原 SDL2源代码分析2：窗口（SDL_Window）

2014年11月03日 00:31:11　阅读数：24251

==========

SDL源代码分析系列文章列表：

SDL2源代码分析1：初始化（SDL_Init()）

SDL2源代码分析2：窗口（SDL_Window）

SDL2源代码分析3：渲染器（SDL_Renderer）

SDL2源代码分析4：纹理（SDL_Texture）

SDL2源代码分析5：更新纹理（SDL_UpdateTexture()）

SDL2源代码分析6：复制到渲染器（SDL_RenderCopy()）

SDL2源代码分析7：显示（SDL_RenderPresent()）

SDL2源代码分析8：视频显示总结

==========

上一篇文章分析了SDL的初始化函数SDL_Init()。这篇文章继续分析SDL的源代码。本文分析SDL的窗口（SDL_Window）。

SDL播放视频的代码流程如下所示。

初始化:

SDL_Init(): 初始化SDL。

SDL_CreateWindow(): 创建窗口（Window）。

SDL_CreateRenderer(): 基于窗口创建渲染器（Render）。

SDL_CreateTexture(): 创建纹理（Texture）。

循环渲染数据:

SDL_UpdateTexture(): 设置纹理的数据。

SDL_RenderCopy(): 纹理复制给渲染器。
SDL_RenderPresent(): 显示。

上篇文章分析了该流程中的第一个函数 **SDL_Init()** 。本文继续分析该流程中的第二个函数 **SDL_CreateWindow()** 。

SDL_Window

SDL_Window结构体定义了一个SDL2中的窗口。如果直接使用SDL2编译好的SDK的话，是看不到它的内部结构的。有关它的定义在头文件中只有一行代码，但是这一行定义前面的注释非常之多，如下所示：

```cpp
/**
 *  \brief The type used to identify a window
 *
 *  \sa SDL_CreateWindow()
 *  \sa SDL_CreateWindowFrom()
 *  \sa SDL_DestroyWindow()
 *  \sa SDL_GetWindowData()
 *  \sa SDL_GetWindowFlags()
 *  \sa SDL_GetWindowGrab()
 *  \sa SDL_GetWindowPosition()
 *  \sa SDL_GetWindowSize()
 *  \sa SDL_GetWindowTitle()
 *  \sa SDL_HideWindow()
 *  \sa SDL_MaximizeWindow()
 *  \sa SDL_MinimizeWindow()
 *  \sa SDL_RaiseWindow()
 *  \sa SDL_RestoreWindow()
 *  \sa SDL_SetWindowData()
 *  \sa SDL_SetWindowFullscreen()
 *  \sa SDL_SetWindowGrab()
 *  \sa SDL_SetWindowIcon()
 *  \sa SDL_SetWindowPosition()
 *  \sa SDL_SetWindowSize()
 *  \sa SDL_SetWindowBordered()
 *  \sa SDL_SetWindowTitle()
 *  \sa SDL_ShowWindow()
 */
typedef struct SDL_Window SDL_Window;
```

在源代码工程中可以看到它的定义，位于video\SDL_sysvideo.h文件中。它的定义如下。

```cpp
/* Define the SDL window structure, corresponding to toplevel windows */
struct SDL_Window
{
    const void *magic;
    Uint32 id;
    char *title;
    SDL_Surface *icon;
    int x, y;
    int w, h;
    int min_w, min_h;
    int max_w, max_h;
    Uint32 flags;
    Uint32 last_fullscreen_flags;


    /* Stored position and size for windowed mode */
    SDL_Rect windowed;


    SDL_DisplayMode fullscreen_mode;


    float brightness;
    Uint16 *gamma;
    Uint16 *saved_gamma;        /* (just offset into gamma) */


    SDL_Surface *surface;
    SDL_bool surface_valid;


    SDL_bool is_destroying;


    SDL_WindowShaper *shaper;


    SDL_WindowUserData *data;


    void *driverdata;


    SDL_Window *prev;
    SDL_Window *next;
};
```

可以看出其中包含了一个“窗口”应该包含的各种属性。这个结构体中的各个变量还没有深入研究，暂不详细分析。下面来看看如何创建这个SDL_Window。

SDL_CreateWindow()

函数简介

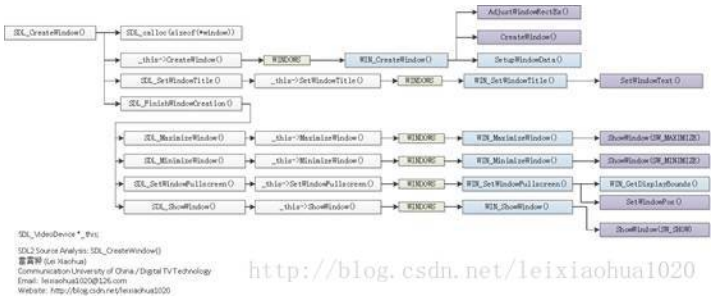
SDL_CreateWindow()用于创建一个视频播放的窗口。SDL_CreateWindow()的原型如下。

```cpp
SDL_Window * SDLCALL SDL_CreateWindow(const char *title,
                                      int x, int y, int w,
                                      int h, Uint32 flags);
```

参数含义如下。
title
：窗口标题
x
：窗口位置x坐标。也可以设置为SDL_WINDOWPOS_CENTERED或SDL_WINDOWPOS_UNDEFINED。
y
：窗口位置y坐标。同上。
w
：窗口的宽
h
：窗口的高
flags ：支持下列标识。包括了窗口的是否最大化、最小化，能否调整边界等等属性。
::SDL_WINDOW_FULLSCREEN, ::SDL_WINDOW_OPENGL,
::SDL_WINDOW_HIDDEN, ::SDL_WINDOW_BORDERLESS,
::SDL_WINDOW_RESIZABLE, ::SDL_WINDOW_MAXIMIZED,
::SDL_WINDOW_MINIMIZED, ::SDL_WINDOW_INPUT_GRABBED,
::SDL_WINDOW_ALLOW_HIGHDPI.
返回创建完成的窗口的ID。如果创建失败则返回0。

函数调用关系图

SDL_ CreateWindow ()关键函数的调用关系可以用下图表示。



上面的函数调用关系图本来是一张高清大图，但是由于博客对图片尺寸有限制，因而显得不太清晰。相册里面上传了一份原始的大图片：

http://my.csdn.net/leixiaohua1020/album/detail/1793195

打开上述相册里面的图片，右键选择“另存为”即可保存原始图片。

源代码分析

SDL_CreateWindow()的源代码位于video\SDL_video.c中，如下所示。

```cpp
SDL_Window * SDL_CreateWindow(const char *title, int x, int y, int w, int h, Uint32 flags)
{
    SDL_Window *window;
    const char *hint;


    if (!_this) {
        /* Initialize the video system if needed */
        if (SDL_VideoInit(NULL) < 0) {
            return NULL;
        }
    }

```

```
14.
15.     /* Some platforms can't create zero-sized windows */
16.     if (w < 1) {
17.         w = 1;
18.     }
19.     if (h < 1) {
20.         h = 1;
21.     }
22.
23.
24.     /* Some platforms have OpenGL enabled by default */
25. #if (SDL_VIDEO_OPENGL && __MACOSX__) || __IPHONEOS__ || __ANDROID__
26.     flags |= SDL_WINDOW_OPENGL;
27. #endif
28.     if (flags & SDL_WINDOW_OPENGL) {
29.         if (!_this->GL_CreateContext) {
30.             SDL_SetError("No OpenGL support in video driver");
31.             return NULL;
32.         }
33.         if (SDL_GL_LoadLibrary(NULL) < 0) {
34.             return NULL;
35.         }
36.     }
37.
38.
39.     /* Unless the user has specified the high-DPI disabling hint, respect the
40.      * SDL_WINDOW_ALLOW_HIGHDPI flag.
41.      */
42.     if (flags & SDL_WINDOW_ALLOW_HIGHDPI) {
43.         hint = SDL_GetHint(SDL_HINT_VIDEO_HIGHDPI_DISABLED);
44.         if (hint && SDL_atoi(hint) > 0) {
45.             flags &= ~SDL_WINDOW_ALLOW_HIGHDPI;
46.         }
47.     }
48.
49.
50.     window = (SDL_Window *)SDL_calloc(1, sizeof(*window));
51.     if (!window) {
52.         SDL_OutOfMemory();
53.         return NULL;
54.     }
55.     window->magic = &_this->window_magic;
56.     window->id = _this->next_object_id++;
57.     window->x = x;
58.     window->y = y;
59.     window->w = w;
60.     window->h = h;
61.     if (SDL_WINDOWPOS_ISUNDEFINED(x) || SDL_WINDOWPOS_ISUNDEFINED(y) ||
62.         SDL_WINDOWPOS_ISCENTERED(x) || SDL_WINDOWPOS_ISCENTERED(y)) {
63.         SDL_VideoDisplay *display = SDL_GetDisplayForWindow(window);
64.         int displayIndex;
65.         SDL_Rect bounds;
66.
67.
68.         displayIndex = SDL_GetIndexOfDisplay(display);
69.         SDL_GetDisplayBounds(displayIndex, &bounds);
70.         if (SDL_WINDOWPOS_ISUNDEFINED(x) || SDL_WINDOWPOS_ISCENTERED(x)) {
71.             window->x = bounds.x + (bounds.w - w) / 2;
72.         }
73.         if (SDL_WINDOWPOS_ISUNDEFINED(y) || SDL_WINDOWPOS_ISCENTERED(y)) {
74.             window->y = bounds.y + (bounds.h - h) / 2;
75.         }
76.     }
77.     window->flags = ((flags & CREATE_FLAGS) | SDL_WINDOW_HIDDEN);
78.     window->last_fullscreen_flags = window->flags;
79.     window->brightness = 1.0f;
80.     window->next = _this->windows;
81.     window->is_destroying = SDL_FALSE;
82.
83.
84.     if (_this->windows) {
85.         _this->windows->prev = window;
86.     }
87.     _this->windows = window;
88.
89.
90.     if (_this->CreateWindow && _this->CreateWindow(_this, window) < 0) {
91.         SDL_DestroyWindow(window);
92.         return NULL;
93.     }
94.
95.
96.     if (title) {
97.         SDL_SetWindowTitle(window, title);
98.     }
99.     SDL_FinishWindowCreation(window, flags);
100.
101.
102.     /* If the window was created fullscreen, make sure the mode code matches */
103.     SDL_UpdateFullscreenMode(window, FULLSCREEN_VISIBLE(window));
104.
```

```
105.
106.     return window;
107. }
```

下面总结一下SDL_CreateWindow()的大致流程。

1.

一些为了保证各个平台的兼容性的初始化工作。 各个平台创建窗口的条件不同。例如，某些平台不支持创建大小为0的窗口。再例如，某些平台默认开启OpenGL。

2.

调用SDL_calloc()为SDL_Window结构体分配一块内存。 同时设置一些基本属性，例如窗口的宽高，位置等等。

PS：上篇文章中已经提过，在这里重复一下SDL中内存分配函数的知识。在SDL中分配内存使用SDL_malloc()，SDL_calloc()，这些函数实际上就是malloc()，calloc()。它们的定义位于stdlib\SDL_malloc.c文件中。如下所示：

```cpp
#define memset  SDL_memset
#define memcpy  SDL_memcpy
#define malloc  SDL_malloc
#define calloc  SDL_calloc
#define realloc SDL_realloc
#define free    SDL_free
```

3.

调用VideoDevice的CreateWindow()方法创建窗口。 这是创建窗口这个函数中最关键的一环。在这里有一点需要注意，SDL中有一个SDL_VideoDevice类型的静态全局变量_this。SDL调用视频驱动的功能都是通过调用该指针完成的。定义如下。

```cpp
static SDL_VideoDevice *_this = NULL;
```

该_this变量代表了当前视频驱动设备。该变量在SDL_Init()中被赋值。如果是Windows下使用，则会被赋值为“Windows视频驱动”；Android下使用，则会被赋值为“Android视频驱动”。这是上篇文章中的内容，不再重复记录。

下面我们以“Windows视频驱动”为例，看看CreateWindow()都会执行哪些函数。

首先回顾一下上篇文章中的一个知识。从上一篇文章的SDL_Init()函数的分析中我们可以得知，Windows视频驱动初始化的时候会给SDL_VideoDevice一系列的函数指针赋值，如下所示。

```cpp
static SDL_VideoDevice *WIN_CreateDevice(int devindex)
{
    SDL_VideoDevice *device;
    SDL_VideoData *data;


    SDL_RegisterApp(NULL, 0, NULL);


    /* Initialize all variables that we clean on shutdown */
    device = (SDL_VideoDevice *) SDL_calloc(1, sizeof(SDL_VideoDevice));
    if (device) {
        data = (struct SDL_VideoData *) SDL_calloc(1, sizeof(SDL_VideoData));
    } else {
        data = NULL;
    }
    if (!data) {
        SDL_free(device);
        SDL_OutOfMemory();
        return NULL;
    }
    device->driverdata = data;


    data->userDLL = SDL_LoadObject("USER32.DLL");
    if (data->userDLL) {
        data->CloseTouchInputHandle = (BOOL (WINAPI *)( HTOUCHINPUT )) SDL_LoadFunction(data->userDLL, "CloseTouchInputHandle");
        data->GetTouchInputInfo = (BOOL (WINAPI *)( HTOUCHINPUT, UINT, PTOUCHINPUT, int )) SDL_LoadFunction(data->userDLL, "GetTouchInputInfo");
        data->RegisterTouchWindow = (BOOL (WINAPI *)( HWND, ULONG )) SDL_LoadFunction(data->userDLL, "RegisterTouchWindow");
    }


    /* Set the function pointers */
    device->VideoInit = WIN_VideoInit;
    device->VideoQuit = WIN_VideoQuit;
    device->GetDisplayBounds = WIN_GetDisplayBounds;
    device->GetDisplayModes = WIN_GetDisplayModes;
```

```
    device->SetDisplayMode = WIN_SetDisplayMode;
    device->PumpEvents = WIN_PumpEvents;


#undef CreateWindow
    device->CreateWindow = WIN_CreateWindow;
    device->CreateWindowFrom = WIN_CreateWindowFrom;
    device->SetWindowTitle = WIN_SetWindowTitle;
    device->SetWindowIcon = WIN_SetWindowIcon;
    device->SetWindowPosition = WIN_SetWindowPosition;
    device->SetWindowSize = WIN_SetWindowSize;
    device->ShowWindow = WIN_ShowWindow;
    device->HideWindow = WIN_HideWindow;
    device->RaiseWindow = WIN_RaiseWindow;
    device->MaximizeWindow = WIN_MaximizeWindow;
    device->MinimizeWindow = WIN_MinimizeWindow;
    device->RestoreWindow = WIN_RestoreWindow;
    device->SetWindowBordered = WIN_SetWindowBordered;
    device->SetWindowFullscreen = WIN_SetWindowFullscreen;
    device->SetWindowGammaRamp = WIN_SetWindowGammaRamp;
    device->GetWindowGammaRamp = WIN_GetWindowGammaRamp;
    device->SetWindowGrab = WIN_SetWindowGrab;
    device->DestroyWindow = WIN_DestroyWindow;
    device->GetWindowWMInfo = WIN_GetWindowWMInfo;
    device->CreateWindowFramebuffer = WIN_CreateWindowFramebuffer;
    device->UpdateWindowFramebuffer = WIN_UpdateWindowFramebuffer;
    device->DestroyWindowFramebuffer = WIN_DestroyWindowFramebuffer;
    device->OnWindowEnter = WIN_OnWindowEnter;


    device->shape_driver.CreateShaper = Win32_CreateShaper;
    device->shape_driver.SetWindowShape = Win32_SetWindowShape;
    device->shape_driver.ResizeWindowShape = Win32_ResizeWindowShape;


#if SDL_VIDEO_OPENGL_WGL
    device->GL_LoadLibrary = WIN_GL_LoadLibrary;
    device->GL_GetProcAddress = WIN_GL_GetProcAddress;
    device->GL_UnloadLibrary = WIN_GL_UnloadLibrary;
    device->GL_CreateContext = WIN_GL_CreateContext;
    device->GL_MakeCurrent = WIN_GL_MakeCurrent;
    device->GL_SetSwapInterval = WIN_GL_SetSwapInterval;
    device->GL_GetSwapInterval = WIN_GL_GetSwapInterval;
    device->GL_SwapWindow = WIN_GL_SwapWindow;
    device->GL_DeleteContext = WIN_GL_DeleteContext;
#endif
    device->StartTextInput = WIN_StartTextInput;
    device->StopTextInput = WIN_StopTextInput;
    device->SetTextInputRect = WIN_SetTextInputRect;


    device->SetClipboardText = WIN_SetClipboardText;
    device->GetClipboardText = WIN_GetClipboardText;
    device->HasClipboardText = WIN_HasClipboardText;


    device->free = WIN_DeleteDevice;


    return device;
}
```

从上文中可以看出，“Windows视频驱动”初始化之后，调用该SDL_VideoDevice的CreateWindow()实际上就等同于调用WIN_CreateWindow()这个函数。因此，我们来看一下WIN_CreateWindow()这个函数的定义（位于video\windows\SDL_windowswindow.c）。

```cpp
int WIN_CreateWindow(_THIS, SDL_Window * window)
{
    HWND hwnd;
    RECT rect;
    DWORD style = STYLE_BASIC;
    int x, y;
    int w, h;


    style |= GetWindowStyle(window);


    /* Figure out what the window area will be */
    rect.left = window->x;
    rect.top = window->y;
    rect.right = window->x + window->w;
    rect.bottom = window->y + window->h;
    AdjustWindowRectEx(&rect, style, FALSE, 0);
    x = rect.left;
    y = rect.top;
    w = (rect.right - rect.left);
    h = (rect.bottom - rect.top);


    hwnd =
        CreateWindow(SDL_Appname, TEXT(""), style, x, y, w, h, NULL, NULL,
                     SDL_Instance, NULL);
    if (!hwnd) {
        return WIN_SetError("Couldn't create window");
    }


    WIN_PumpEvents(_this);


    if (SetupWindowData(_this, window, hwnd, SDL_TRUE) < 0) {
        DestroyWindow(hwnd);
        return -1;
    }


#if SDL_VIDEO_OPENGL_WGL
    /* We need to initialize the extensions before deciding how to create ES profiles */
    if (window->flags & SDL_WINDOW_OPENGL) {
        WIN_GL_InitExtensions(_this);
    }
#endif


#if SDL_VIDEO_OPENGL_ES2
    if ((window->flags & SDL_WINDOW_OPENGL) &&
        _this->gl_config.profile_mask == SDL_GL_CONTEXT_PROFILE_ES
#if SDL_VIDEO_OPENGL_WGL
        && (!_this->gl_data || !_this->gl_data->HAS_WGL_EXT_create_context_es2_profile)
#endif
        ) {
#if SDL_VIDEO_OPENGL_EGL
        if (WIN_GLES_SetupWindow(_this, window) < 0) {
            WIN_DestroyWindow(_this, window);
            return -1;
        }
#else
        return SDL_SetError("Could not create GLES window surface (no EGL support available)");
#endif /* SDL_VIDEO_OPENGL_EGL */
    } else
#endif /* SDL_VIDEO_OPENGL_ES2 */


#if SDL_VIDEO_OPENGL_WGL
    if (window->flags & SDL_WINDOW_OPENGL) {
        if (WIN_GL_SetupWindow(_this, window) < 0) {
            WIN_DestroyWindow(_this, window);
            return -1;
        }
    }
#endif


    return 0;
}
```

从该函数的代码中我们可以看到很多的Win32的API。最核心的函数只有一个，就是CreateWindow()。正是这个Win32的API最终创建了SDL的窗口。当然，为了创建出来的窗口更“优质”，包含了一些初始化的工作，例如AdjustWindowRectEx()；以及一些收尾工作，例如SetupWindowData()（该函数主要用于设置SDL_Window的参数）。

4.

完成一些收尾工作。 例如设置窗口的标题，如果是“全屏模式”则设置全屏显示等等。在这里简单介绍几个函数。

SDL_SetWindowTitle()用于设置窗口的标题，它的定义如下。

```cpp
void SDL_SetWindowTitle(SDL_Window * window, const char *title)
{
    CHECK_WINDOW_MAGIC(window, );


    if (title == window->title) {
        return;
    }
    SDL_free(window->title);
    if (title && *title) {
        window->title = SDL_strdup(title);
    } else {
        window->title = NULL;
    }


    if (_this->SetWindowTitle) {
        _this->SetWindowTitle(_this, window);
    }
}
```

该函数调用了SDL_VideoDevice的SetWindowTitle()。在“Windows视频驱动”中，实际的执行函数是WIN_SetWindowTitle()。该函数的定义如下。

```cpp
void WIN_SetWindowTitle(_THIS, SDL_Window * window)
{
    HWND hwnd = ((SDL_WindowData *) window->driverdata)->hwnd;
    LPTSTR title;


    if (window->title) {
        title = WIN_UTF8ToString(window->title);
    } else {
        title = NULL;
    }
    SetWindowText(hwnd, title ? title : TEXT(""));
    SDL_free(title);
}
```

从代码中可以看出，该函数调用了Win32的API函数SetWindowText()设置窗口的标题。

SDL_FinishWindowCreation()完成一些窗口的收尾工作。该函数的定义如下。

```cpp
static void SDL_FinishWindowCreation(SDL_Window *window, Uint32 flags)
{
    window->windowed.x = window->x;
    window->windowed.y = window->y;
    window->windowed.w = window->w;
    window->windowed.h = window->h;


    if (flags & SDL_WINDOW_MAXIMIZED) {
        SDL_MaximizeWindow(window);
    }
    if (flags & SDL_WINDOW_MINIMIZED) {
        SDL_MinimizeWindow(window);
    }
    if (flags & SDL_WINDOW_FULLSCREEN) {
        SDL_SetWindowFullscreen(window, flags);
    }
    if (flags & SDL_WINDOW_INPUT_GRABBED) {
        SDL_SetWindowGrab(window, SDL_TRUE);
    }
    if (!(flags & SDL_WINDOW_HIDDEN)) {
        SDL_ShowWindow(window);
    }
}
```

从代码中可以看出，如果创建窗口的时候：

指定了“最大化”，则会执行SDL_MaximizeWindow()；

指定了“最小化”，则会执行SDL_MinimizeWindow()；

指定了“全屏”，则会执行SDL_SetWindowFullscreen()；

指定了“抓取”（这个没有试过），则会执行SDL_SetWindowGrab()；

指定了“隐藏”，则会执行SDL_ShowWindow()。

下面分别看一下SDL_MaximizeWindow()，SDL_MinimizeWindow()，SDL_SetWindowFullscreen()，SDL_ShowWindow()的代码。

SDL_MaximizeWindow()定义如下。

```cpp
void SDL_MaximizeWindow(SDL_Window * window)
{
    CHECK_WINDOW_MAGIC(window, );


    if (window->flags & SDL_WINDOW_MAXIMIZED) {
        return;
    }

    /* !!! FIXME: should this check if the window is resizable? */


    if (_this->MaximizeWindow) {
        _this->MaximizeWindow(_this, window);
    }
}
```

从代码中可以看出，SDL_MaximizeWindow()调用了SDL_VideoDevice的MaximizeWindow()函数。在“Windows视频驱动”下，实际上调用了WIN_MaximizeWindow()函数，该函数的定义如下。

```cpp
void WIN_MaximizeWindow(_THIS, SDL_Window * window)
{
    SDL_WindowData *data = (SDL_WindowData *)window->driverdata;
    HWND hwnd = data->hwnd;
    data->expected_resize = TRUE;
    ShowWindow(hwnd, SW_MAXIMIZE);
    data->expected_resize = FALSE;
}
```

从上述代码中可以看出WIN_MaximizeWindow()调用了Win32的API函数ShowWindow()。

SDL_MinimizeWindow()定义如下。

```cpp
void SDL_MinimizeWindow(SDL_Window * window)
{
    CHECK_WINDOW_MAGIC(window, );


    if (window->flags & SDL_WINDOW_MINIMIZED) {
        return;
    }


    SDL_UpdateFullscreenMode(window, SDL_FALSE);


    if (_this->MinimizeWindow) {
        _this->MinimizeWindow(_this, window);
    }
}
```

从代码中可以看出，SDL_MinimizeWindow()调用了SDL_VideoDevice的MinimizeWindow()函数。在“Windows视频驱动”下，实际上调用了WIN_MinimizeWindow()函数，该函数的定义如下。

```cpp
void WIN_MinimizeWindow(_THIS, SDL_Window * window)
{
    HWND hwnd = ((SDL_WindowData *) window->driverdata)->hwnd;
    ShowWindow(hwnd, SW_MINIMIZE);
}
```

从上述代码中可以看出WIN_MinimizeWindow()调用了Win32的API函数ShowWindow()。

SDL_SetWindowFullscreen()定义如下。

```cpp
int SDL_SetWindowFullscreen(SDL_Window * window, Uint32 flags)
{
    CHECK_WINDOW_MAGIC(window, -1);


    flags &= FULLSCREEN_MASK;


    if ( flags == (window->flags & FULLSCREEN_MASK) ) {
        return 0;
    }


    /* clear the previous flags and OR in the new ones */
    window->flags &= ~FULLSCREEN_MASK;
    window->flags |= flags;


    SDL_UpdateFullscreenMode(window, FULLSCREEN_VISIBLE(window));


    return 0;
}
```

从代码中可以看出，SDL_SetWindowFullscreen()调用了SDL_UpdateFullscreenMode()函数，该函数的定义如下。

```cpp
static void SDL_UpdateFullscreenMode(SDL_Window * window, SDL_bool fullscreen)
{
    SDL_VideoDisplay *display;
    SDL_Window *other;


#ifdef __MACOSX__
    if (Cocoa_SetWindowFullscreenSpace(window, fullscreen)) {
        window->last_fullscreen_flags = window->flags;
        return;
    }
#endif


    display = SDL_GetDisplayForWindow(window);


    if (fullscreen) {
        /* Hide any other fullscreen windows */
        if (display->fullscreen_window &&
            display->fullscreen_window != window) {
            SDL_MinimizeWindow(display->fullscreen_window);
        }
    }


    /* See if anything needs to be done now */
    if ((display->fullscreen_window == window) == fullscreen) {
        if ((window->last_fullscreen_flags & FULLSCREEN_MASK) == (window->flags & FULLSCREEN_MASK)) {
            return;
        }
    }


    /* See if there are any fullscreen windows */
    for (other = _this->windows; other; other = other->next) {
        SDL_bool setDisplayMode = SDL_FALSE;


        if (other == window) {
            setDisplayMode = fullscreen;
        } else if (FULLSCREEN_VISIBLE(other) &&
                   SDL_GetDisplayForWindow(other) == display) {
            setDisplayMode = SDL_TRUE;
        }

```

```
        if (setDisplayMode) {
            SDL_DisplayMode fullscreen_mode;


            if (SDL_GetWindowDisplayMode(other, &fullscreen_mode) == 0) {
                SDL_bool resized = SDL_TRUE;


                if (other->w == fullscreen_mode.w && other->h == fullscreen_mode.h) {
                    resized = SDL_FALSE;
                }


                /* only do the mode change if we want exclusive fullscreen */
                if ((window->flags & SDL_WINDOW_FULLSCREEN_DESKTOP) != SDL_WINDOW_FULLSCREEN_DESKTOP) {
                    SDL_SetDisplayModeForDisplay(display, &fullscreen_mode);
                } else {
                    SDL_SetDisplayModeForDisplay(display, NULL);
                }


                if (_this->SetWindowFullscreen) {
                    _this->SetWindowFullscreen(_this, other, display, SDL_TRUE);
                }
                display->fullscreen_window = other;


                /* Generate a mode change event here */
                if (resized) {
                    SDL_SendWindowEvent(other, SDL_WINDOWEVENT_RESIZED,
                                        fullscreen_mode.w, fullscreen_mode.h);
                } else {
                    SDL_OnWindowResized(other);
                }


                SDL_RestoreMousePosition(other);


                window->last_fullscreen_flags = window->flags;
                return;
            }
        }
    }


    /* Nope, restore the desktop mode */
    SDL_SetDisplayModeForDisplay(display, NULL);


    if (_this->SetWindowFullscreen) {
        _this->SetWindowFullscreen(_this, window, display, SDL_FALSE);
    }
    display->fullscreen_window = NULL;


    /* Generate a mode change event here */
    SDL_OnWindowResized(window);


    /* Restore the cursor position */
    SDL_RestoreMousePosition(window);


    window->last_fullscreen_flags = window->flags;
}
```

SDL_UpdateFullscreenMode()代码很长，在这里我们只选择最关键的代码进行分析。SDL_UpdateFullscreenMode()最关键的地方在于它调用了SDL_VideoDevice的SetWindowFullscreen()函数。在“Windows视频驱动”下，实际上调用了WIN_SetWindowFullscreen()函数，该函数的定义如下。

```cpp
void WIN_SetWindowFullscreen(_THIS, SDL_Window * window, SDL_VideoDisplay * display, SDL_bool fullscreen)
{
    SDL_WindowData *data = (SDL_WindowData *) window->driverdata;
    HWND hwnd = data->hwnd;
    RECT rect;
    SDL_Rect bounds;
    DWORD style;
    HWND top;
    BOOL menu;
    int x, y;
    int w, h;


    if (SDL_ShouldAllowTopmost() && (window->flags & (SDL_WINDOW_FULLSCREEN|SDL_WINDOW_INPUT_FOCUS)) == (SDL_WINDOW_FULLSCREEN|SDL_WINDOW_INPUT_FOCUS)) {
        top = HWND_TOPMOST;
    } else {
        top = HWND_NOTOPMOST;
    }


    style = GetWindowLong(hwnd, GWL_STYLE);
    style &= ~STYLE_MASK;
    style |= GetWindowStyle(window);


    WIN_GetDisplayBounds(_this, display, &bounds);


    if (fullscreen) {
        x = bounds.x;
        y = bounds.y;
        w = bounds.w;
        h = bounds.h;
    } else {
        rect.left = 0;
        rect.top = 0;
        rect.right = window->windowed.w;
        rect.bottom = window->windowed.h;
        menu = (style & WS_CHILDWINDOW) ? FALSE : (GetMenu(hwnd) != NULL);
        AdjustWindowRectEx(&rect, style, menu, 0);
        w = (rect.right - rect.left);
        h = (rect.bottom - rect.top);
        x = window->windowed.x + rect.left;
        y = window->windowed.y + rect.top;
    }
    SetWindowLong(hwnd, GWL_STYLE, style);
    data->expected_resize = TRUE;
    SetWindowPos(hwnd, top, x, y, w, h, SWP_NOCOPYBITS | SWP_NOACTIVATE);
    data->expected_resize = FALSE;
}
```

从代码中可以看出，该函数通过WIN_GetDisplayBounds()获得屏幕的尺寸，然后通过SetWindowPos()函数设置全屏窗口的大小和位置。

SDL_ShowWindow()的定义如下。

```cpp
void SDL_ShowWindow(SDL_Window * window)
{
    CHECK_WINDOW_MAGIC(window, );


    if (window->flags & SDL_WINDOW_SHOWN) {
        return;
    }


    if (_this->ShowWindow) {
        _this->ShowWindow(_this, window);
    }
    SDL_SendWindowEvent(window, SDL_WINDOWEVENT_SHOWN, 0, 0);
}
```

SDL_ShowWindow ()调用了SDL_VideoDevice的ShowWindow()函数。在“Windows视频驱动”下，实际上调用了WIN_ShowWindow()函数，该函数的定义如下。

```cpp
void WIN_ShowWindow(_THIS, SDL_Window * window)
{
    HWND hwnd = ((SDL_WindowData *) window->driverdata)->hwnd;
    ShowWindow(hwnd, SW_SHOW);
}
```

该函数比较简单，直接调用了Win32中的ShowWindow()方法。

文章标签： SDL SDL_Window 窗口 源代码 Win32

个人分类： SDL

所属专栏： 开源多媒体项目源代码分析