

## 原 SDL2源代码分析6：复制到渲染器（SDL\_RenderCopy()）

2014年11月08日 00:54:00 阅读数：8449

=====

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更新纹理像素数据的函数SDL\_UpdateTexture()。这篇文章继续分析SDL的源代码。本文分析SDL纹理复制到渲染目标的函数SDL\_RenderCopy()。

□

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5个函数SDL\_UpdateTexture()。本文继续分析该流程中的第6个函数SDL\_RenderCopy()。

## SDL\_RenderCopy()

### 函数简介

SDL使用SDL\_RenderCopy()将纹理数据复制给渲染目标。SDL\_RenderCopy()的原型如下。

```
[cpp]    
1. int SDLCALL SDL_RenderCopy(SDL_Renderer * renderer,  
2.                                SDL_Texture * texture,  
3.                                const SDL_Rect * srcrect,  
4.                                const SDL_Rect * dstrect);
```

参数的含义如下。

renderer：渲染目标。

texture：输入纹理。

srcrect：选择输入纹理的一块矩形区域作为输入。设置为NULL的时候整个纹理作为输入。

dstrect：选择渲染目标的一块矩形区域作为输出。设置为NULL的时候整个渲染目标作为输出。

成功的话返回0，失败的话返回-1。

## 函数调用关系图

SDL\_RenderCopy() 关键函数的调用关系可以用下图表示。

□

上面的图片不太清晰，更清晰的图片上传到了相册里面：

<http://my.csdn.net/leixiaohua1020/album/detail/1793911>

把相册里面的图片保存下来就可以得到清晰的图片了。

## 源代码分析

SDL\_RenderCopy() 的源代码位于 render\SDL\_render.c 中，如下所示。

```
[cpp]    
1.  int SDL_RenderCopy(SDL_Renderer * renderer, SDL_Texture * texture,  
2.      const SDL_Rect * srcrect, const SDL_Rect * dstrect)  
3.  {  
4.      SDL_Rect real_srcrect = { 0, 0, 0, 0 };  
5.      SDL_Rect real_dstrect = { 0, 0, 0, 0 };  
6.      SDL_FRect frect;  
7.  
8.  
9.      CHECK_RENDERER_MAGIC(renderer, -1);  
10.     CHECK_TEXTURE_MAGIC(texture, -1);  
11.  
12.  
13.     if (renderer != texture->renderer) {  
14.         return SDL_SetError("Texture was not created with this renderer");  
15.     }  
16.  
17.  
18.     real_srcrect.x = 0;  
19.     real_srcrect.y = 0;  
20.     real_srcrect.w = texture->w;  
21.     real_srcrect.h = texture->h;  
22.     if (srcrect) {  
23.         if (!SDL_IntersectRect(srcrect, &real_srcrect, &real_srcrect)) {  
24.             return 0;  
25.         }  
26.     }  
27.  
28.  
29.     SDL_RenderGetViewport(renderer, &real_dstrect);  
30.     real_dstrect.x = 0;  
31.     real_dstrect.y = 0;  
32.     if (dstrect) {  
33.         if (!SDL_HasIntersection(dstrect, &real_dstrect)) {  
34.             return 0;  
35.         }  
36.         real_dstrect = *dstrect;  
37.     }  
38.  
39.  
40.     if (texture->native) {  
41.         texture = texture->native;  
42.     }  
43.  
44.  
45.     /* Don't draw while we're hidden */  
46.     if (renderer->hidden) {  
47.         return 0;  
48.     }  
49.  
50.  
51.     frect.x = real_dstrect.x * renderer->scale.x;  
52.     frect.y = real_dstrect.y * renderer->scale.y;  
53.     frect.w = real_dstrect.w * renderer->scale.x;  
54.     frect.h = real_dstrect.h * renderer->scale.y;  
55.  
56.  
57.     return renderer->RenderCopy(renderer, texture, &real_srcrect, &frect);  
58. }
```

从源代码中可以看出，SDL\_RenderCopy() 的大致流程如下。

1. 检查输入参数的合理性。
2. 调用 SDL\_Renderer 的 RenderCopy () 方法复制纹理到渲染目标。 这一步是整个函数的核心。

下面我们详细看一下几种不同的渲染器的RenderCopy()的方法。

1.

## Direct3D

Direct3D 渲染器中对应RenderCopy()的函数是D3D\_RenderCopy(), 它的源代码如下所示 (位于render\direct3d\SDL\_render\_d3d.c)。

```
[cpp] [🔍] [📄]  
1. static int D3D_RenderCopy(SDL_Renderer * renderer, SDL_Texture * texture,  
2. const SDL_Rect * srcrect, const SDL_FRect * dstrect)  
3. {  
4.     D3D_RenderData *data = (D3D_RenderData *) renderer->driverdata;  
5.     D3D_TextureData *texturedata;  
6.     LPDIRECT3DPixelShader9 shader = NULL;  
7.     float minx, miny, maxx, maxy;  
8.     float minu, maxu, minv, maxv;  
9.     DWORD color;  
10.    Vertex vertices[4];  
11.    HRESULT result;  
12.  
13.  
14.    if (D3D_ActivateRenderer(renderer) < 0) {  
15.        return -1;  
16.    }  
17.  
18.  
19.    texturedata = (D3D_TextureData *)texture->driverdata;  
20.    if (!texturedata) {  
21.        SDL_SetError("Texture is not currently available");  
22.        return -1;  
23.    }  
24.  
25.  
26.    minx = dstrect->x - 0.5f;  
27.    miny = dstrect->y - 0.5f;  
28.    maxx = dstrect->x + dstrect->w - 0.5f;  
29.    maxy = dstrect->y + dstrect->h - 0.5f;  
30.  
31.  
32.    minu = (float) srcrect->x / texture->w;  
33.    maxu = (float) (srcrect->x + srcrect->w) / texture->w;  
34.    minv = (float) srcrect->y / texture->h;  
35.    maxv = (float) (srcrect->y + srcrect->h) / texture->h;  
36.  
37.  
38.    color = D3DCOLOR_ARGB(texture->a, texture->r, texture->g, texture->b);  
39.  
40.  
41.    vertices[0].x = minx;  
42.    vertices[0].y = miny;  
43.    vertices[0].z = 0.0f;  
44.    vertices[0].color = color;  
45.    vertices[0].u = minu;  
46.    vertices[0].v = minv;  
47.  
48.  
49.    vertices[1].x = maxx;  
50.    vertices[1].y = miny;  
51.    vertices[1].z = 0.0f;  
52.    vertices[1].color = color;  
53.    vertices[1].u = maxu;  
54.    vertices[1].v = minv;  
55.  
56.  
57.    vertices[2].x = maxx;  
58.    vertices[2].y = maxy;  
59.    vertices[2].z = 0.0f;  
60.    vertices[2].color = color;  
61.    vertices[2].u = maxu;  
62.    vertices[2].v = maxv;  
63.  
64.  
65.    vertices[3].x = minx;  
66.    vertices[3].y = maxy;  
67.    vertices[3].z = 0.0f;  
68.    vertices[3].color = color;  
69.    vertices[3].u = minu;  
70.    vertices[3].v = maxv;  
71.  
72.  
73.    D3D_SetBlendMode(data, texture->blendMode);  
74.  
75.  
76.    D3D_UpdateTextureScaleMode(data, texturedata, 0);  
77.  
78.  
79.    result =  
80.        IDirect3DDevice9_SetTexture(data->device, 0, (IDirect3DBaseTexture9 *)  
81.            texturedata->texture);
```

```

81.         texturedata->texture;
82.     if (FAILED(result)) {
83.         return D3D_SetError("SetTexture()", result);
84.     }
85.
86.
87.     if (texturedata->yuv) {
88.         shader = data->ps_yuv;
89.
90.
91.         D3D_UpdateTextureScaleMode(data, texturedata, 1);
92.         D3D_UpdateTextureScaleMode(data, texturedata, 2);
93.
94.
95.         result =
96.             IDirect3DDevice9_SetTexture(data->device, 1, (IDirect3DBaseTexture9 *)
97.                 texturedata->utexture);
98.         if (FAILED(result)) {
99.             return D3D_SetError("SetTexture()", result);
100.        }
101.
102.
103.        result =
104.            IDirect3DDevice9_SetTexture(data->device, 2, (IDirect3DBaseTexture9 *)
105.                texturedata->vtexture);
106.        if (FAILED(result)) {
107.            return D3D_SetError("SetTexture()", result);
108.        }
109.    }
110.
111.
112.    if (shader) {
113.        result = IDirect3DDevice9_SetPixelShader(data->device, shader);
114.        if (FAILED(result)) {
115.            return D3D_SetError("SetShader()", result);
116.        }
117.    }
118.    result =
119.        IDirect3DDevice9_DrawPrimitiveUP(data->device, D3DPT_TRIANGLEFAN, 2,
120.            vertices, sizeof(*vertices));
121.    if (FAILED(result)) {
122.        return D3D_SetError("DrawPrimitiveUP()", result);
123.    }
124.    if (shader) {
125.        result = IDirect3DDevice9_SetPixelShader(data->device, NULL);
126.        if (FAILED(result)) {
127.            return D3D_SetError("SetShader()", result);
128.        }
129.    }
130.    return 0;
131. }

```

从代码中可以看出，D3D\_RenderCopy()函数按照执行的顺序调用了如下函数：

D3D\_ActivateRenderer()：激活渲染器。其内部使用Direct3D的API函数IDirect3DDevice9\_BeginScene()开始一个D3D的场景。

D3D\_SetBlendMode()：设置渲染器状态。其内部使用Direct3D的API函数IDirect3DDevice9\_SetRenderState()设置渲染器的状态。

D3D\_UpdateTextureScaleMode()：设置纹理采样方式。其内部调用使用Direct3D的API函数IDirect3DDevice9\_SetSamplerState()设置D3D的纹理采样方式。

IDirect3DDevice9\_SetTexture()：Direct3D的API，用于设置当前启用的纹理。

IDirect3DDevice9\_SetPixelShader()：Direct3D的API，用于设置使用的像素着色器。

IDirect3DDevice9\_DrawPrimitiveUP()：Direct3D的API，用于渲染。

上述几个函数中，前3个函数是SDL中的函数，后3个函数是Direct3D的API。在此附上前三个函数的代码。

**D3D\_ActivateRenderer()：激活渲染器。**

```

1. static int D3D_ActivateRenderer(SDL_Renderer * renderer)
2. {
3.     D3D_RenderData *data = (D3D_RenderData *) renderer->driverdata;
4.     HRESULT result;
5.
6.
7.     if (data->updateSize) {
8.         SDL_Window *window = renderer->window;
9.         int w, h;
10.
11.
12.         SDL_GetWindowSize(window, &w, &h);
13.         data->pparams.BackBufferWidth = w;
14.         data->pparams.BackBufferHeight = h;
15.         if (SDL_GetWindowFlags(window) & SDL_WINDOW_FULLSCREEN) {
16.             data->pparams.BackBufferFormat =
17.                 PixelFormatToD3DFMT(SDL_GetWindowPixelFormat(window));
18.         } else {
19.             data->pparams.BackBufferFormat = D3DFMT_UNKNOWN;
20.         }
21.         if (D3D_Reset(renderer) < 0) {
22.             return -1;
23.         }
24.
25.
26.         data->updateSize = SDL_FALSE;
27.     }
28.     if (data->beginScene) {
29.         result = IDirect3DDevice9_BeginScene(data->device);
30.         if (result == D3DERR_DEVICELOST) {
31.             if (D3D_Reset(renderer) < 0) {
32.                 return -1;
33.             }
34.             result = IDirect3DDevice9_BeginScene(data->device);
35.         }
36.         if (FAILED(result)) {
37.             return D3D_SetError("BeginScene()", result);
38.         }
39.         data->beginScene = SDL_FALSE;
40.     }
41.     return 0;
42. }

```

D3D\_SetBlendMode()：设置渲染器状态。

```

1. static void D3D_SetBlendMode(D3D_RenderData * data, int blendMode)
2. {
3.     switch (blendMode) {
4.     case SDL_BLENDMODE_NONE:
5.         IDirect3DDevice9_SetRenderState(data->device, D3DRS_ALPHABLENDENABLE,
6.                                         FALSE);
7.         break;
8.     case SDL_BLENDMODE_BLEND:
9.         IDirect3DDevice9_SetRenderState(data->device, D3DRS_ALPHABLENDENABLE,
10.                                         TRUE);
11.         IDirect3DDevice9_SetRenderState(data->device, D3DRS_SRCBLEND,
12.                                         D3DBLEND_SRCALPHA);
13.         IDirect3DDevice9_SetRenderState(data->device, D3DRS_DESTBLEND,
14.                                         D3DBLEND_INVSRCALPHA);
15.         if (data->enableSeparateAlphaBlend) {
16.             IDirect3DDevice9_SetRenderState(data->device, D3DRS_SRCBLENDALPHA,
17.                                             D3DBLEND_ONE);
18.             IDirect3DDevice9_SetRenderState(data->device, D3DRS_DESTBLENDALPHA,
19.                                             D3DBLEND_INVSRCALPHA);
20.         }
21.         break;
22.     case SDL_BLENDMODE_ADD:
23.         IDirect3DDevice9_SetRenderState(data->device, D3DRS_ALPHABLENDENABLE,
24.                                         TRUE);
25.         IDirect3DDevice9_SetRenderState(data->device, D3DRS_SRCBLEND,
26.                                         D3DBLEND_SRCALPHA);
27.         IDirect3DDevice9_SetRenderState(data->device, D3DRS_DESTBLEND,
28.                                         D3DBLEND_ONE);
29.         if (data->enableSeparateAlphaBlend) {
30.             IDirect3DDevice9_SetRenderState(data->device, D3DRS_SRCBLENDALPHA,
31.                                             D3DBLEND_ZERO);
32.             IDirect3DDevice9_SetRenderState(data->device, D3DRS_DESTBLENDALPHA,
33.                                             D3DBLEND_ONE);
34.         }
35.         break;
36.     case SDL_BLENDMODE_MOD:
37.         IDirect3DDevice9_SetRenderState(data->device, D3DRS_ALPHABLENDENABLE,
38.                                         TRUE);
39.         IDirect3DDevice9_SetRenderState(data->device, D3DRS_SRCBLEND,
40.                                         D3DBLEND_ZERO);
41.         IDirect3DDevice9_SetRenderState(data->device, D3DRS_DESTBLEND,
42.                                         D3DBLEND_SRCCOLOR);
43.         if (data->enableSeparateAlphaBlend) {
44.             IDirect3DDevice9_SetRenderState(data->device, D3DRS_SRCBLENDALPHA,
45.                                             D3DBLEND_ZERO);
46.             IDirect3DDevice9_SetRenderState(data->device, D3DRS_DESTBLENDALPHA,
47.                                             D3DBLEND_ONE);
48.         }
49.         break;
50.     }
51. }

```

**D3D\_UpdateTextureScaleMode()：设置纹理采样方式。**

```

1. static void D3D_UpdateTextureScaleMode(D3D_RenderData *data, D3D_TextureData *texturedata, unsigned index)
2. {
3.     if (texturedata->scaleMode != data->scaleMode[index]) {
4.         IDirect3DDevice9_SetSamplerState(data->device, index, D3DSAMP_MINFILTER,
5.                                         texturedata->scaleMode);
6.         IDirect3DDevice9_SetSamplerState(data->device, index, D3DSAMP_MAGFILTER,
7.                                         texturedata->scaleMode);
8.         data->scaleMode[index] = texturedata->scaleMode;
9.     }
10. }

```

## 2.

### OpenGL

OpenGL渲染器中对应RenderCopy()的函数是GL\_RenderCopy(), 它的源代码如下所示（位于render\opengl\SDL\_render\_gl.c）。

```

1. static int GL_RenderCopy(SDL_Renderer * renderer, SDL_Texture * texture,
2.     const SDL_Rect * srcrect, const SDL_FRect * dstrect)
3. {
4.     GL_RenderData *data = (GL_RenderData *) renderer->driverdata;
5.     GL_TextureData *texturedata = (GL_TextureData *) texture->driverdata;
6.     GLfloat minx, miny, maxx, maxy;
7.     GLfloat minu, maxu, minv, maxv;
8.
9.
10.    GL_ActivateRenderer(renderer);
11.
12.
13.    data->glEnable(texturedata->type);
14.    if (texturedata->yuv) {
15.        data->glActiveTextureARB(GL_TEXTURE2_ARB);
16.        data->glBindTexture(texturedata->type, texturedata->vtexture);
17.
18.
19.        data->glActiveTextureARB(GL_TEXTURE1_ARB);
20.        data->glBindTexture(texturedata->type, texturedata->utexture);
21.
22.
23.        data->glActiveTextureARB(GL_TEXTURE0_ARB);
24.    }
25.    data->glBindTexture(texturedata->type, texturedata->texture);
26.
27.
28.    if (texture->modMode) {
29.        GL_SetColor(data, texture->r, texture->g, texture->b, texture->a);
30.    } else {
31.        GL_SetColor(data, 255, 255, 255, 255);
32.    }
33.
34.
35.    GL_SetBlendMode(data, texture->blendMode);
36.
37.
38.    if (texturedata->yuv) {
39.        GL_SetShader(data, SHADER_YV12);
40.    } else {
41.        GL_SetShader(data, SHADER_RGB);
42.    }
43.
44.
45.    minx = dstrect->x;
46.    miny = dstrect->y;
47.    maxx = dstrect->x + dstrect->w;
48.    maxy = dstrect->y + dstrect->h;
49.
50.
51.    minu = (GLfloat) srcrect->x / texture->w;
52.    minu *= texturedata->texw;
53.    maxu = (GLfloat) (srcrect->x + srcrect->w) / texture->w;
54.    maxu *= texturedata->texw;
55.    minv = (GLfloat) srcrect->y / texture->h;
56.    minv *= texturedata->texh;
57.    maxv = (GLfloat) (srcrect->y + srcrect->h) / texture->h;
58.    maxv *= texturedata->texh;
59.
60.
61.    data->glBegin(GL_TRIANGLE_STRIP);
62.    data->glTexCoord2f(minu, minv);
63.    data->glVertex2f(minx, miny);
64.    data->glTexCoord2f(maxu, minv);
65.    data->glVertex2f(maxx, miny);
66.    data->glTexCoord2f(minu, maxv);
67.    data->glVertex2f(minx, maxy);
68.    data->glTexCoord2f(maxu, maxv);
69.    data->glVertex2f(maxx, maxy);
70.    data->glEnd();
71.
72.
73.    data->glDisable(texturedata->type);
74.
75.
76.    return GL_CheckError("", renderer);
77. }

```

从代码中可以看出，GL\_RenderCopy()函数调用了OpenGL的API函数glActiveTexture(), glBindTexture()创建了一个纹理。并且使用GL\_SetBlendMode(), GL\_SetShader()设置了有关的一些参数。

有一点需要注意，在OpenGL渲染器中，如果输入像素格式是YUV，就会使用3个纹理。

### 3.

#### Software

Software渲染器中对应RenderCopy()的函数是SW\_RenderCopy(), 它的源代码如下所示 (位于render\software\SDL\_render\_sw.c)。

```
[cpp]
1. static int SW_RenderCopy(SDL_Renderer * renderer, SDL_Texture * texture,
2.     const SDL_Rect * srcrect, const SDL_FRect * dstrect)
3. {
4.     SDL_Surface *surface = SW_ActivateRenderer(renderer);
5.     SDL_Surface *src = (SDL_Surface *) texture->driverdata;
6.     SDL_Rect final_rect;
7.
8.
9.     if (!surface) {
10.         return -1;
11.     }
12.
13.
14.     if (renderer->viewport.x || renderer->viewport.y) {
15.         final_rect.x = (int)(renderer->viewport.x + dstrect->x);
16.         final_rect.y = (int)(renderer->viewport.y + dstrect->y);
17.     } else {
18.         final_rect.x = (int)dstrect->x;
19.         final_rect.y = (int)dstrect->y;
20.     }
21.     final_rect.w = (int)dstrect->w;
22.     final_rect.h = (int)dstrect->h;
23.
24.
25.     if ( srcrect->w == final_rect.w && srcrect->h == final_rect.h ) {
26.         return SDL_BlitSurface(src, srcrect, surface, &final_rect);
27.     } else {
28.         return SDL_BlitScaled(src, srcrect, surface, &final_rect);
29.     }
30. }
```

该函数的源代码还没有详细分析。

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/40895593>

文章标签： [SDL](#) [Direct3D](#) [OpenGL](#) [GDI](#) [渲染](#)

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