

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：

```

1.  /**
2.   * \brief The type used to identify a window
3.   *
4.   * \sa SDL_CreateWindow()
5.   * \sa SDL_CreateWindowFrom()
6.   * \sa SDL_DestroyWindow()
7.   * \sa SDL_GetWindowData()
8.   * \sa SDL_GetWindowFlags()
9.   * \sa SDL_GetWindowGrab()
10.  * \sa SDL_GetWindowPosition()
11.  * \sa SDL_GetWindowSize()
12.  * \sa SDL_GetWindowTitle()
13.  * \sa SDL_HideWindow()
14.  * \sa SDL_MaximizeWindow()
15.  * \sa SDL_MinimizeWindow()
16.  * \sa SDL_RaiseWindow()
17.  * \sa SDL_RestoreWindow()
18.  * \sa SDL_SetWindowData()
19.  * \sa SDL_SetWindowFullscreen()
20.  * \sa SDL_SetWindowGrab()
21.  * \sa SDL_SetWindowIcon()
22.  * \sa SDL_SetWindowPosition()
23.  * \sa SDL_SetWindowSize()
24.  * \sa SDL_SetWindowBordered()
25.  * \sa SDL_SetWindowTitle()
26.  * \sa SDL_ShowWindow()
27.  */
28.  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。

```

1.  /* Define the SDL window structure, corresponding to toplevel windows */
2.  struct SDL_Window
3.  {
4.      const void *magic;
5.      Uint32 id;
6.      char *title;
7.      SDL_Surface *icon;
8.      int x, y;
9.      int w, h;
10.     int min_w, min_h;
11.     int max_w, max_h;
12.     Uint32 flags;
13.     Uint32 last_fullscreen_flags;
14.
15.
16.     /* Stored position and size for windowed mode */
17.     SDL_Rect windowed;
18.
19.
20.     SDL_DisplayMode fullscreen_mode;
21.
22.
23.     float brightness;
24.     Uint16 *gamma;
25.     Uint16 *saved_gamma;      /* (just offset into gamma) */
26.
27.
28.     SDL_Surface *surface;
29.     SDL_bool surface_valid;
30.
31.
32.     SDL_bool is_destroying;
33.
34.
35.     SDL_WindowShaper *shaper;
36.
37.
38.     SDL_WindowUserData *data;
39.
40.
41.     void *driverdata;
42.
43.
44.     SDL_Window *prev;
45.     SDL_Window *next;
46. };

```

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]
1.  SDL_Window * SDLCALL SDL_CreateWindow(const char *title,
2.                                     int x, int y, int w,
3.                                     int h, Uint32 flags);
```

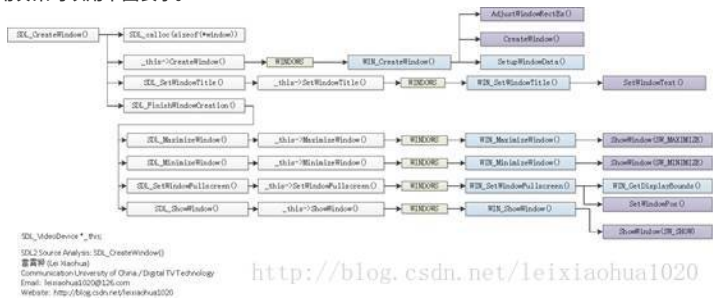
参数含义如下。

- 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]
1.  SDL_Window * SDL_CreateWindow(const char *title, int x, int y, int w, int h, Uint32 flags)
2.  {
3.      SDL_Window *window;
4.      const char *hint;
5.
6.
7.      if (!_this) {
8.          /* Initialize the video system if needed */
9.          if (SDL_VideoInit(NULL) < 0) {
10.              return NULL;
11.          }
12.      }
13.  }
```

```

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_malloc(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_malloc()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：

```

1. #define memset    SDL_memset
2. #define memcpy    SDL_memcpy
3. #define malloc    SDL_malloc
4. #define calloc    SDL_calloc
5. #define realloc   SDL_realloc
6. #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。

```

1. 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。

```

1. static SDL_VideoDevice *WIN_CreateDevice(int devindex)
2. {
3.     SDL_VideoDevice *device;
4.     SDL_VideoData *data;
5.
6.
7.     SDL_RegisterApp(NULL, 0, NULL);
8.
9.
10.    /* Initialize all variables that we clean on shutdown */
11.    device = (SDL_VideoDevice *) SDL_calloc(1, sizeof(SDL_VideoDevice));
12.    if (device) {
13.        data = (struct SDL_VideoData *) SDL_calloc(1, sizeof(SDL_VideoData));
14.    } else {
15.        data = NULL;
16.    }
17.    if (!data) {
18.        SDL_free(device);
19.        SDL_OutOfMemory();
20.        return NULL;
21.    }
22.    device->driverdata = data;
23.
24.
25.    data->userDLL = SDL_LoadObject("USER32.DLL");
26.    if (data->userDLL) {
27.        data->CloseTouchInputHandle = (BOOL (WINAPI *) ( HTOUCHINPUT )) SDL_LoadFunction(data->userDLL, "CloseTouchInputHandle");
28.        data->GetTouchInputInfo = (BOOL (WINAPI *) ( HTOUCHINPUT, UINT, PTOUCHINPUT, int )) SDL_LoadFunction(data->userDLL, "GetTouchInputInfo");
29.        data->RegisterTouchWindow = (BOOL (WINAPI *) ( HWND, ULONG )) SDL_LoadFunction(data->userDLL, "RegisterTouchWindow");
30.    }
31.
32.
33.    /* Set the function pointers */
34.    device->VideoInit = WIN_VideoInit;
35.    device->VideoQuit = WIN_VideoQuit;
36.    device->GetDisplayBounds = WIN_GetDisplayBounds;
37.    device->GetDisplayModes = WIN_GetDisplayModes;

```

```

38.     device->SetDisplayMode = WIN_SetDisplayMode;
39.     device->PumpEvents = WIN_PumpEvents;
40.
41.
42. #undef CreateWindow
43.     device->CreateWindow = WIN_CreateWindow;
44.     device->CreateWindowFrom = WIN_CreateWindowFrom;
45.     device->SetWindowTitle = WIN_SetWindowTitle;
46.     device->SetWindowIcon = WIN_SetWindowIcon;
47.     device->SetWindowPosition = WIN_SetWindowPosition;
48.     device->SetWindowSize = WIN_SetWindowSize;
49.     device->ShowWindow = WIN_ShowWindow;
50.     device->HideWindow = WIN_HideWindow;
51.     device->RaiseWindow = WIN_RaiseWindow;
52.     device->MaximizeWindow = WIN_MaximizeWindow;
53.     device->MinimizeWindow = WIN_MinimizeWindow;
54.     device->RestoreWindow = WIN_RestoreWindow;
55.     device->SetWindowBordered = WIN_SetWindowBordered;
56.     device->SetWindowFullscreen = WIN_SetWindowFullscreen;
57.     device->SetWindowGammaRamp = WIN_SetWindowGammaRamp;
58.     device->GetWindowGammaRamp = WIN_GetWindowGammaRamp;
59.     device->SetWindowGrab = WIN_SetWindowGrab;
60.     device->DestroyWindow = WIN_DestroyWindow;
61.     device->GetWindowWMInfo = WIN_GetWindowWMInfo;
62.     device->CreateWindowFramebuffer = WIN_CreateWindowFramebuffer;
63.     device->UpdateWindowFramebuffer = WIN_UpdateWindowFramebuffer;
64.     device->DestroyWindowFramebuffer = WIN_DestroyWindowFramebuffer;
65.     device->OnWindowEnter = WIN_OnWindowEnter;
66.
67.
68.     device->shape_driver.CreateShaper = Win32_CreateShaper;
69.     device->shape_driver.SetWindowShape = Win32_SetWindowShape;
70.     device->shape_driver.ResizeWindowShape = Win32_ResizeWindowShape;
71.
72.
73. #if SDL_VIDEO_OPENGL_WGL
74.     device->GL_LoadLibrary = WIN_GL_LoadLibrary;
75.     device->GL_GetProcAddress = WIN_GL_GetProcAddress;
76.     device->GL_UnloadLibrary = WIN_GL_UnloadLibrary;
77.     device->GL_CreateContext = WIN_GL_CreateContext;
78.     device->GL_MakeCurrent = WIN_GL_MakeCurrent;
79.     device->GL_SetSwapInterval = WIN_GL_SetSwapInterval;
80.     device->GL_GetSwapInterval = WIN_GL_GetSwapInterval;
81.     device->GL_SwapWindow = WIN_GL_SwapWindow;
82.     device->GL_DeleteContext = WIN_GL_DeleteContext;
83. #endif
84.     device->StartTextInput = WIN_StartTextInput;
85.     device->StopTextInput = WIN_StopTextInput;
86.     device->SetTextInputRect = WIN_SetTextInputRect;
87.
88.
89.     device->SetClipboardText = WIN_SetClipboardText;
90.     device->GetClipboardText = WIN_GetClipboardText;
91.     device->HasClipboardText = WIN_HasClipboardText;
92.
93.
94.     device->free = WIN_DeleteDevice;
95.
96.
97.     return device;
98. }

```

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）。

```

1. int WIN_CreateWindow(_THIS, SDL_Window * window)
2. {
3.     HWND hwnd;
4.     RECT rect;
5.     DWORD style = STYLE_BASIC;
6.     int x, y;
7.     int w, h;
8.
9.
10.    style |= GetWindowStyle(window);
11.
12.
13.    /* Figure out what the window area will be */
14.    rect.left = window->x;
15.    rect.top = window->y;
16.    rect.right = window->x + window->w;
17.    rect.bottom = window->y + window->h;
18.    AdjustWindowRectEx(&rect, style, FALSE, 0);
19.    x = rect.left;
20.    y = rect.top;
21.    w = (rect.right - rect.left);
22.    h = (rect.bottom - rect.top);
23.
24.
25.    hwnd =
26.        CreateWindow(SDL_Appname, TEXT(""), style, x, y, w, h, NULL, NULL,
27.                      SDL_Instance, NULL);
28.    if (!hwnd) {
29.        return WIN_SetError("Couldn't create window");
30.    }
31.
32.
33.    WIN_PumpEvents(_this);
34.
35.
36.    if (SetupWindowData(_this, window, hwnd, SDL_TRUE) < 0) {
37.        DestroyWindow(hwnd);
38.        return -1;
39.    }
40.
41.
42.    #if SDL_VIDEO_OPENGL_WGL
43.    /* We need to initialize the extensions before deciding how to create ES profiles */
44.    if (window->flags & SDL_WINDOW_OPENGL) {
45.        WIN_GL_InitExtensions(_this);
46.    }
47.    #endif
48.
49.
50.    #if SDL_VIDEO_OPENGL_ES2
51.    if ((window->flags & SDL_WINDOW_OPENGL) &&
52.        _this->gl_config.profile_mask == SDL_GL_CONTEXT_PROFILE_ES
53.    #if SDL_VIDEO_OPENGL_WGL
54.        && (!_this->gl_data || !_this->gl_data->HAS_WGL_EXT_create_context_es2_profile)
55.    #endif
56.    ) {
57.    #if SDL_VIDEO_OPENGL_EGL
58.        if (WIN_GLES_SetupWindow(_this, window) < 0) {
59.            WIN_DestroyWindow(_this, window);
60.            return -1;
61.        }
62.    #else
63.        return SDL_SetError("Could not create GLES window surface (no EGL support available)");
64.    #endif /* SDL_VIDEO_OPENGL_EGL */
65.    } else
66.    #endif /* SDL_VIDEO_OPENGL_ES2 */
67.
68.
69.    #if SDL_VIDEO_OPENGL_WGL
70.    if (window->flags & SDL_WINDOW_OPENGL) {
71.        if (WIN_GL_SetupWindow(_this, window) < 0) {
72.            WIN_DestroyWindow(_this, window);
73.            return -1;
74.        }
75.    }
76.    #endif
77.
78.
79.    return 0;
80. }

```

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]
1. void SDL_SetWindowTitle(SDL_Window * window, const char *title)
2. {
3.     CHECK_WINDOW_MAGIC(window, );
4.
5.
6.     if (title == window->title) {
7.         return;
8.     }
9.     SDL_free(window->title);
10.    if (title && *title) {
11.        window->title = SDL_strdup(title);
12.    } else {
13.        window->title = NULL;
14.    }
15.
16.
17.    if (_this->SetWindowTitle) {
18.        _this->SetWindowTitle(_this, window);
19.    }
20. }
```

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]
1. void WIN_SetWindowTitle(_THIS, SDL_Window * window)
2. {
3.     HWND hwnd = ((SDL_WindowData *) window->driverdata)->hwnd;
4.     LPCTSTR title;
5.
6.
7.     if (window->title) {
8.         title = WIN_UTF8ToString(window->title);
9.     } else {
10.        title = NULL;
11.    }
12.    SetWindowText(hwnd, title ? title : TEXT(""));
13.    SDL_free(title);
14. }
```

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]
1. static void SDL_FinishWindowCreation(SDL_Window *window, Uint32 flags)
2. {
3.     window->windowed.x = window->x;
4.     window->windowed.y = window->y;
5.     window->windowed.w = window->w;
6.     window->windowed.h = window->h;
7.
8.
9.     if (flags & SDL_WINDOW_MAXIMIZED) {
10.        SDL_MaximizeWindow(window);
11.    }
12.     if (flags & SDL_WINDOW_MINIMIZED) {
13.        SDL_MinimizeWindow(window);
14.    }
15.     if (flags & SDL_WINDOW_FULLSCREEN) {
16.        SDL_SetWindowFullscreen(window, flags);
17.    }
18.     if (flags & SDL_WINDOW_INPUT_GRABBED) {
19.        SDL_SetWindowGrab(window, SDL_TRUE);
20.    }
21.     if (!(flags & SDL_WINDOW_HIDDEN)) {
22.        SDL_ShowWindow(window);
23.    }
24. }
```

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]
1. void SDL_MaximizeWindow(SDL_Window * window)
2. {
3.     CHECK_WINDOW_MAGIC(window, );
4.
5.
6.     if (window->flags & SDL_WINDOW_MAXIMIZED) {
7.         return;
8.     }
9.
10.
11.     /* !!! FIXME: should this check if the window is resizable? */
12.
13.
14.     if (_this->MaximizeWindow) {
15.         _this->MaximizeWindow(_this, window);
16.     }
17. }
```

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]
1. void WIN_MaximizeWindow(_THIS, SDL_Window * window)
2. {
3.     SDL_WindowData *data = (SDL_WindowData *)window->driverdata;
4.     HWND hwnd = data->hwnd;
5.     data->expected_resize = TRUE;
6.     ShowWindow(hwnd, SW_MAXIMIZE);
7.     data->expected_resize = FALSE;
8. }
```

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]
1. void SDL_MinimizeWindow(SDL_Window * window)
2. {
3.     CHECK_WINDOW_MAGIC(window, );
4.
5.
6.     if (window->flags & SDL_WINDOW_MINIMIZED) {
7.         return;
8.     }
9.
10.
11.     SDL_UpdateFullscreenMode(window, SDL_FALSE);
12.
13.
14.     if (_this->MinimizeWindow) {
15.         _this->MinimizeWindow(_this, window);
16.     }
17. }
```

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]
1. void WIN_MinimizeWindow(_THIS, SDL_Window * window)
2. {
3.     HWND hwnd = ((SDL_WindowData *) window->driverdata)->hwnd;
4.     ShowWindow(hwnd, SW_MINIMIZE);
5. }
```

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]
1. int SDL_SetWindowFullscreen(SDL_Window * window, Uint32 flags)
2. {
3.     CHECK_WINDOW_MAGIC(window, -1);
4.
5.
6.     flags &= FULLSCREEN_MASK;
7.
8.
9.     if ( flags == (window->flags & FULLSCREEN_MASK) ) {
10.        return 0;
11.    }
12.
13.
14.    /* clear the previous flags and OR in the new ones */
15.    window->flags &= ~FULLSCREEN_MASK;
16.    window->flags |= flags;
17.
18.
19.    SDL_UpdateFullscreenMode(window, FULLSCREEN_VISIBLE(window));
20.
21.
22.    return 0;
23. }
```

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]
1. static void SDL_UpdateFullscreenMode(SDL_Window * window, SDL_bool fullscreen)
2. {
3.     SDL_VideoDisplay *display;
4.     SDL_Window *other;
5.
6.
7.     #ifdef __MACOSX__
8.         if (Cocoa_SetWindowFullscreenSpace(window, fullscreen)) {
9.             window->last_fullscreen_flags = window->flags;
10.            return;
11.        }
12.    #endif
13.
14.
15.    display = SDL_GetDisplayForWindow(window);
16.
17.
18.    if (fullscreen) {
19.        /* Hide any other fullscreen windows */
20.        if (display->fullscreen_window &&
21.            display->fullscreen_window != window) {
22.            SDL_MinimizeWindow(display->fullscreen_window);
23.        }
24.    }
25.
26.
27.    /* See if anything needs to be done now */
28.    if ((display->fullscreen_window == window) == fullscreen) {
29.        if ((window->last_fullscreen_flags & FULLSCREEN_MASK) == (window->flags & FULLSCREEN_MASK)) {
30.            return;
31.        }
32.    }
33.
34.
35.    /* See if there are any fullscreen windows */
36.    for (other = _this->windows; other; other = other->next) {
37.        SDL_bool setDisplayMode = SDL_FALSE;
38.
39.
40.        if (other == window) {
41.            setDisplayMode = fullscreen;
42.        } else if (FULLSCREEN_VISIBLE(other) &&
43.            SDL_GetDisplayForWindow(other) == display) {
44.            setDisplayMode = SDL_TRUE;
45.        }
46.    }
```

```

47.
48.     if (setDisplayMode) {
49.         SDL_DisplayMode fullscreen_mode;
50.
51.
52.         if (SDL_GetWindowDisplayMode(other, &fullscreen_mode) == 0) {
53.             SDL_bool resized = SDL_TRUE;
54.
55.
56.             if (other->w == fullscreen_mode.w && other->h == fullscreen_mode.h) {
57.                 resized = SDL_FALSE;
58.             }
59.
60.
61.             /* only do the mode change if we want exclusive fullscreen */
62.             if ((window->flags & SDL_WINDOW_FULLSCREEN_DESKTOP) != SDL_WINDOW_FULLSCREEN_DESKTOP) {
63.                 SDL_SetDisplayModeForDisplay(display, &fullscreen_mode);
64.             } else {
65.                 SDL_SetDisplayModeForDisplay(display, NULL);
66.             }
67.
68.
69.             if (_this->SetWindowFullscreen) {
70.                 _this->SetWindowFullscreen(_this, other, display, SDL_TRUE);
71.             }
72.             display->fullscreen_window = other;
73.
74.
75.             /* Generate a mode change event here */
76.             if (resized) {
77.                 SDL_SendWindowEvent(other, SDL_WINEVENT_RESIZED,
78.                                     fullscreen_mode.w, fullscreen_mode.h);
79.             } else {
80.                 SDL_OnWindowResized(other);
81.             }
82.
83.
84.             SDL_RestoreMousePosition(other);
85.
86.
87.             window->last_fullscreen_flags = window->flags;
88.             return;
89.         }
90.     }
91. }
92.
93.
94. /* Nope, restore the desktop mode */
95. SDL_SetDisplayModeForDisplay(display, NULL);
96.
97.
98. if (_this->SetWindowFullscreen) {
99.     _this->SetWindowFullscreen(_this, window, display, SDL_FALSE);
100. }
101. display->fullscreen_window = NULL;
102.
103.
104. /* Generate a mode change event here */
105. SDL_OnWindowResized(window);
106.
107.
108. /* Restore the cursor position */
109. SDL_RestoreMousePosition(window);
110.
111.
112. window->last_fullscreen_flags = window->flags;
113. }

```

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。

```

1. void WIN_SetWindowFullscreen(_THIS, SDL_Window * window, SDL_VideoDisplay * display, SDL_bool fullscreen)
2. {
3.     SDL_WindowData *data = (SDL_WindowData *) window->driverdata;
4.     HWND hwnd = data->hwnd;
5.     RECT rect;
6.     SDL_Rect bounds;
7.     DWORD style;
8.     HWND top;
9.     BOOL menu;
10.    int x, y;
11.    int w, h;
12.
13.
14.    if (SDL_ShouldAllowTopmost() && (window->flags & (SDL_WINDOW_FULLSCREEN|SDL_WINDOW_INPUT_FOCUS)) == (SDL_WINDOW_FULLSCREEN|SDL_WINDOW_INPUT_FOCUS)) {
15.        top = HWND_TOPMOST;
16.    } else {
17.        top = HWND_NOTOPMOST;
18.    }
19.
20.
21.    style = GetWindowLong(hwnd, GWL_STYLE);
22.    style &= ~STYLE_MASK;
23.    style |= GetWindowStyle(window);
24.
25.
26.    WIN_GetDisplayBounds(_this, display, &bounds);
27.
28.
29.    if (fullscreen) {
30.        x = bounds.x;
31.        y = bounds.y;
32.        w = bounds.w;
33.        h = bounds.h;
34.    } else {
35.        rect.left = 0;
36.        rect.top = 0;
37.        rect.right = window->windowed.w;
38.        rect.bottom = window->windowed.h;
39.        menu = (style & WS_CHILDWINDOW) ? FALSE : (GetMenu(hwnd) != NULL);
40.        AdjustWindowRectEx(&rect, style, menu, 0);
41.        w = (rect.right - rect.left);
42.        h = (rect.bottom - rect.top);
43.        x = window->windowed.x + rect.left;
44.        y = window->windowed.y + rect.top;
45.    }
46.    SetWindowLong(hwnd, GWL_STYLE, style);
47.    data->expected_resize = TRUE;
48.    SetWindowPos(hwnd, top, x, y, w, h, SWP_NOCOPYBITS | SWP_NOACTIVATE);
49.    data->expected_resize = FALSE;
50. }

```

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。

```

1. void SDL_ShowWindow(SDL_Window * window)
2. {
3.     CHECK_WINDOW_MAGIC(window, );
4.
5.
6.     if (window->flags & SDL_WINDOW_SHOWN) {
7.         return;
8.     }
9.
10.
11.     if (_this->ShowWindow) {
12.         _this->ShowWindow(_this, window);
13.     }
14.     SDL_SendWindowEvent(window, SDL_WINEVENT_SHOWN, 0, 0);
15. }

```

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。

```

1. void WIN_ShowWindow(_THIS, SDL_Window * window)
2. {
3.     HWND hwnd = ((SDL_WindowData *) window->driverdata)->hwnd;
4.     ShowWindow(hwnd, SW_SHOW);
5. }

```

该函数比较简单，直接调用了Win32中的ShowWindow()方法。

版权声明：本文为博主原创文章，未经博主允许不得转载。 <https://blog.csdn.net/leixiaohua1020/article/details/40701203>

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](#)

此PDF由[spygg](#)生成,请尊重原作者版权!!!

我的邮箱:liushidc@163.com