio buffer length
        audio_len =pcm_buffer_size;
        audio_pos = audio_chunk;

        while(audio_len>0)//Wait until finish
            SDL_Delay(1);
    }
    free(pcm_buffer);
    SDL_Quit();

    return 0;
}
```

运行结果

运行的结果如下图所示。运行的时候可以听见音乐播放的声音。



下载

代码位于“Simplest Media Play”中

SourceForge项目地址： https://sourceforge.net/projects/simplestmediaplay/

CSDN下载地址： http://download.csdn.net/detail/leixiaohua1020/8054395

注：

该项目会不定时的更新并修复一些小问题，最新的版本请参考该系列文章的总述页面：

《最简单的视音频播放示例1：总述》

上述工程包含了使用各种API（Direct3D，OpenGL，GDI，DirectSound，SDL2）播放多媒体例子。其中音频输入为PCM采样数据。输出至系统的声卡播放出来。视频输入为YUV/RGB像素数据。输出至显示器上的一个窗口播放出来。
通过本工程的代码初学者可以快速学习使用这几个API播放视频和音频的技术。
一共包括了如下几个子工程：
simplest_audio_play_directsound:
使用DirectSound播放PCM音频采样数据。
simplest_audio_play_sdl2:
使用SDL2播放PCM音频采样数据。
simplest_video_play_direct3d:
使用Direct3D的Surface播放RGB/YUV视频像素数据。
simplest_video_play_direct3d_texture:使用Direct3D的Texture播放RGB视频像素数据。
simplest_video_play_gdi:
使用GDI播放RGB/YUV视频像素数据。
simplest_video_play_opengl:
使用OpenGL播放RGB/YUV视频像素数据。
simplest_video_play_opengl_texture:
使用OpenGL的Texture播放YUV视频像素数据。
simplest_video_play_sdl2:
使用SDL2播放RGB/YUV视频像素数据。

文章标签： SDL PCM 音频 播放

个人分类： 我的开源项目 SDL