User's Manual

RENESAS

CONFIDENTIAL

RH850/D1x Device Family

Renesas Graphics Library
2D Graphics (DRW2D) Driver

User's Manual: Software

Renesas Electronics
www.renesas.com

Rev. 2.0 Dec 2021

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

How to Use This Manual

1. Purpose and Target Readers

This manual is designed to provide the user with an understanding the functions of DRW2D. This manual is written for engineers who use DRW2D.

> Particular attention should be paid to the precautionary notes when using the manual. These notes occur within the body of the text, at the end of each section, and in the Usage Notes section.

> The revision history summarizes the locations of revisions and additions. It does not list all revisions. Refer to the text of the manual for details.

Please refer to documents of drivers and hardware for a target system implementing DRW2D as necessary.

The following documents are related documents. Make sure to refer to the latest versions of these documents.

Document Type	Description	Document Title	Document No.
User's manual for Hardware	Hardware specifications (pin assignments, memory maps, peripheral function specifications, electrical characteristics, timing charts) and operation description	RH850/D1L/D1M Group User's Manual: Hardware	R01UH0451EJ0220
User's manual for Software	Description of RGL overview	Renesas Graphics Library User's Manual: Software	R01US0181ED0400
	Description of WM	Renesas Graphics Library Window Manager (WM) Driver User's Manual: Software	LLWEB-10035990
	Description of SPEA	Renesas Graphics Library Sprite Engine A (SPEA) Driver User's Manual: Software	LLWEB-10035991
	Description of VDCE	Renesas Graphics Library Video Data Controller E (VDCE) Driver User's Manual: Software	LLWEB-10035992
	Description of VOWE	Renesas Graphics Library Video Output Warping Engine (VOWE) Driver User's Manual: Software	LLWEB-10035993
	Description of JCUA	Renesas Graphics Library JPEG Codec Unit A (JCUA) Driver User's Manual: Software	LLWEB-10035994
	Description of SFMA	Renesas Graphics Library Serial Flash Memory Interface A (SFMA) Driver User's Manual: Software	LLWEB-10064753
	Description of HYPB	Renesas Graphics Library HyperBus Controller (HYPB) Driver User's Manual: Software	LLWEB-10064754
	Description of OCTA	Renesas Graphics Library OctaBus Controller (OCTA) Driver User's Manual: Software	LLWEB-10064755
	Description of VOCA	Renesas Graphics Library Video Output Checker (VOCA) Driver User's Manual: Software	LLWEB-10063801

	Description of DISCOM	Renesas Graphics Library Display Output Comparator (DISCOM) Driver User's Manual: Software	LLWEB-10063802
	Description of DRW2D	Renesas Graphics Library 2D Graphics (DRW2D) Driver User's Manual: Software	LLWEB-10059472 (This manual)
Porting Layer Guide	Description of porting layer of RGL	Renesas Graphics Library Porting Layer Guide	LLWEB-10035995

2. Notation of Numbers and Symbols

This manual uses the following notation.

Binary	0bXXXXXXXX	(X=0 or 1)
Decimal	XXX	(X=0-9)
Hex	0xXXXXXXXX	(X=0-9,A-F)

3. List of Abbreviations and Acronyms

Abbreviation	Full Form
API	Application Programming Interface
Context	An internal state machine of the single framework
CLUT	Color look up table.
CPU	Central Processing Unit. The microprocessor core of the LSI.
D/AVE HD	(a.k.a DHD) Display controller / Accelerated Vector Engine, High Definition. SW and HW name for 2D graphics.
Device	A SW abstraction of the HW or SW macro
Drw2D	Draw Two Dimensions. SW abstraction layer for 2D graphics.
Framebuffer	A region in the memory attached to a window that can be shown on the screen; a region in the memory holding the bitmap as the result of GPU rendering activities
GPU	Graphics Processing Unit.The graphical processing unit HW macro of the LSI
GPU2D	HW macro name of the LSI.
HW	Hardware
Layer	A HW concept of the stackable visual area on the display
Pitch	(a.k.a. stride) Distance in pixels between two adjacent pixel rows of the framebuffer in the memory
RLE	Run Length Encoding. TARGA run-length encoded image standard, for easy image compression, supported by the DaveHD
Screen	A physical display surface; a SW abstraction of the attached physical display
Surface	A concrete (i.e. physical) implementation of the window's area
SW	Software
Target	A platform (HW or SW) where the framework and application are intended to run
Texture	A binary image registered with the GPU driver that can be transformed and drawn to the framebuffer in a HW accelerated way
Window	A SW abstraction of the rectangular visual area that can be shown on the display
WM	Window manager.

Table of Contents

1. Overview

1.1 Feature and Scope

Drw2D is a simple and slim abstraction layer to render primitives and textures.
Its underlying graphics stack is the hardware-accelerated D/AVE HD GPU or CPU drawing. Which one is provided depends on the RH850 device.

- Drw2D with D/AVE HD
- Drw2D with CPU drawing

1.2 Component Structure

The component structure of Drw2D with D/AVE HD is shown in ***Figure 1.1***.

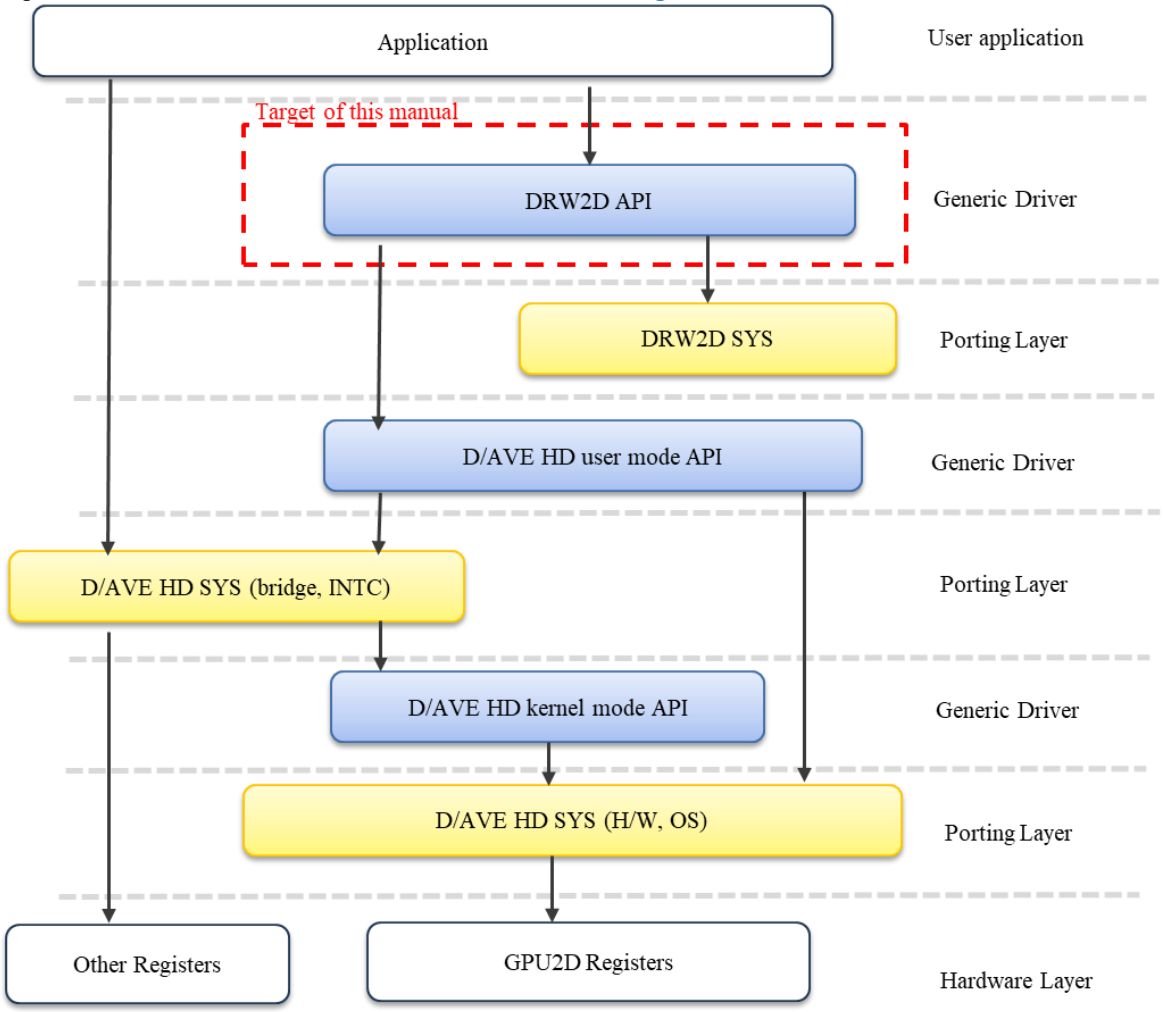

Figure 1.1 Component Structure of Drw2D with D/AVE HD

The component structure of Drw2D with CPU drawing is shown in *Figure 1.2.*

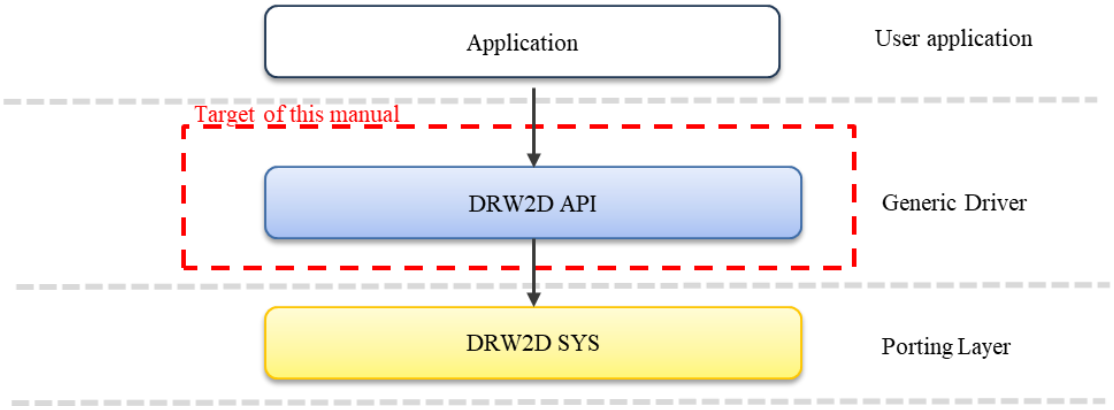

Figure 1.2 Component Structure of Drw2D with CPU drawing

For the details of the API, please refer to *Chapter.4*.

2. Basic Specification

2.1 Summary Specification

The summary of Drw2D with D/AVE HD specification is described in *Table 2-1.*

Table 2-1 Summary Specification of Drw2D with D/AVE HD

Items	Description
Target LSI	RH850/D1M1(H), RH850/D1M1-V2, RH850/D1M1A, RH850/D1M2(H)
Main Feature	• 2D, 2.5D Texture handling (copy, mapping, transformation, etc.) • Multiple instance handling • Color Fill • Texture blending including alpha channel blending • Filtered 2D Texture mapping including rotation, scaling, distortion (for e.g. HUD display) • 3D texture mapping to allow 2.5D effects (e.g. cover flow) • Use of image data directly from specified memory area (e.g. flash) without allocation of additional memory (no video RAM copy of the data as in e.g. OpenVG) • Clipping • Window support for source and destination (different stride and width possible) • Execution-in-place of pre-prepared display lists from any memory (e.g. from flash) • RLE decoding (8, 16, 24, 32 bpp) • Primitive rendering (Lines, Rectangles, Ellipses, etc) • Retrieving performance values from the driver • CLUTs and Color Keying
Semaphore / Mutex	N/A. This can be implemented with the porting layer.
Interrupts	Used in the Drw2D driver. For more details please see *section 2.3.*

The summary of Drw2D with CPU drawing specification is described in *Table 2-2.*

Table 2-2 Summary Specification of Drw2D with CPU drawing

Items	Description
Target LSI	RH850/D1L2(H)
Main Feature	• 2D Texture handling (copy, transformation, etc.) • Color Fill • Texture blending including alpha channel blending • Window support for source and destination (different stride and width possible) • RLE decoding (8, 16, 24, 32 bpp)
Semaphore / Mutex	N/A. This can be implemented with the porting layer.
Interrupts	N/A.

2.2 Reserved word

Drw2D and D/AVE HD uses the following prefixes for avoiding confusion from other software. Prefixes of Drw2D and D/AVE HD is described in *Table 2-3*.

Table 2-3 Prefixes

Prefix	Description
R_DRW2D_*	Prefix for Drw2D Module
r_drw2d_*	
DHD_*	Prefix for D/AVE HD Module
E_DHD_*	
dhd_*	
tagdhd_*	

2.3 Interrupt Handler List

Drw2D with D/AVE HD uses the GPU2D interrupt. Interrupt handler exists in D/AVE HD Porting layer.

Table 2-4 Interrupt Handler List

No.	Interrupt Name	Interrupt Handler Name	Description
(1)	INTGPU2D0PAUSE	dhd_sys_IsrDave	Pause interrupt
(2)	INTGPU2D0SYNC	dhd_sys_IsrDave	SYNC interrupt
(3)	INTGPU2D0SP	dhd_sys_IsrDave	Stop / Stall / MBI Error interrupt.

Drw2D with CPU drawing does not use interrupt.

2.4 Error Handling

2.4.1 Return code

The error code generated by each API synchronously is notified by both return code and error callback.
The error callback is installed by R_DRW2D_ErrCallbackSet and R_DRW2D_GlobalErrCallbackSet.

2.4.2 Fatal errors of D/AVE HD

For the D/AVE HD implementation there is one error path not routed through the Drw2D API but accessible directly in the porting layer of D/AVE HD.
Within the RGL it can be found in:
vlib/macro/gpu/davehd/user/platform/d1mx/<your_os>/davehd_os_hw.c
The function dhd_debug_panic() is the entry-point for fatal GPU or D/AVE HD driver errors. In the rare case this happens the application should reset the system.

3. Function Description

3.1 Fundamental concepts

3.1.1 Device and multithreading

A *device* is the basic object, required by as parameter to all functions. This object encapsulates all the information necessary for one thread to use the drawing features: default context, HW pointers etc...
In case of a multithreaded application, where each thread should use the framework independently, it is recommended for every thread to open its own device. In that case, no further thread synchronization is needed by user regarding subsequent Drw2D API calls.

3.1.2 Context

A context is merely an internal state of the drawing engine, where all relevant information that influence the rendering process is stored:

- Framebuffer
- Background and foreground color
- Textures
- Fill mode
- Texture and vertex matrices
- Matrix transformations modes
- Blend mode and factors
- Image quality
- Line width
- Culling
- Clipping
- Striping
- Viewport

An application can create several contexts but only one is active. The pointer to the active context is stored in the *device* object.
All functions that change the context are prefixed with R_DRW2D_Ctx*.
The API interface deals with setting all context options, not with getting the previously set options. The application has to remember its settings by itself.

3.1.3 Rendering target framebuffer

Target framebuffer is a memory region where the GPU writes the result of its rendering operations. Drw2D itself does not provide any means for creating and maintaining the framebuffer, so it is the sole responsibility of user to provide it somehow, and then use the Drw2D APIs to register it for use within it. The preferred way of doing this is by using the WM API, i.e. selecting the desired Window’s framebuffer.

3.1.4 Rendering functions

Rendering is the process of rasterizing, i.e. transforming a mathematically specified geometric primitive into its bitmap representation and writing it to the framebuffer.
The following geometric primitives are supported:

- Triangle
- Rectangle
- Quad
- Line
- Polyline
- Ellipse (as well as its special case circle)
- Bezier curve

All rendering functions are prefixed with R_DRW2D_Draw*.

3.1.5 Textures

Texture is an image bitmap registered within the drawing engine with following properties:

- Memory address
- Width, Height and Pitch (Stride)
- Color format

Textures are normally obtained by converting the images from the well-known file formats to raw formats supported by the GPU. Normally, they are initially stored into some kind of permanent memory, e.g. Flash-ROM. During runtime, user is free to use textures from whatever memory accessible by the GPU, for example copying them from Flash-ROM to Video RAM prior to using for performance gain. Please refer to the HW User manual for a list of memory macros visible to the GPU.
IMPORTANT: unlike some other graphics frameworks (OpenGL, OpenVG), the drawing engine does NOT copy the texture content to some internal buffer when the texture gets registered.
Also, textures can be drawn non-UV mapped, where they are independently transformed and then clipped by the drawing primitive surface, or UV mapped (if supported by the drawing primitive), where the texture image rectangle is mapped to the drawing primitive vertices.

3.1.6 Transformation matrices

Transformation matrices are one of the fundamental concepts in computer graphics, used for accumulating transformations. Drw2D uses separate matrices for transforming drawing primitive's vertices and textures. Moreover, the texture matrix is actually a transformation matrix for blitting a textured rectangle, not a matrix modifying the mapping of a texture within primitive as in OpenGL and similar APIs.
The coordinate system is left-handed, the matrices are stored column-wise in the memory, and the current matrix in the context gets post-multiplied with the incoming one.
Using these matrices is optional.

3.1.7 Command lists

Drw2D with D/AVE HD GPU, command lists are the main interface between the driver and the GPU. They are the final output of issuing all the Drw2D API functions which do modify some internal state of the GPU. They contain instructions on how to set the control registers within the GPU and can be directly executed by it.
There is always a default command list where all the rendering operations end up, if no other command list is setup.
Several command lists can be created, but only one can be executed (processed by the GPU) at the time.
During the processing of one command list by the GPU, the CPU can prepare another command list in parallel. User can instruct the GPU to finish the processing of the command list, and wait for that to happen in a blocking or non-blocking way.
The command lists are transparent to the Drw2D user.

3.1.8 Rendering steps example

A simple rendering loop with double-buffering looks like this:

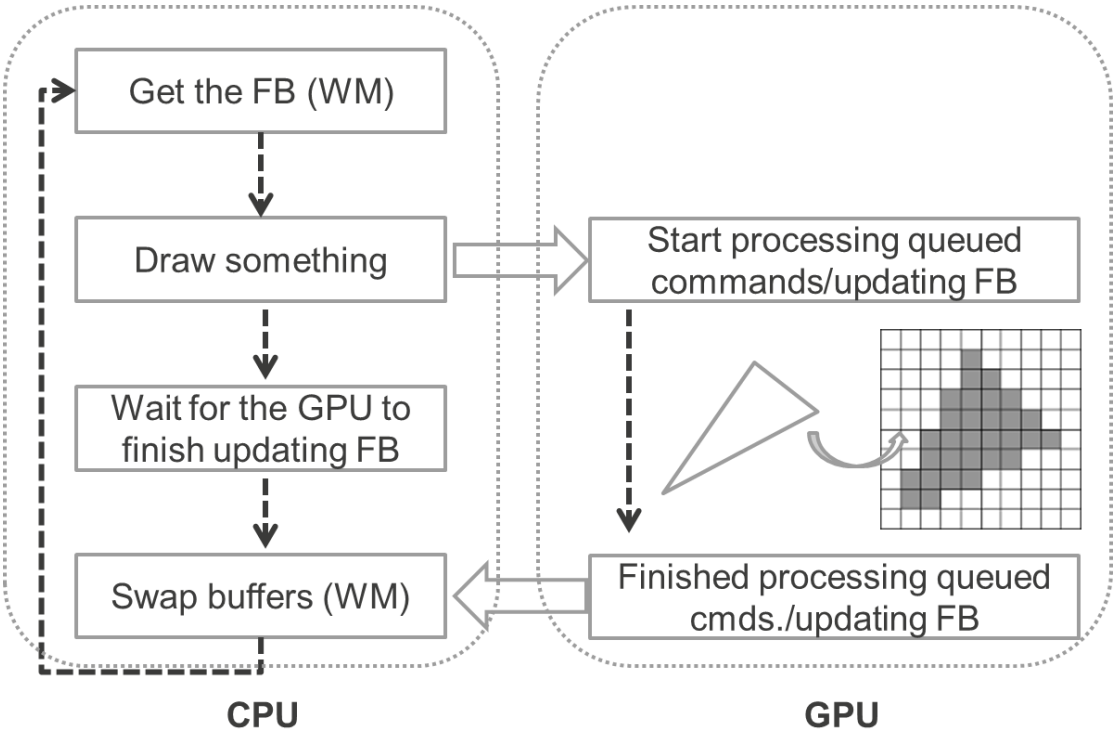

Figure 3.1 Rendering steps diagram

3.2 Using the API

3.2.1 Framework initialization

The very first API call issued by user should be R_DRW2D_Init(), followed by R_DRW2D_Open(*Unit*, *DriverUnit, InternalDevice*, *RetDevice*). In case of a multithreaded application, each thread should make a call to R_DRW2D_Open to initialize its own device, and use only this device for subsequent framework calls.
The underlying drawing engine should be initialized prior to calling this function. In case of a D/AVE HD, this means calling the D/AVE HD kernel mode API initialization functions and enable the GPU2D interrupts. Sample code is provided for these processes. Within the RGL, it can be found in vlib/app/common/dhd/r_util_dhd.c.

3.2.2 Rendering target framebuffer

Prior to issuing drawing calls, the framework needs to know the whereabouts and some properties (dimensions, color format etc.) of the target framebuffer, which can be specified with R_DRW2D_FramebufferSet(*Device*, *Framebuffer*) call.

```
/* Example: setting-up the portrait QVGA-sized framebuffer for drawing
   The actual framebuffer comes from the previously created window
   using the WM API
*/
r_drw2d_Framebuffer_t fb;
fb.Handle = R_NULL;
fb.Flags = (r_drw2d_FramebufferFlags_t)0;
fb.Buffer.Size.Width = 240;
fb.Buffer.Pitch = 256;
fb.Buffer.Size.Height = 320;
fb.Buffer.PixelFormat = R_DRW2D_PIXELFORMAT_ARGB8888;
fb.Buffer.Data = R_WM_WindowNewDrawBufGet(loc_WmDev, &loc_Window);
R_DRW2D_FramebufferSet(loc_Drw2dDev, &fb);
```

3.2.3 Textures

R_DRW2D_CtxTextureSet(*Device, TextureUnit, Texture*) registers within the context the current texture that will be used for drawing textured primitives. Drw2D with D/AVE HD supports two *TextureUnit*. The drawing functions work directly (without 'Effects' involved) however only with the texture unit 0, texture unit 1 can be used only through the effects API. Additionally, if using the effects API, always use texture unit 0 for single texture operations. Texture unit 1 may only be used if texture units 0 is already in use.
Apart from texture image dimensions and color format, user can specify additional texture usage flags concerning texture wrapping, filtering (i.e. bilinear), perspective correction, run-length encoded image bits, swizzling and virtual tiling. Once set, the texture remains active until substituted with another one.

```
/* Example: setting-up the texture
   The actual texture bits come for example from FLASH-ROM.
*/
r_drw2d_Texture_t tex;
tex.Handle = R_NULL;
tex.Flags = R_DRW2D_TEX_NONE;
tex.Buffer.Pitch = Pitch;
tex.Buffer.Size.Width = Width;
tex.Buffer.Size.Height = Height;
tex.Buffer.PixelFormat = R_DRW2D_PIXELFORMAT_ARGB8888;
tex.Buffer.Data = texture_image_bits;
R_DRW2D_CtxTextureSet(drw2d_dev, 0, &tex);
```

To use texture with drawing primitives, user must call the R_DRW2D_CtxFillMode(*Device*, R_DRW2D_FILLMODE_TEXTURE).

3.2.3.1 Texture flags

R_DRW2D_TEX_WRAPU and R_DRW2D_TEX_WRAPV enable repeating the texture bitmap, in X and Y direction respectively, when filling the area of a drawn shape.
R_DRW2D_TEX_BILINEAR enables the bilinear filtering when writing the texture pixels to the framebuffer.
R_DRW2D_TEX_RLE notifies the runtime that the texture bitmap is RLE compressed, where supported by the GPU HW.
R_DRW2D_TEX_PERSPECTIVE enables texture distortion according to the perspective vertex matrix.
R_DRW2D_TEX_SWIZZLE indicates a swizzled texture.
R_DRW2D_TEX_VT indicates a virtual tiled texture.

3.2.4 Transformation matrices

There are two independent transformation matrices stored in the context: vertex matrix and texture matrix.
The functions influencing these matrices have the form: R_DRW2D_Ctx[Texture]_*Operation*, where *Operation* can be one of the following: *Identity*, *Translate*, *Rotate*, *Scale* and *Transform*. *Identity*, *Translate*, *Rotate* and *Scale* change the current transformation matrix by multiplying it with the matrix created on the fly from the given arguments. *Transform* is the most general case where the user provides the fully defined arbitrary matrix to replace the current matrix.
There are two transformation calls specific to vertex matrix only: *Rotate3D* and *Frustum*.
Another function available is R_DRW2D_CtxMatrix. This function retrieves either the vertex matrix or the texture matrix from the driver. This gives the opportunity to manipulate the matrix for transformations not yet supported by the driver or simply for restoration a previous driver state. To transfer the matrixes back into the driver, use *Transform*.
All multiplications are so-called post-multiplications, where the existing matrix is on the left, and the operand matrix comes on the right side of the multiplication expression.
Example: matrix operations implementation – forming the rotation matrix and multiplying it with the current matrix in case of R_DRW2D_CtxRotate:

```
(a e i m)       (cos(a)  -sin(a)  0  0)      (a*cos(a) + e*sin(a), a*-sin(a) + e*cos(a), i, m )
(b f j  n)  *   (sin(a)  cos(a)   0  0)  =   (b*cos(a) + f*sin(a), b*-sin(a) + f*cos(a),  j, n )
(c g k o)       (0       0        1  0)      (c*cos(a) + g*sin(a), c*-sin(a) + g*cos(a), k, o )
(d h l  p)      (0       0        0  1)      (d*cos(a) + h*sin(a), d*-sin(a) + h*cos(a), l, p )
```

3.2.5 Effects

3.2.5.1 Introduction

The effects API exposes the functionality usually implemented by setting up some kind of a multi-stage GPU drawing pipeline. Particularly for D/AVE HD, its 3-stage color unit is used.
The effects are always related to the currently drawn geometric primitive, and constrained by its area, just as textures.
Every stage can be assigned a single effect, and the output of the previous stage can be used as the input of the following one. Not all effects can be stacked in an arbitrary order.
The output of a stage is always an RGBA quad (vector). The output of a stage is obtained by calculating the effect assigned to it. If the effect produces a scalar value, it will be repeated in every RGBA vector component.

3.2.5.2 Describing an effect

```
typedef struct
{
    r_drw2d_EffectName_t  Name;    // Describes the concrete operation taking place
    r_drw2d_EffectParam_t Args[4]; // Describes the arguments for the operation
} r_drw2d_EffectStage_t;
```

A single effect is described using the r_drw2d_EffectStage_t structure above. The effect is determined by an r_drw2d_EffectStage_t object's *Name* element. The parameters for the specified effect are given by the *Args* array. The number and type of parameters depends on the used effect.
An effect parameter is described using the r_drw2d_EffectParam_t structure as given below:

```
typedef struct
{
    r_drw2d_EffectParamSource_t Source;
    union {
        struct {
            union {
                uint32_t TextureUnit;   /* sampled texture */
                uint32_t ConstantColor; /* directly specified */
            } Source;
            r_drw2d_EffectColorParamOperand_t Operand;
        } Color;
        r_drw2d_FixedP_t Constant;      /* used with constant alpha */
        r_drw2d_Point_t Point;          /* used with gradient */
    } Param;
} r_drw2d_EffectParam_t;
```

Which part of the *Param* and *Source* union inside of an object is valid, is specified by the *Source* element. There are 5 parameter types available. Each of them renders one combination of the *Param* and *Source* union's elements valid as follows:

Table 3-1 Combination of the parameter and source union's

Source value	Valid Param element	Valid Color element	Description
R_DRW2D_EFFECT_SOURCE_TEXTURE_UNIT	Color	TextureUnit	Samples color values from the given texture unit.
		Operand	Determines in what way the color parameter should be used in calculation, i.e. the whole RGBA or just the A channel (repeated in other channels as AAAA), as is or inverted.
R_DRW2D_EFFECT_SOURCE_CONSTANT_COLOR	Color	ConstantColor	Uses the directly specified color value in ARGB8888 format.
		Operand	As Operand above.
R_DRW2D_EFFECT_SOURCE_CONSTANT	Constant	Invalid	Uses the given scalar (for the constant alpha effect).
R_DRW2D_EFFECT_SOURCE_POINT	Point	Invalid	Uses the given two-dimensional vector (for the gradient effect).
R_DRW2D_EFFECT_SOURCE_PREV_STAGE	Color	Source union is ignored	Uses the color values from the previous effect stage.
		Operand	As Operand above.

3.2.5.3 Setting up an effect

An effect is set up by creating an instance of the r_drw2d_EffectStage_t. Then, its Name element has to be set to the desired effects name. According to the chosen effect, the instance's Args element has to be set. The number of Args entries that have to be set depends on the chosen effect, just as the valid values for the particular Args entry's Source element.
The following table shows, which Source values can be chosen for each effect and how many Args entries have to be set up (which r_drw2d_EffectParam_t elements are rendered valid by the particular Source values can be learned from the table above):

Table 3-2 Args entries and valid source values for effect

Effect	Args entries	Valid Source values
R_DRW2D_EFFECT_REPLACE	1	R_DRW2D_EFFECT_SOURCE_TEXTURE_UNIT R_DRW2D_EFFECT_SOURCE_CONSTANT_COLOR R_DRW2D_EFFECT_SOURCE_PREV_STAGE
R_DRW2D_EFFECT_MODULATE	2	R_DRW2D_EFFECT_SOURCE_TEXTURE_UNIT R_DRW2D_EFFECT_SOURCE_CONSTANT_COLOR R_DRW2D_EFFECT_SOURCE_PREV_STAGE
R_DRW2D_EFFECT_ADD	2	R_DRW2D_EFFECT_SOURCE_TEXTURE_UNIT R_DRW2D_EFFECT_SOURCE_CONSTANT_COLOR R_DRW2D_EFFECT_SOURCE_PREV_STAGE
R_DRW2D_EFFECT_ADD_SIGNED	2	R_DRW2D_EFFECT_SOURCE_TEXTURE_UNIT R_DRW2D_EFFECT_SOURCE_CONSTANT_COLOR R_DRW2D_EFFECT_SOURCE_PREV_STAGE
R_DRW2D_EFFECT_SUBTRACT	2	R_DRW2D_EFFECT_SOURCE_TEXTURE_UNIT R_DRW2D_EFFECT_SOURCE_CONSTANT_COLOR R_DRW2D_EFFECT_SOURCE_PREV_STAGE
R_DRW2D_EFFECT_INTERPOLATE	3	R_DRW2D_EFFECT_SOURCE_TEXTURE_UNIT R_DRW2D_EFFECT_SOURCE_CONSTANT_COLOR R_DRW2D_EFFECT_SOURCE_PREV_STAGE
R_DRW2D_EFFECT_DOT3	2	R_DRW2D_EFFECT_SOURCE_TEXTURE_UNIT R_DRW2D_EFFECT_SOURCE_CONSTANT_COLOR R_DRW2D_EFFECT_SOURCE_PREV_STAGE
R_DRW2D_EFFECT_CONSTANT_ALPHA	1	R_DRW2D_EFFECT_SOURCE_CONSTANT
R_DRW2D_EFFECT_GRADIENT	2 + 2 (Requires exactly 2 POINT and 2 CONSTANT entries)	R_DRW2D_EFFECT_SOURCE_POINT R_DRW2D_EFFECT_SOURCE_CONSTANT

After the instance of the r_drw2d_EffectStage_t structure was set up properly, the effect can be enabled by calling R_DRW2D_CtxEffectsSet for the instance. It takes a pointer to the instance and the number of effects in the instance as parameters. Thereafter, all render calls (R_DRW2D_Draw* and R_DRW2D_TextureBlit) will be executed using the set effect. The effects can be disabled by a single call to R_DRW2D_CtxEffectsDelete.
It is important that the r_drw2d_EffectStage_t instance stays readable until the following R_DRW2D_CtxEffectsDelete call. For example, creating the instance on the stack and leaving the corresponding function may lead to overwriting of the effect parameters when the stack grows again.
A pointer to the structure is used internally until R_DRW2D_CtxEffectsDelete was called.
The used effects can be changed individually by a call to R_DRW2D_CtxEffectsUpdate. This is useful if a certain parameter of the effect in use must be changed during the rendering calls.

3.2.5.4 Arithmetic effects

Following table shows the formula and output of each effects.

Table 3-3 Kind of arithmetic effects

	Effect	Formula	Stage output
(1)	R_DRW2D_EFFECT_REPLACE	S = op	Unchanged value.
(2)	R_DRW2D_EFFECT_MODULATE	S = op1 * op2	Member wise product.
(3)	R_DRW2D_EFFECT_ADD	S = op1 + op2	Member wise addition.
(4)	R_DRW2D_EFFECT_ADD_SIGNED	S = op1 + op2 – 0.5	Member wise signed addition.
(5)	R_DRW2D_EFFECT_SUBTRACT	S = op1 - op2	Member wise subtraction.
(6)	R_DRW2D_EFFECT_INTERPOLATE		Interpolation of the input parameters with the third one.
(7)	R_DRW2D_EFFECT_DOT3		Scalar Multiplication of the vectors.

The details of each effect are explained below.

(1) Replace effect

The replace effect reads the unchanged value from the source specified by the *Source* member of the effect's parameter description. The *Operand* member determines if all channels (RGBA) or just the alpha channel (AAAA) is read in from the source. The channels can also be read in inverted.

Since the replace effect does not change the read in values, it is usually used to insert values into the effect stages that can be processed in the following stage, e.g. in a constant alpha effect (see Constant alpha effect example below). However, with an appropriately set *Operand* member, it can be used to invert the color channels and create a simple inversion effect.

An example for the use of the replace effect as a simple inversion effect for a texture from texture unit 0 is given below:

```
static r_drw2d_EffectStage_t Effects[1];
Effects[0].Name = R_DRW2D_EFFECT_REPLACE;
Effects[0].Args[0].Source = R_DRW2D_EFFECT_SOURCE_TEXTURE_UNIT;
Effects[0].Args[0].Param.Color.Source.TextureUnit = 0;
Effects[0].Args[0].Param.Color.Operand = R_DRW2D_EFFECT_COLOR_OPERAND_ONE_MINUS_RGBA;
R_DRW2D_CtxEffectsSet(g_drw2d_dev, Effects, 1);
```

The color data is inversed by setting the Operand element to R_DRW2D_EFFECT_COLOR_OPERAND_ONE_MINUS_RGBA. This will invert all 4 color channels read in from texture unit 0.

(2) Modulate effect

The modulate effect multiplies color values of two given sources pixel wise. Since the effect multiplies two sources, it needs two *Color* arguments in the Args member.
For example, the effect can be used to colorize a monochrome texture (e.g. a font texture). A code snippet to colorize a texture assigned to texture unit 0 in a yellow color is given below:

```
static r_drw2d_EffectStage_t Effects[1];
Effects[0].Name = R_DRW2D_EFFECT_MODULATE;
Effects[0].Args[0].Source = R_DRW2D_EFFECT_SOURCE_TEXTURE_UNIT;
Effects[0].Args[0].Param.Color.Source.TextureUnit = 0;
Effects[0].Args[0].Param.Color.Operand = R_DRW2D_EFFECT_COLOR_OPERAND_RGBA;
Effects[0].Args[1].Source = R_DRW2D_EFFECT_SOURCE_CONSTANT_COLOR;
Effects[0].Args[1].Param.Color.Source.ConstantColor = 0xffffff00;
Effects[0].Args[1].Param.Color.Operand = R_DRW2D_EFFECT_COLOR_OPERAND_RGBA;
R_DRW2D_CtxEffectsSet(g_drw2d_dev, Effects, 1);
```

The effect uses texture unit 0 and a constant color as operands. Texture unit 0 holds the monochrome texture, whereas the constant color is stored in the ConstantColor element of the Source union.
This effect can be used for fading in/out the textures with alpha value change by ConstatntColor.

Another use case of the modulate effect is the "screen" effect. The screen effect is the opposite effect to multiply. It will brighten the texture instead of darkening it and can be used, for example, to create a highlighting effect.
How much each part of the texture should be brightened can be defined pixel wise in a separate monochrome texture. Both textures are then inverted, multiplied and inverted again in the effect stage. The screen effect can be implemented with the following effect settings:

```
static r_drw2d_EffectStage_t Effects[2];
Effects[0].Name = R_DRW2D_EFFECT_MODULATE;
Effects[0].Args[0].Source = R_DRW2D_EFFECT_SOURCE_TEXTURE_UNIT;
Effects[0].Args[0].Param.Color.Source.TextureUnit = 0;
Effects[0].Args[0].Param.Color.Operand = R_DRW2D_EFFECT_COLOR_OPERAND_ONE_MINUS_RGBA;
Effects[0].Args[1].Source = R_DRW2D_EFFECT_SOURCE_TEXTURE_UNIT;
Effects[0].Args[1].Param.Color.Source.TextureUnit = 1;
Effects[0].Args[1].Param.Color.Operand = R_DRW2D_EFFECT_COLOR_OPERAND_ONE_MINUS_RGBA;
Effects[1].Name = R_DRW2D_EFFECT_MODULATE;
Effects[1].Args[0].Source = R_DRW2D_EFFECT_SOURCE_PREV_STAGE;
Effects[1].Args[0].Param.Color.Operand = R_DRW2D_EFFECT_COLOR_OPERAND_ONE_MINUS_RGBA;
Effects[1].Args[1].Source = R_DRW2D_EFFECT_SOURCE_CONSTANT_COLOR;
Effects[1].Args[1].Param.Color.Source.ConstantColor = 0xffffffff;
Effects[1].Args[1].Param.Color.Operand = R_DRW2D_EFFECT_COLOR_OPERAND_RGBA;
R_DRW2D_CtxEffectsSet(g_drw2d_dev, &Effects[0], 2);
```

The modulate effect on the second effect stage uses color data from the previous stage. Therefore, its first operand's Source element is set to R_DRW2D_EFFECT_SOURCE_PREV_STAGE. Its Operand element is set to the inversion mode (R_DRW2D_EFFECT_COLOR_OPERAND_ONE_MINUS_RGBA) to invert the data from the previous stage. Note that the Param.Color.Source element is not set at all because it is ignored anyway. Also, note that the second modulate stage is just used as an inversion effect. Therefore, its second parameter is constant white color.

(3) Add effect

The add effect adds the color values of two given sources pixel wise. Since the effect adds two sources, it also requires two *Color* arguments in the Args member.
The effect can for example be used for lighting effects (e.g. glowing effects) or other effects that should brighten the processed image data. Note that the result of two added color values of a single channel will be clamped to 255.
To add a yellow glowing effect to a texture, the effect can be used in combination with the modulate effect from above. The modulate effect is used to colorize a monochrome texture in a yellow color. The add effect is then used in the second effect stage to add the output of the previous effect stage to the texture from texture unit 1. Therefore, the first Args entry's *Source* element is set to R_DRW2D_EFFECT_SOURCE_PREV_STAGE. The second Args entry's *Source* element is set up to use texture unit 1.

```
static r_drw2d_EffectStage_t Effects[2];
Effects[0].Name = R_DRW2D_EFFECT_MODULATE;
Effects[0].Args[0].Source = R_DRW2D_EFFECT_SOURCE_CONSTANT_COLOR;
Effects[0].Args[0].Param.Color.Source.ConstantColor = 0xffffff00;
Effects[0].Args[0].Param.Color.Operand = R_DRW2D_EFFECT_COLOR_OPERAND_RGBA;
Effects[0].Args[1].Source = R_DRW2D_EFFECT_SOURCE_TEXTURE_UNIT;
Effects[0].Args[1].Param.Color.Source.TextureUnit = 0;
Effects[0].Args[1].Param.Color.Operand = R_DRW2D_EFFECT_COLOR_OPERAND_RGBA;
Effects[1].Name = R_DRW2D_EFFECT_ADD;
Effects[1].Args[0].Source = R_DRW2D_EFFECT_SOURCE_PREV_STAGE;
Effects[1].Args[0].Param.Color.Source.ConstantColor = 0xffffffff;
Effects[1].Args[0].Param.Color.Operand = R_DRW2D_EFFECT_COLOR_OPERAND_RGBA;
Effects[1].Args[1].Source = R_DRW2D_EFFECT_SOURCE_TEXTURE_UNIT;
Effects[1].Args[1].Param.Color.Source.TextureUnit = 1;
Effects[1].Args[1].Param.Color.Operand = R_DRW2D_EFFECT_COLOR_OPERAND_RGBA;
R_DRW2D_CtxEffectsSet(g_drw2d_dev, &Effects[0], 2);
```

(4) Add signed effect

The add effect adds the color values of two given sources pixel wise and then subtracts 0.5 (respectively 127). Since the effect uses two sources, it also requires two *Color* arguments in the Args member.
Compared to the add effect, the add signed effect tends to darken the color data.
To create an add sign effect that combines two textures, the following code snippet can be used:

```
static r_drw2d_EffectStage_t Effects[2];
Effects[0].Name = R_DRW2D_EFFECT_MODULATE;
Effects[0].Args[0].Source = R_DRW2D_EFFECT_SOURCE_TEXTURE_UNIT;
Effects[0].Args[0].Param.Color.Source.TextureUnit = 1;
Effects[0].Args[0].Param.Color.Operand = R_DRW2D_EFFECT_COLOR_OPERAND_RGBA;
Effects[0].Args[1].Source = R_DRW2D_EFFECT_SOURCE_CONSTANT_COLOR;
Effects[0].Args[1].Param.Color.Source.ConstantColor = 0xff0000ff;
Effects[0].Args[1].Param.Color.Operand = R_DRW2D_EFFECT_COLOR_OPERAND_RGBA;
Effects[1].Name = R_DRW2D_EFFECT_ADD_SIGNED;
Effects[1].Args[0].Source = R_DRW2D_EFFECT_SOURCE_TEXTURE_UNIT;
Effects[1].Args[0].Param.Color.Source.TextureUnit = 0;
Effects[1].Args[0].Param.Color.Operand = R_DRW2D_EFFECT_COLOR_OPERAND_RGBA;
Effects[1].Args[1].Source = R_DRW2D_EFFECT_SOURCE_PREV_STAGE;
Effects[1].Args[1].Param.Color.Operand = R_DRW2D_EFFECT_COLOR_OPERAND_RGBA;
R_DRW2D_CtxEffectsSet(g_drw2d_dev, &Effects[0], 2);
```

The modulate effect is again used to colorize a (monochrome) texture in a blue color here. In the second effect stage, the color values from the previous stage are an add signed combined with the texture data from texture unit 0.

(5) Subtract effect

The subtract effect subtracts the color values of two given sources pixel wise. Since the effect uses two sources, it also requires two *Color* arguments in the Args member.
The following code snippet can be used to create a subtract effect:

```
static r_drw2d_EffectStage_t Effects[2];
Effects[0].Name = R_DRW2D_EFFECT_MODULATE;
Effects[0].Args[0].Source = R_DRW2D_EFFECT_SOURCE_TEXTURE_UNIT;
Effects[0].Args[0].Param.Color.Source.TextureUnit = 1;
Effects[0].Args[0].Param.Color.Operand = R_DRW2D_EFFECT_COLOR_OPERAND_RGBA;
Effects[0].Args[1].Source = R_DRW2D_EFFECT_SOURCE_CONSTANT_COLOR;
Effects[0].Args[1].Param.Color.Source.ConstantColor = 0x00ffff00;
Effects[0].Args[1].Param.Color.Operand = R_DRW2D_EFFECT_COLOR_OPERAND_ONE_MINUS_RGBA;
Effects[1].Name = R_DRW2D_EFFECT_SUBTRACT;
Effects[1].Args[0].Source = R_DRW2D_EFFECT_SOURCE_TEXTURE_UNIT;
Effects[1].Args[0].Param.Color.Source.TextureUnit = 0;
Effects[1].Args[0].Param.Color.Operand = R_DRW2D_EFFECT_COLOR_OPERAND_RGBA;
Effects[1].Args[1].Source = R_DRW2D_EFFECT_SOURCE_PREV_STAGE;
Effects[1].Args[1].Param.Color.Operand = R_DRW2D_EFFECT_COLOR_OPERAND_RGBA;
R_DRW2D_CtxEffectsSet(g_drw2d_dev, &Effects[0], 2);
```

The given effect configuration uses the modulate effect on the first stage to colorize the texture assigned to texture unit 1 in an inverted yellowish color. Therefore, the Operand is set to R_DRW2D_EFFECT_COLOR_OPERAND_ONE_MINUS _RGBA. The color is inverted here, since when subtracting, the yellow part will not be removed.
The subtract effect is then used in the second stage on texture unit 0. To subtract the color data from the previous effect stage, the Source element of Args[1] is set to R_DRW2D_EFFECT_SOURCE_PREV_STAGE.

(6) Interpolate effect

The interpolate effects interpolates between the color vectors of the first two given sources. The weight for the interpolation is taken from the third given source. Since the effect uses three sources, it also requires three *Color* arguments in the Args member.
The following code snippet creates an interpolate effect by interpolating between two constant colors, using the texture assigned to texture unit 0 as weight:

```
static r_drw2d_EffectStage_t Effects[1];
Effects[0].Name = R_DRW2D_EFFECT_INTERPOLATE;
Effects[0].Args[0].Source = R_DRW2D_EFFECT_SOURCE_CONSTANT_COLOR;
Effects[0].Args[0].Param.Color.Source.ConstantColor = 0xff0000ff;
Effects[0].Args[0].Param.Color.Operand = R_DRW2D_EFFECT_COLOR_OPERAND_RGBA;
Effects[0].Args[1].Source = R_DRW2D_EFFECT_SOURCE_CONSTANT_COLOR;
Effects[0].Args[1].Param.Color.Source.ConstantColor = 0xffff0000;
Effects[0].Args[1].Param.Color.Operand = R_DRW2D_EFFECT_COLOR_OPERAND_RGBA;
Effects[0].Args[2].Source = R_DRW2D_EFFECT_SOURCE_TEXTURE_UNIT;
Effects[0].Args[2].Param.Color.Source.TextureUnit = 0;
Effects[0].Args[2].Param.Color.Operand = R_DRW2D_EFFECT_COLOR_OPERAND_RGBA;
R_DRW2D_CtxEffectsSet(g_drw2d_dev, &Effects[0], 1);
```

The color values from the source in Args[2] determine the influence of the vector elements from the other two sources. The color values specify the influence of the color values from the first source (Args[0]). The higher the value, the higher is the influence. A value of 255 means, that Args[1] has no influence at all.

(7) Dot3 effect

The dot3 effect interprets the color values of two sources as three-dimensional vectors and computes their dot product pixel wise. Since the effect uses two sources, it requires two *Color* arguments in the Args member.
The color values of the form (A,R,G,B) are interpreted as a three-dimensional vector (R,G,B). The A value is ignored.
The effect can be used create a bump mapping effect. Therefore, the effect is applied to a texture that contains the normal vectors of a surface. The lighting vector is then set as a constant color, as shown in the following code snippet:

```
static r_drw2d_EffectStage_t Effects[1];
Effects[0].Name = R_DRW2D_EFFECT_DOT3;
Effects[0].Args[0].Source = R_DRW2D_EFFECT_SOURCE_TEXTURE_UNIT;
Effects[0].Args[0].Param.Color.Source.TextureUnit = 0;
Effects[0].Args[0].Param.Color.Operand = R_DRW2D_EFFECT_COLOR_OPERAND_RGBA;
Effects[0].Args[1].Source = R_DRW2D_EFFECT_SOURCE_CONSTANT_COLOR;
Effects[0].Args[1].Param.Color.Source.ConstantColor = 0x0000b5b5;
Effects[0].Args[1].Param.Color.Operand = R_DRW2D_EFFECT_COLOR_OPERAND_RGBA;
R_DRW2D_CtxEffectsSet(g_drw2d_dev, &Effects[0], 1);
```

Since the dot product of two vectors is a scalar, the resulting color data will hold the scalar in every component. However, the effect can be used in combination with effects like modulate to introduce constant colors or texture color values in the second effect stage.

3.2.5.5 Miscellaneous effects

(1) Constant alpha effect

The effect requires one *Constant* argument in the Args member, which is the constant alpha value applied to the color values by the effect. A valid alpha value lies in the range between 0.0 and 1.0.
This effect is blended after blending with R_DRW2D_CtxBlendMode.
Each pixel is obtained by the following formular:

```
DstAlpha = ConstAlpha * SrcAlpha + (1 - ConstAlpha) * DstAlpha
DstColor = ConstAlpha * SrcColor + (1 - ConstAlpha) * DstColor
```

The following code snippets uses the constant alpha effect to set a texture’s alpha value to 0.5:

```
static r_drw2d_EffectStage_t Effects[2];
Effects[0].Name = R_DRW2D_EFFECT_REPLACE;
Effects[0].Args[0].Source = R_DRW2D_EFFECT_SOURCE_TEXTURE_UNIT;
Effects[0].Args[0].Param.Color.Source.TextureUnit = 0;
Effects[0].Args[0].Param.Color.Operand = R_DRW2D_EFFECT_COLOR_OPERAND_RGBA;
Effects[1].Name = R_DRW2D_EFFECT_CONSTANT_ALPHA;
Effects[1].Args[0].Source = R_DRW2D_EFFECT_SOURCE_CONSTANT;
Effects[1].Args[0].Param.Constant = R_DRW2D_2X(0.5f);
R_DRW2D_CtxEffectsSet(g_drw2d_dev, &Effects[0], 2);
```

The first effects stage is filled with the desired texture using the replace effect, the second one uses the constant alpha effect to set all the texture’s alpha values to 0.5.

(2) Gradient effect

The gradient effect alpha blends color values according to a given linear gradient. The gradient is specified by two points and two alpha values. Thus, the effect requires two *Point* and two *Constant* values in its Args member. It is crucial, to keep the arguments (Args array entries) in the correct order. A valid alpha value lies in the range between 0.0 and 1.0. The order of the gradient effects arguments must be:

Table 3-4 Order of the gradient effects arguments

Args entry	Param element	Description
Args[0]	Point	Starting point of the gradient. The starting point is relative to the rendered geometry. It affects the vertex matrix.
Args[1]	Point	Ending point of the gradient. The ending point is relative to the rendered geometry. It affects the vertex matrix.
Args[2]	Constant	Alpha value at the starting point. Note that the alpha value's slope, computed for the distance between the starting and ending point, is also applied in front of the starting point.
Args[3]	Constant	Alpha value at the ending point. Note that the alpha value's slope, computed for the distance between the starting and ending point, is also applied behind the starting point.

The following code snippet shows an example of using the gradient effect to alpha blend a texture:

```
static r_drw2d_EffectStage_t Effects[2];
Effects[0].Name = R_DRW2D_EFFECT_REPLACE;
Effects[0].Args[0].Source = R_DRW2D_EFFECT_SOURCE_TEXTURE_UNIT;
Effects[0].Args[0].Param.Color.Source.TextureUnit = 0;
Effects[0].Args[0].Param.Color.Operand = R_DRW2D_EFFECT_COLOR_OPERAND_RGBA;
Effects[1].Name = R_DRW2D_EFFECT_GRADIENT;
Effects[1].Args[0].Source = R_DRW2D_EFFECT_SOURCE_POINT;
Effects[1].Args[0].Param.Point.X = R_DRW2D_2X(0);
Effects[1].Args[0].Param.Point.Y = R_DRW2D_2X(0);
Effects[1].Args[1].Source = R_DRW2D_EFFECT_SOURCE_POINT;
Effects[1].Args[1].Param.Point.X =  R_DRW2D_2X(100);
Effects[1].Args[1].Param.Point.Y =  R_DRW2D_2X(100);
Effects[1].Args[2].Source = R_DRW2D_EFFECT_SOURCE_CONSTANT;
Effects[1].Args[2].Param.Constant = R_DRW2D_2X(0.0);
Effects[1].Args[3].Source = R_DRW2D_EFFECT_SOURCE_CONSTANT;
Effects[1].Args[3].Param.Constant = R_DRW2D_2X(1.0);
R_DRW2D_CtxEffectsSet(g_drw2d_dev, &Effects[0], 2);
```

The replace effect is used to insert color values from texture unit 0 into the effect stage 0. These color values are then used with the gradient effect. The gradient is configured to start at (0,0) with an alpha value of 0.0 and end at (100,100) with an alpha value of 1.0.
Note that the gradient's start and end points are given relatively to the geometry and are also transformed by the vertex matrix.
Gradient effect is not valid when transform mode is R_DRW2D_TRANSFORM_3D.

3.2.6 Drawing primitives

3.2.6.1 General

Geometric primitives are fast and reliable basic building blocks for generating complex graphical content. In case of Drw2D with D/AVE HD there is a direct HW support for most of them, i.e. they get drawn in HW accelerated way.
All primitives can be drawn in two coloring modes: solid fill and texture fill, controlled by the R_DRW2D_CtxFillMode call. In case of a texture mode, either the texture specified in unit 0 is used directly, or any number of the textures (specified in 0 and other texture units) through the Effects API can be used.
Furthermore, the R_DRW2D_CtxTransformMode call specifies if the current vertex transformation matrix should influence the primitive's coordinates or not.
Some of the primitives provide two modes for applying textures: non-UV and UV, the latter one invoked via R_DRW2D_*UV() functions.
UV mode means that the texture is mapped to the primitive's geometry, in the same way as with OpenGL, Direct3D and similar APIs.
In non-UV mode, texture is drawn independently from primitive. The primitive's surface should be understood as a clipping geometry, or a stencil for the underlying independently drawn texture.

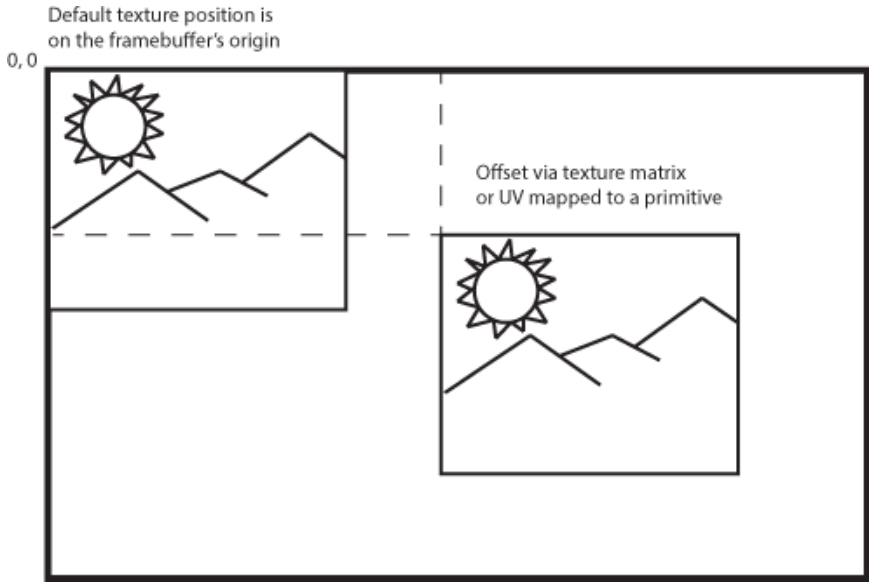

Figure 3.2 Example of texture transform

The texture itself can be (but doesn't need to be) transformed by the separate current texture matrix, governed by the R_DRW2D_CtxTextureTransformMode function. In case of non-UV mode, the matrix influences the drawing position of the texture on the framebuffer (the origin of a texture always being aligned with the target framebuffer origin), and in UV mode it transforms the UV coordinates of the texture mapped to the primitive.
In case of a non-UV mode, applying texture matrix yields a texture_image_pixel/framebuffer_pixel ratio. This means that, in case of e.g. scaling with factors less than one, the overall bitmap displayed will be greater than the original bitmap. The translation part of the matrix represents the offset from the origin of the framebuffer in pixels.

3.2.6.2 Rectangles

R_DRW2D_DrawRect and R_DRW2D_DrawRectUV draw a rectangle in with or without the UV mapping for drawing in textured mode. To quickly (in terms of coding) blit a texture somewhere on the framebuffer, the R_DRW2D_TextureBlit function can be used, otherwise this could be achieved by aligning the texture manually with a rectangle drawn with R_DRW2D_DrawRect, or using the R_DRW2D_DrawRectUV and supplying a UV mapping.

3.2.6.3 Triangles

When drawing triangles with R_DRW2D_DrawTriangles[UV](*Device*, *Points*, *Count*, *EdgeFlags*), the Count / 3 triangles will be drawn, i.e. every triangle is specified with all 3 points. Additionally, the antialiasing can be applied on edge basis, governed by the *EdgeFlags* parameter, or NULL in which case no antialiasing will occur.

3.2.6.4 Lines and polylines

The only difference between R_DRW2D_DrawLines and R_DRW2D_DrawPolyline is that in case of lines, all line segments should be provided fully defined, and with polylines, the segments are treated as connected, so after the 1st segment, every other segment is provided with just one (end) point.
The rendered line properties can be influenced by using the R_DRW2D_CtxLineStyle.

3.2.7 Convolution

3.2.7.1 Introduction

Convolution is an image processing feature for applying a small matrix (a.k.a kernel) over the current texture to achieve effects like blurring, sharpening, embossing, edge-detection, and more.
The Drw2D comes with a few fixed 1D and 2D kernel presets for gauss blurring, sobel filtering (edge detection), embossing and sharpening. Additionally, the user can generate custom kernels for convolution.

3.2.7.2 API Overview

R_DRW2D_DrawRectConvolve*N*dx(*Device*, *Rect*, *TextureOffX*, *TextureOffY*) draws the texture with applied *N*-dimensional convolution preset set with R_DRW2D_CtxConvolutionKernelPreset*N*d(*Device*, *Preset*), where *N* can be 1 or 2, in the area specified by the *Rect*. If desired, the application of the kernel can be limited to the texture subregion, which start can be specified with *TextureOffX* and *TextureOffY.*
R_DRW2D_DrawRectConvolve (*Device*, *Rect*, *TextureOffX*, *TextureOffY*) draws the texture with applied user defined convolution kernel set with R_DRW2D_CtxConvolutionKernel (*Device*, *Kernel*), in the area specified by the *Rect*. If desired, the application of the kernel can be limited to the texture subregion, which start can be specified with *TextureOffX* and *TextureOffY.*
R_DRW2D_CtxConvolutionMode (*Device*, *Mode*) sets the convolution mode. A texture can be convoluted and drawn trimmed to its actual size (R_DRW2D_CONVMODE_TRIMMED) or using a bleeding effect (R_DRW2D_CONVMODE_BLEEDING). Using the bleeding mode, the kernel is also applied to transparent pixels outside of the texture in order to generate a bleeding effect (e.g. when using Gaussian blur kernels). For correct render results, the bleeding mode needs a pre-multiplied alpha blend mode.
R_DRW2D_GetGaussKernel (*Device, Kernel, Width, Height, Sigma*) generates a gauss kernel of size *Width* * *Height* with σ=*Sigma*. The coefficients are stored in *Kernel*.
R_DRW2D_CtxConvolutionKernel (*Device, Kernel*) sets a user defined kernel for convolution.

3.2.7.3 Setting up a custom kernel

A kernel in Drw2D is described using the r_drw2d_ConvKernel_t structure. The structure contains:

- *Coeff*
  The kernel's coefficients.
- *Channel*
  The color channels that are output by the filter.
- *Width, Height*
  The kernel's dimension.
- *Bias*
  Bias value added to the resulting color channel values (range: -1.0 to 1.0).

The following code fragment shows how to set up a kernel for convolution:

```
static const r_drw2d_FixedP_t loc_kernel_edge_coeff[3*3] =
{
    R_DRW2D_2X( 0.0), R_DRW2D_2X( 1.0), R_DRW2D_2X( 0.0),
    R_DRW2D_2X( 1.0), R_DRW2D_2X(-4.0), R_DRW2D_2X( 1.0),
    R_DRW2D_2X( 0.0), R_DRW2D_2X( 1.0), R_DRW2D_2X( 0.0),
};
static const r_drw2d_ConvKernel_t loc_kernel_config =
{
    loc_kernel_edge_coeff,
    R_DRW2D_CONVKERNEL_COLOR_CHANNEL_RGB,
    3,
    3,
    R_DRW2D_2X(0.5)
};
```

The coefficients are given as an array of fixed point values in usual matrix style. A pointer to the coefficient array is used in the r_drw2d_ConvKernel_t object. Since we want to do a kind of edge detection, we limit the convolution output to the color channels only (R_DRW2D_CONVKERNEL_COLOR_CHANNEL_RGB). Since application of the kernel may lead to negative values, we push the results above 0 by setting the bias to 0.5. For 8 bit color channels, this will lead to an addition of 128 to each color channel value.
The kernel can then be set and the current texture can be convoluted and rendered by calling:

```
R_DRW2D_CtxConvolutionKernel(g_drw2d_dev, &loc_kernel_config);
R_DRW2D_DrawRectConvolve(g_drw2d_dev, &rect, 0, 0)
```

3.2.7.4 Convolution with color bleeding

As in the example above, a kernel has to be set up first:

```
static const r_drw2d_FixedP_t loc_kernel_blur_coeff[7*7];
static const r_drw2d_ConvKernel_t loc_kernel_config =
{
    loc_kernel_blur_coeff,
    R_DRW2D_CONVKERNEL_COLOR_CHANNEL_RGBA,
    7,
    7,
    R_DRW2D_2X(0.0)
};
```

The kernel's coefficients are computed using the Drw2D utility function R_DRW2D_GetGaussKernel(). The function fills the coefficient array with the values for a 7*7, σ=5 gauss kernel by calling:

```
R_DRW2D_GetGaussKernel(g_drw2d_dev, loc_kernel_blur_coeff, 7, 7, R_DRW2D_2X(5.0));
```

Afterwards the kernel has to be set by a call to R_DRW2D_CtxConvolutionKernel().
To get correct rendering results, the blend mode has to be set to a pre-multiplied alpha mode. The alpha should be already multiplied in src_color during texture creation. This is done by the appropriate calls to R_DRW2D_CtxBlendMode() and R_DRW2D_CtxBlendFactors() in the code snippet below:

```
R_DRW2D_CtxConvolutionKernel(g_drw2d_dev, &loc_kernel_config);
R_DRW2D_CtxBlendMode(g_drw2d_dev, R_DRW2D_BLENDMODE_CUSTOM);
R_DRW2D_CtxBlendFactors(g_drw2d_dev,
                        R_DRW2D_BLEND_ONE, R_DRW2D_BLEND_ONE_MINUS_SRC_ALPHA,
                        R_DRW2D_BLEND_SRC_ALPHA, R_DRW2D_BLEND_ONE_MINUS_SRC_ALPHA);
R_DRW2D_DrawRectConvolve(g_drw2d_dev,
                         &rect,
                         0,
                         0);
```

The final call of R_DRW2D_DrawRectConvolve() convolutes and draws the texture to the current frame buffer.

3.2.8 Special features

3.2.8.1 Bezier curves

R_DRW2D_DrawBezierCurves(*Device*, *Points*, *Count*) draws a Bezier curve consisting of one or more quadratic Bezier segments. All curve *Points* will be transformed by the current transformation matrix. The rendering result will be influenced by the *Width* and *IsClosed* properties of the current line style settings in the context (see R_DRW2D_CtxLineStyle in reference manual). If IsClosed is set, then the last segment will be connected with the first one through a straight line.

3.2.8.2 Command lists

It is possible to create/execute/store hand-made command-lists with the R_DRW2D_GpuCmdList*() interface.
For each command-list the following additional memory requirements have to be taken into account:

1) Cpu Heap: 4 bytes (Drw2D) + 144 bytes (D/AVE HD) + size of your command-list (D/AVE HD)
2) Video Heap: 4 bytes + size of your command-list

3.2.8.3 Custom Blending

WM driver controls the Video Output, in order to stack several layers with alpha information and blend them automatically during layer compositing. In some cases, it may be necessary to mimic this behavior by the GPU. In order to achieve the same blending result with the Drw2D, the following configuration is to be used for blending textures into the current framebuffer,

```
R_DRW2D_CtxBlendMode(g_drw2d_dev, R_DRW2D_BLENDMODE_CUSTOM);
R_DRW2D_CtxBlendFactors(g_drw2d_dev,
                        R_DRW2D_BLEND_SRC_ALPHA, R_DRW2D_BLEND_ONE_MINUS_SRC_ALPHA,
                        R_DRW2D_BLEND_ONE, R_DRW2D_BLEND_ONE_MINUS_SRC_ALPHA);
```

3.3 Device difference

The following table shows the function differences depending on the device.

Table 3-5 APIs supported by Drw2D driver

Function	RH850/D1x Device Name	
	D1L2(H)	D1M1(H) / D1M1-V2 / D1M1A / D1M2(H)
Math utility functions R_DRW2D_FixMul R_DRW2D_FixDiv R_DRW2D_FixAbs R_DRW2D_FixSin R_DRW2D_FixCos R_DRW2D_FixTan R_DRW2D_FixSqrt	OK	OK
Basic functions R_DRW2D_Init R_DRW2D_Open R_DRW2D_Exit R_DRW2D_Close R_DRW2D_VersionString	OK	OK
Context management functions R_DRW2D_ContextInit R_DRW2D_ContextSelect	OK	OK
Context control functions R_DRW2D_CtxBgColor R_DRW2D_CtxClipRect R_DRW2D_CtxBlendMode R_DRW2D_CtxTransformMode	OK	OK
Effect functions R_DRW2D_CtxEffectsSet R_DRW2D_CtxEffectsUpdate R_DRW2D_CtxEffectsDelete	OK	OK
Texture functions R_DRW2D_TextureSet R_DRW2D_TextureBlit	OK	OK
Matrix transformation functions R_DRW2D_CtxIdentity R_DRW2D_CtxTransform R_DRW2D_CtxRotate R_DRW2D_CtxScale R_DRW2D_CtxTranslate	OK (*1)	OK
Framebuffer functions R_DRW2D_FramebufferSet R_DRW2D_FramebufferClear	OK	OK
Display list control functions R_DRW2D_GpuFinish R_DRW2D_GpuFinished	OK	OK
Error handling functions R_DRW2D_ErrCallbackSet R_DRW2D_GlobalErrCallbackSet	OK	OK
Other than above functions	NG	OK

(*1) Transform, Scaling and Translate feature is supported partly.

The following table shows the feature differences depending on the device.

Table 3-6 Features supported by Drw2D driver

Feature	RH850/D1x Device Name	
	D1L2(H)	D1M1(H) / D1M1-V2 / D1M1A / D1M2(H)
Frame buffer format R_DRW2D_PIXELFORMAT_LUM8 R_DRW2D_PIXELFORMAT_RGB565 R_DRW2D_PIXELFORMAT_ARGB1555 R_DRW2D_PIXELFORMAT_RGBA5551 R_DRW2D_PIXELFORMAT_ARGB4444 R_DRW2D_PIXELFORMAT_RGBA4444 R_DRW2D_PIXELFORMAT_ARGB8888 R_DRW2D_PIXELFORMAT_RGBA8888	OK	OK
R_DRW2D_PIXELFORMAT_ALPHA8 R_DRW2D_PIXELFORMAT_AL17 R_DRW2D_PIXELFORMAT_AL44 R_DRW2D_PIXELFORMAT_AL88	NG	OK
Texture format R_DRW2D_PIXELFORMAT_AL17 R_DRW2D_PIXELFORMAT_AL44 R_DRW2D_PIXELFORMAT_AL88 R_DRW2D_PIXELFORMAT_AL1 R_DRW2D_PIXELFORMAT_AL2 R_DRW2D_PIXELFORMAT_AL4 R_DRW2D_PIXELFORMAT_AL8 R_DRW2D_PIXELFORMAT_RGB565 R_DRW2D_PIXELFORMAT_ARGB1555 R_DRW2D_PIXELFORMAT_RGBA5551 R_DRW2D_PIXELFORMAT_ARGB4444 R_DRW2D_PIXELFORMAT_RGBA4444 R_DRW2D_PIXELFORMAT_ARGB8888 R_DRW2D_PIXELFORMAT_RGBA8888	OK	OK
R_DRW2D_PIXELFORMAT_CLUT_8 R_DRW2D_PIXELFORMAT_CLUT_4 R_DRW2D_PIXELFORMAT_CLUT_2 R_DRW2D_PIXELFORMAT_CLUT_1	NG	OK
Texture option R_DRW2D_TEX_NONE R_DRW2D_TEX_RLE R_DRW2D_TEX_SWIZZLE R_DRW2D_TEX_BILINEAR	OK (*1)	OK
R_DRW2D_TEX_WRAPU R_DRW2D_TEX_WRAPV R_DRW2D_TEX_PERSPECTIVE R_DRW2D_TEX_VT	NG	OK
Blend Mode R_DRW2D_BLENDMODE_SRC R_DRW2D_BLENDMODE_SRC_OVER	OK	OK
R_DRW2D_BLENDMODE_CUSTOM R_DRW2D_BLENDMODE_DST_OVER R_DRW2D_BLENDMODE_SRC_IN R_DRW2D_BLENDMODE_DST_IN R_DRW2D_BLENDMODE_MULTIPLY R_DRW2D_BLENDMODE_SCREEN R_DRW2D_BLENDMODE_DARKEN R_DRW2D_BLENDMODE_LIGHTEN R_DRW2D_BLENDMODE_ADDITIVE	NG	OK
Effect R_DRW2D_EFFECT_MODULATE R_DRW2D_EFFECT_CONSTANT_ALPHA R_DRW2D_EFFECT_REPLACE	OK (*2)	OK

R_DRW2D_EFFECT_ADD R_DRW2D_EFFECT_ADD_SIGNED R_DRW2D_EFFECT_SUBTRACT R_DRW2D_EFFECT_INTERPOLATE R_DRW2D_EFFECT_DOT3 R_DRW2D_EFFECT_GRADIENT	NG	OK
Vertex matrix transform mode R_DRW2D_TRANSFORM_NONE R_DRW2D_TRANSFORM_2D	OK	OK
R_DRW2D_TRANSFORM_3D	NG	OK

(*1) Only single setting is supported.
(*2) MODULATE or the combination of CONSTANT_ALPHA and REPLACE is supported.

3.4 Header File List

Table 3-7 Header File List

No.	Header File Name	Description
(1)	r_drw2d_api.h	Header file for Drw2D API.
(2)	r_drw2d_os.h	Header file for OS part for Drw2D API.
(3)	r_config_drw2d.h	Header file for configuration of Drw2D API.
(4)	r_drw2d_ctx_dhd.h	Header file for context of D/AVE HD API. This file is needed in case of Drw2D with D/AVE HD.
(5)	r_drw2d_ctx_cpu.h	Header file for context of CPU drawing API. This file is needed in case of Drw2D with CPU drawing.
(6)	r_typedefs.h	Header file for predefined data types.

4. Functions

4.1 Function List

This section shows the Drw2D API functions in *Table 4-1*. And executable state of each function is shown in the specification of each function.

Table 4-1 List of Drw2D API Functions

Function Name	Purpose
R_DRW2D_FixMul	Multiply two fixed point values.
R_DRW2D_FixDiv	Divide fixed point value "A" by value "B".
R_DRW2D_FixAbs	Returns the absolute value of parameter "A".
R_DRW2D_FixSin	Calculate sine function for Angle.
R_DRW2D_FixCos	Calculate cosine function for Angle.
R_DRW2D_FixTan	Calculate tangent function for Angle.
R_DRW2D_FixSqrt	Calculate square root of fixed point value.
R_DRW2D_Init	Initialize Drw2D API and initialize global Drw2D resources.
R_DRW2D_Open	Initialize Drw2D unit and driver-dependent graphics engine and return device handle.
R_DRW2D_Exit	Shutdown Drw2D API and de-initialize global Drw2D resources.
R_DRW2D_Close	Shutdown Drw2D unit, de-initialize driver-dependent graphics engine and cleans up its internally used resources.
R_DRW2D_VersionString	Returns the version string of the Drw2D API.
R_DRW2D_NativeDriverHandleGet	Returns a handle to the low-level driver instance.
R_DRW2D_NativeDriverBegin	Notify Drw2D that the application wants to access the low level, hardware-specific driver directly.
R_DRW2D_NativeDriverEnd	Notify Drw2D that the application has finished accessing the low level, hardware-specific driver directly.
R_DRW2D_ContextInit	Initialize a render context with default settings.
R_DRW2D_ContextSelect	Sets the given context as the current one.
R_DRW2D_CtxFgColor	Set the foreground color to be used for drawing primitives.
R_DRW2D_CtxBgColor	Set the background color to be used for drawing primitives.
R_DRW2D_CtxClipRect	Sets a global clipping rectangle for subsequent drawing operations.
R_DRW2D_CtxFillMode	Set the filling mode for drawing with primitives.
R_DRW2D_CtxCullMode	Set the culling mode for drawing with primitives.
R_DRW2D_CtxLineStyle	Set the line drawing style (e.g. line caps, line width, ...).
R_DRW2D_CtxBlendMode	Set preset color/alpha source/destination blending equations.
R_DRW2D_CtxBlendFactors	Set color/alpha source/destination blending factors.
R_DRW2D_CtxImgQuality	Sets for the current context a global quality value used for graphics primitives.
R_DRW2D_CtxTransformMode	Set the vertex transform/projection mode.
R_DRW2D_CtxTextureTransformMode	Set the texture coordinate transformation mode.
R_DRW2D_CtxViewport	Set viewport for drawing operation.
R_DRW2D_CtxStripingEnable	Enable striped pixel enumeration (for performance reasons).
R_DRW2D_CtxStripingDisable	Disable striped pixel enumeration.
R_DRW2D_CtxEffectsSet	Sets an array of effects to be used for colorization and blending.
R_DRW2D_CtxEffectsUpdate	updates effect at stage.
R_DRW2D_CtxEffectsDelete	Deletes all effects.
R_DRW2D_CtxTextureSet	Set source texture.
R_DRW2D_TextureBlit	Blit texture from Src to Dest.

Function Name	Purpose
R_DRW2D_CtxTextureColorKeyEnable	Enables Color Keying for the provided RGB color.
R_DRW2D_CtxTextureColorKeyDisable	Disabled a previously set color keying in the GPU driver.
R_DRW2D_CtxIdentity	Reset vertex transformation matrix.
R_DRW2D_CtxTextureIdentity	Reset texture matrix.
R_DRW2D_CtxTransform	Set 4x4 vertex transformation matrix.
R_DRW2D_CtxTextureTransform	Set 3x2 texture transformation matrix.
R_DRW2D_CtxRotate	Multiply current vertex matrix by rotation matrix.
R_DRW2D_CtxRotate3d	Multiply current vertex matrix by 3d rotation matrix.
R_DRW2D_CtxTextureRotate	Multiply current texture matrix by rotation matrix.
R_DRW2D_CtxScale	Multiply current vertex matrix by scaling matrix.
R_DRW2D_CtxTextureScale	Multiply current texture matrix by scaling matrix.
R_DRW2D_CtxTranslate	Multiply current vertex matrix by translation matrix.
R_DRW2D_CtxTextureTranslate	Multiply current texture matrix by translation matrix.
R_DRW2D_CtxFrustum	Multiply current vertex matrix by perspective matrix.
R_DRW2D_VtxTransform	Transform a list of vertices by the current vertex transformation matrix.
R_DRW2D_CtxMatrix	Get the 4x4 vertex transformation matrix and the 3x2 texture transformation matrix.
R_DRW2D_ClutAlloc	Allocates space for a CLUT used by R_DRW2D_ClutSet and R_DRW2D_CtxClutSet
R_DRW2D_ClutFree	Frees CLUT memory previously allocated with R_DRW2D_ClutAlloc.
R_DRW2D_CtxClutSet	Assign a previously created CLUT with the Offset ClutBase (as returned by R_DRW2D_ClutAlloc) to the texture.
R_DRW2D_ClutSet	Create and set a CLUT that was previously allocated with R_DRW2D_ClutAlloc.
R_DRW2D_FramebufferSet	Set current destination framebuffer.
R_DRW2D_FramebufferClear	Clears the current clip rectangle with the current background color (can be set with R_DRW2D_CtxBgColor).
R_DRW2D_DrawTriangles	Render an array of triangles.
R_DRW2D_DrawTrianglesUV	Render an array of UV texture mapped triangles.
R_DRW2D_DrawRect	Render a rectangle.
R_DRW2D_DrawRectUV	Render a UV texture mapped rectangle.
R_DRW2D_DrawQuads	Renders an array of quadrilaterals.
R_DRW2D_DrawQuadsUV	Renders an array of UV texture mapped quadrilaterals.
R_DRW2D_DrawQuads3dUV	Renders an array of UV texture mapped 3D-quadrilaterals.
R_DRW2D_DrawEllipse	Render an ellipse at Point with the specified x and y radius.
R_DRW2D_DrawLines	Render an array of lines.
R_DRW2D_DrawPolyline	Render a polyline consisting of one or many line segments.
R_DRW2D_DrawBezierCurves	Render a bezier curve consisting of one or more quadratic bezier segments.
R_DRW2D_DrawRectConvolve1dx	Apply one dimensional convolution filter to texture and store result in framebuffer.
R_DRW2D_DrawRectConvolve1dy	Apply one dimensional convolution filter to texture and store result in framebuffer.
R_DRW2D_DrawRectConvolve2d	Apply two dimensional convolution filter to texture and store result in framebuffer.
R_DRW2D_DrawRectConvolve	Apply two dimensional convolution filter to texture and store result in framebuffer.
R_DRW2D_CtxConvolutionKernelPreset1d	Select 1d convolution kernel size and weights.
R_DRW2D_CtxConvolutionKernelPreset2d	Select 2d convolution kernel size and weights.

Function Name	Purpose
R_DRW2D_GetGaussKernel	Computes a gauss kernel with the given size and sigma.
R_DRW2D_CtxConvolutionKernel	Select 2d convolution kernel size and weights.
R_DRW2D_CtxConvolutionMode	Sets the convolution mode.
R_DRW2D_GpuFinish	Tell the driver to explicitly trigger the finishing of the current drawing scene operation (display list execution).
R_DRW2D_GpuFinished	Queries the driver for a yes/no whether there are still pending jobs in its pipeline.
R_DRW2D_GpuCmdListCreate	Allocate empty command list.
R_DRW2D_GpuCmdListGenerate	Record command list by calling an application provided function that invokes render commands.
R_DRW2D_GpuCmdListExec	Execute previously recorded command list.
R_DRW2D_GpuCmdListCopy	Copy command list data to memory area.
R_DRW2D_GpuCmdListDelete	Delete command list.
R_DRW2D_PerfCountersAlloc	Allocate hardware performance counter resources for this device context.
R_DRW2D_PerfCountersFree	Free hardware performance counter resources for this device context.
R_DRW2D_PerfValueGet	Query the driver for HW cycles specified by Type and return the value in RetValue.
R_DRW2D_PerfValueReset	Reset the HW cycles of the given performance type to 0.
R_DRW2D_ErrCallbackSet	Install a device context / thread specific application error handler for the driver.
R_DRW2D_GlobalErrCallbackSet	Install a global error handler for the driver.

4.2 Drw2D API Function

This section shows the specification of each function in the Drw2D API functions.

4.2.1 Math utility functions

4.2.1.1 R_DRW2D_FixMul

Function Prototypes

```
r_drw2d_FixedP_t R_DRW2D_FixMul(r_drw2d_FixedP_t   A,
                                r_drw2d_FixedP_t   B)
```

Parameter

Table 4-2 Parameter of R_DRW2D_FixMul

Parameter	Description
A	Fixed point number of value "A".
B	Fixed point number of value "B".

Return Codes

Multiplication result of fixed point by value "A" and value "B".

Description

Multiply two fixed point values.

See also

r_drw2d_FixedP_t

4.2.1.2 R_DRW2D_FixDiv

Function Prototypes

```
r_drw2d_FixedP_t R_DRW2D_FixDiv(r_drw2d_FixedP_t   A,
                                r_drw2d_FixedP_t   B)
```

Parameter

Table 4-3 Parameter of R_DRW2D_FixDiv

Parameter	Description
A	Fixed point number of value "A".
B	Fixed point number of value "B".

Return Codes

Division result of fixed point value "A" by value "B".

Description

Divide fixed point value "A" by value "B".

See also

r_drw2d_FixedP_t

4.2.1.3 R_DRW2D_FixAbs

Function Prototypes

```
r_drw2d_FixedP_t R_DRW2D_FixAbs(r_drw2d_FixedP_t   A)
```

Parameter

Table 4-4 Parameter of R_DRW2D_FixAbs

Parameter	Description
A	Fixed point number of value "A".

Return Codes

Absolute value of value "A".

Description

Returns the absolute value of parameter A.
Not defined for A = 0x80000000 = R_DRW2D_2X(-32768).

See also

r_drw2d_FixedP_t

4.2.1.4 R_DRW2D_FixSin

Function Prototypes

```
r_drw2d_FixedP_t R_DRW2D_FixSin(r_drw2d_FixedP_t   Angle)
```

Parameter

Table 4-5 Parameter of R_DRW2D_FixSin

Parameter	Description
Angle	Fixed point number of Angle. It is a value when 90-degree is 1.0. Example: Angle = R_DRW2D_2X(0.0) : 0-degree Angle = R_DRW2D_2X(0.2) : 18-degree Angle = R_DRW2D_2X(0.5) : 45-degree Angle = R_DRW2D_2X(1.0) : 90-degree Angle = R_DRW2D_2X(2.0) : 180-degree Angle = R_DRW2D_2X(4.0) : 360-degree

Return Codes

Result of sin (Angle)
If you want to convert fixed point value to INT type, FLOAT type and handle it, we recommend using R_DRW2D_2I, R_DRW2D_2F macros.
(see R_DRW2D_2X, R_DRW2D_2I, R_DRW2D_2F)

Description

Calculate sine function for Angle.
For the setting range of parameter, see *Table 6-2*.

See also

r_drw2d_FixedP_t ,R_DRW2D_2X, R_DRW2D_2I, R_DRW2D_2F

4.2.1.5 R_DRW2D_FixCos

Function Prototypes

```
r_drw2d_FixedP_t R_DRW2D_FixCos(r_drw2d_FixedP_t   Angle)
```

Parameter

Table 4-6 Parameter of R_DRW2D_FixCos

Parameter	Description
Angle	Fixed point number of Angle. It is a value when 90-degree is 1.0. Example: Angle = R_DRW2D_2X(0.0) : 0-degree Angle = R_DRW2D_2X(0.2) : 18-degree Angle = R_DRW2D_2X(0.5) : 45-degree Angle = R_DRW2D_2X(1.0) : 90-degree Angle = R_DRW2D_2X(2.0) : 180-degree Angle = R_DRW2D_2X(4.0) : 360-degree

Return Codes

Result of cos (Angle).
If you want to convert fixed point value to INT type, FLOAT type and handle it, we recommend using R_DRW2D_2I, R_DRW2D_2F macros.
(see R_DRW2D_2X, R_DRW2D_2I, R_DRW2D_2F)

Description

Calculate cosine function for Angle.
For the setting range of parameter, see *Table 6-2*.

See also

r_drw2d_FixedP_t, R_DRW2D_2X, R_DRW2D_2I, R_DRW2D_2F

4.2.1.6 R_DRW2D_FixTan

Function Prototypes

```
r_drw2d_FixedP_t R_DRW2D_FixTan(r_drw2d_FixedP_t   Angle)
```

Parameter

Table 4-7 Parameter of R_DRW2D_FixTan

Parameter	Description
Angle	Fixed point number of Angle. It is a value when 90-degree is 1.0. Example: Angle = R_DRW2D_2X(0.0) : 0-degree Angle = R_DRW2D_2X(0.2) : 18-degree Angle = R_DRW2D_2X(0.5) : 45-degree Angle = R_DRW2D_2X(1.0) : 90-degree Angle = R_DRW2D_2X(2.0) : 180-degree Angle = R_DRW2D_2X(4.0) : 360-degree

Return Codes

Result of tan (Angle).
If you want to convert fixed point value to INT type, FLOAT type and handle it, we recommend using R_DRW2D_2I, R_DRW2D_2F macros.
(see R_DRW2D_2X, R_DRW2D_2I, R_DRW2D_2F)

Description

Calculate tangent function for Angle.
For the setting range of parameter, see *Table 6-2*.

See also

r_drw2d_FixedP_t, R_DRW2D_2X, R_DRW2D_2I, R_DRW2D_2F

4.2.1.7 R_DRW2D_FixSqrt

Function Prototypes

```
r_drw2d_FixedP_t R_DRW2D_FixSqrt(r_drw2d_FixedP_t  Value)
```

Parameter

Table 4-8 Parameter of R_DRW2D_FixSqrt

Parameter	Description
Value	Fixed point value.

Return Codes

Result of sqrt (Value).

Description

Calculate square root of fixed point value.

See also

r_drw2d_FixedP_t

4.2.2 Basic functions

4.2.2.1 R_DRW2D_Init

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_Init(void)
```

Parameter

None

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK	- No error occurred.
R_DRW2D_ERR_SYS_MUTEX_CREATE	- Failed to create mutex.

Description

Initialize Drw2D API and initialize global Drw2D resources.
Not thread safe application must ensure that only one thread calls this function.
The underlying drawing engine should be initialized prior to calling this function. In case of a D/AVE HD, this means calling the D/AVE HD kernel mode API initialization functions and enable the GPU2D interrupts.

See also

R_DRW2D_Exit, r_drw2d_Error_t

4.2.2.2 R_DRW2D_Open

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_Open(r_drw2d_Unit_t      Unit,
                             int32_t             DriverUnit,
                             void               *DeviceInternal,
                             r_drw2d_Device_t   *RetDevice)
```

Parameter

Table 4-9 Parameter of R_DRW2D_Open

Parameter	Description
Unit	Unit number (see r_drw2d_Unit_t). Set "0" to Unit.
DriverUnit	Driver unit number. Set "0" to DriverUnit
DeviceInternal	Pointer to the gfx driver handle. Set "r_drw2d_DeviceDHD_t" defined in "r_drw2d_ctx_dhd.h" in case of Drw2D with D/AVE HD. Set "r_drw2d_DeviceCPU_t" defined in "r_drw2d_ctx_cpu.h" in case of Drw2D with CPU drawing. Normally, user never uses this variable. But it must be kept until R_DRW2D_Close is called.
RetDevice	Returns the (opaque) Drw2d device handle.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK	- No error occurred.
R_DRW2D_ERR_UNIT_OUTOFBOUNDS	- Invalid unit number.
R_DRW2D_ERR_DEVICE_INIT	- Failed to initialize device context.
R_DRW2D_ERR_DEVICE_HWINSTANCENR	- Invalid instance (hw unit) nr.
R_DRW2D_ERR_INVALID_VALUE_NULLPTR	- Parameter pointer argument is NULL.
R_DRW2D_ERR_SYS_MUTEX_LOCK	- Failed to lock mutex.
R_DRW2D_ERR_SYS_MUTEX_UNLOCK	- Failed to unlock mutex.
R_DRW2D_ERR_SYS_MUTEX_CREATE	- Failed to create mutex.

Description

Initialize Drw2D unit and driver-dependent graphics engine and return device handle.
This function initializes the driver-dependent graphics engine to its default configuration. It initializes the internal device structure and returns an opaque handle to that structure.
A default render context is created implicitly.
R_DRW2D_Open must be called before any drawing function can take place.
The application must ensure that the RetDevice is not used in more than one thread at a time. Ensure that you provide a real structure pointer as DeviceInternal, not a void pointer! Also, ensure that the DeviceInternal structure members are initialized with 0.
For the setting range of parameter, see *Table 6-2*.

See also

R_DRW2D_Close, r_drw2d_Error_t, r_drw2d_Unit_t, int32_t, r_drw2d_Device_t

4.2.2.3 R_DRW2D_Exit

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_Exit(void)
```

Parameter

None

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK	- No error occurred.
R_DRW2D_ERR_SYS_MUTEX_DESTROY	- Failed to destroy mutex.

Description

Shutdown Drw2D API and de-initialize global Drw2D resources.
Must not be called when R_DRW2D_Init has failed.
Not thread safe application must ensure that all Drw2D units have been closed and only one thread calls this function.
The underlying drawing engine should be de-initialized after calling this function. In case of a D/AVE HD, this means calling the D/AVE HD kernel mode API shutdown functions and disables the GPU2D interrupts.

See also

R_DRW2D_Init, r_drw2d_Error_t

4.2.2.4 R_DRW2D_Close

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_Close(r_drw2d_Device_t  Device)
```

Parameter

Table 4-10 Parameter of R_DRW2D_Close

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_DEVICE_INTERNAL_FINISH - Internal device driver error (during finish).
R_DRW2D_ERR_DEVICE_INTERNAL_SHUTDOWN - Internal device driver error (during shutdown).
R_DRW2D_ERR_SYS_MUTEX_LOCK - Failed to lock mutex.
R_DRW2D_ERR_SYS_MUTEX_UNLOCK - Failed to unlock mutex.

Description

Shutdown Drw2D unit, de-initialize driver-dependent graphics engine and cleans up its internally used resources.

See also

R_DRW2D_Open, r_drw2d_Error_t, r_drw2d_Device_t

4.2.2.5 R_DRW2D_VersionString

Function Prototypes

```
const char_t *R_DRW2D_VersionString(void)
```

Parameter

None

Return Codes

Version string

Description

Returns the version string of the Drw2D API.

See also

char_t

4.2.3 Native driver interface

The following functions can be used to bypass the Drw2D API and access the low level, hardware-specific driver (i.e. D/AVE HD driver) directly.

4.2.3.1 R_DRW2D_NativeDriverHandleGet

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_NativeDriverHandleGet(
                                    r_drw2d_Device_t         Device,
                                    void                   **RetNativeDrvHandle)
```

Parameter

Table 4-11 Parameter of R_DRW2D_NativeDriverHandleGet

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
RetNativeDrvHandle	Returns Native driver handle.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK	- No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE	- Invalid device handle.
R_DRW2D_ERR_INVALID_VALUE_NULLPTR	- Parameter pointer argument is NULL.
R_DRW2D_ERR_DEVICE_NATIVEDRVHANDLE	- Failed to query native driver handle.

Description

Returns a handle to the D/AVE HD driver instance.
Drw2D must have been initialized using R_DRW2D_Init prior to calling this function.

See also

R_DRW2D_NativeDriverBegin, R_DRW2D_NativeDriverEnd, r_drw2d_Error_t, r_drw2d_NativeDrvFlags_t, r_drw2d_Device_t

4.2.3.2 R_DRW2D_NativeDriverBegin

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_NativeDriverBegin(r_drw2d_Device_t   Device,
                                          uint32_t           Flags)
```

Parameter

Table 4-12 Parameter of R_DRW2D_NativeDriverBegin

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Flags	The upper 16bits of this parameter are reserved for driver-specific extensions. (See r_drw2d_NativeDrvFlags_t)

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_DEVICE_SAVESTATE - Failed to backup low level driver state.
R_DRW2D_ERR_DEVICE_SAVESTATEALLOC - Failed to create save state.
R_DRW2D_ERR_DEVICE_INTERNAL_ALLOCCLUT- Internal device driver error (while handling CLUT memory).
R_DRW2D_ERR_FRAMEBUFFER_HANDLE - Invalid framebuffer handle.
R_DRW2D_ERR_TEXTURE_ADDR - Invalid texture buffer address.
R_DRW2D_ERR_TEXTURE_PIXELFMT - Unsupported texel format.
R_DRW2D_ERR_TEXTURE_RLE_BPP - Bits per texel not suitable for RLE decoder (D/AVE HD specific).
R_DRW2D_ERR_EFFECT_INVALID_TEXTURE - Invalid Texture Index.
R_DRW2D_ERR_BUFFER_PIXELFMT - Invalid/unsupported pixel format.
R_DRW2D_ERR_BUFFER_ALIGNMENT - Buffer alignment not correct (for framebuffer).
R_DRW2D_ERR_INVALID_VALUE_FILLMODE - Invalid fill mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDMODE - Invalid blend mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCRGB_UNSUPPORTED - Unsupported SrcRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTRGB_UNSUPPORTED - Unsupported DstRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCALPHA_UNSUPPORTED - Unsupported SrcAlpha blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTALPHA_UNSUPPORTED - Unsupported DstAlpha blend factor.
R_DRW2D_ERR_EFFECT_INVALID_OPERAND - Invalid Parameters provided for effects.
R_DRW2D_ERR_EFFECT_INVALID_OPERATION - Invalid effect name.
R_DRW2D_ERR_EFFECT_OUT_OF_RESOURCES - Combination of effects cannot be realized.

Description

Notify Drw2D that the application wants to access the low level, hardware-specific driver directly.

See also

R_DRW2D_NativeDriverHandleGet, R_DRW2D_NativeDriverEnd, r_drw2d_Error_t, r_drw2d_Device_t, uint32_t, r_drw2d_NativeDrvFlags_t

4.2.3.3 R_DRW2D_NativeDriverEnd

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_NativeDriverEnd(r_drw2d_Device_t  Device,
                                        uint32_t          Flags)
```

Parameter

Table 4-13 Parameter of R_DRW2D_NativeDriverEnd

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Flags	Reserved for future extensions. Pass 0 for now. (See r_drw2d_NativeDrvFlags_t)

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_DEVICE_SAVESTATEALLOC - Failed to create save state.
R_DRW2D_ERR_DEVICE_RESTORESTATE - Failed to restore low level driver state.

Description

Notify Drw2D that the application has finished accessing the low level, hardware-specific driver directly.
For the setting range of parameter, see ***Table 6-2***.

See also

R_DRW2D_NativeDriverHandleGet, R_DRW2D_NativeDriverBegin, r_drw2d_Error_t, r_drw2d_Device_t, uint32_t, r_drw2d_NativeDrvFlags_t

4.2.4 Context management functions

The application can initialize and select render contexts with this API.
A context stores the following settings:

- Clipping rectangle (see R_DRW2D_CtxClipRect)
- View port (see R_DRW2D_CtxViewport)
- Foreground color and alpha (see R_DRW2D_CtxFgColor)
- Background color and alpha (see R_DRW2D_CtxBgColor)
- Cull mode (see R_DRW2D_CtxCullMode)
- Striping (see R_DRW2D_CtxStripingEnable, R_DRW2D_CtxStripingDisable)
- 1D convolution filter kernel presets (see R_DRW2D_CtxConvolutionKernelPreset1d)
- 2D convolution filter kernel presets (see R_DRW2D_CtxConvolutionKernelPreset2d)
- Convolution filter kernel (see R_DRW2D_CtxConvolutionKernel)
- Effect stages (see R_DRW2D_CtxEffectsSet, R_DRW2D_CtxEffectsDelete , R_DRW2D_CtxEffectsUpdate)
- Number of effect stages (see R_DRW2D_CtxEffectsSet, R_DRW2D_CtxEffectsDelete , R_DRW2D_CtxEffectsUpdate)
- Fill mode (see R_DRW2D_CtxFillMode, r_drw2d_FillMode_t)
- Blend mode (see R_DRW2D_CtxBlendMode, r_drw2d_BlendMode_t)
- Blend factors (see R_DRW2D_CtxBlendFactors, r_drw2d_BlendFactor_t)
- Transformation matrix (see R_DRW2D_CtxTransform, R_DRW2D_CtxTranslate, R_DRW2D_CtxRotate, R_DRW2D_CtxRotate3d, R_DRW2D_CtxScale)
- Texture matrix (see R_DRW2D_CtxTextureTransform, R_DRW2D_CtxTextureTranslate, R_DRW2D_CtxTextureRotate, R_DRW2D_CtxTextureScale)
- Line style (see R_DRW2D_CtxLineStyle, r_drw2d_LineCap_t, r_drw2d_LineJoin_t, r_drw2d_LineStyle_t)
- Image quality / antialiasing mode (see R_DRW2D_CtxImgQuality, r_drw2d_ImgQuality_t)
- Transform mode (see R_DRW2D_CtxTransformMode, r_drw2d_TransformMode_t)
- Source texture (see R_DRW2D_CtxTextureSet, r_drw2d_Texture_t)

The context state can be changed by calling one of the R_DRW2D_Ctx*() functions.
The Drw2D API provides one default context by default, which is initialized in R_DRW2D_Open.
Custom render contexts can be setup using R_DRW2D_ContextInit and selected with R_DRW2D_ContextSelect.

4.2.4.1 R_DRW2D_ContextInit

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_ContextInit(
                                r_drw2d_Device_t                 Device,
                                struct r_drw2d_RenderContext_s  *RenderContext,
                                r_drw2d_Context_t               *RetContext)
```

Parameter

Table 4-14 Parameter of R_DRW2D_ContextInit

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
RenderContext	Reference to uninitialized render context structure (see r_drw2d_RenderContext_s). The instance must be kept while this context is selecting.
RetContext	The context pointer that is initialized.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_INVALID_VALUE_NULLPTR - Parameter pointer argument is NULL.

Description

Initialize a render context with default settings.

See also

R_DRW2D_ContextSelect, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_RenderContext_s, r_drw2d_Context_t

4.2.4.2 R_DRW2D_ContextSelect

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_ContextSelect(r_drw2d_Device_t    Device,
                                      r_drw2d_Context_t   Context)
```

Parameter

Table 4-15 Parameter of R_DRW2D_ContextSelect

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Context	Render context handle (see r_drw2d_Context_t).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_CONTEXT_NOTINUSE - Context not in use. (while calling R_DRW2D_ContextSelect R_DRW2D_ContextInit has to be called first.)

Description

Sets the given context as the current one.
Passing Context handle NULL will select the default context.

See also

R_DRW2D_ContextInit, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_Context_t

4.2.5 Context control functions

All of the functions in this section operate globally on the currently selected context (state machine-like in OpenGL/OpenVG). That means they influence primitive drawing.
Resetting values or setting back to default is the responsibility of the application/framework above Drw2D.

4.2.5.1 R_DRW2D_CtxFgColor

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxFgColor(r_drw2d_Device_t   Device,
                                   r_drw2d_Color_t    Color)
```

Parameter

Table 4-16 Parameter of R_DRW2D_CtxFgColor

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Color	The color value (32bit packed ARGB). (see r_drw2d_Color_t)

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.
R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_EFFECT_INVALID_OPERATION - Invalid effect name.

Description

Set the foreground color to be used for drawing primitives with solid mode.

Between calls to R_DRW2D_CtxEffectsSet and R_DRW2D_CtxEffectsDelete, this function is invalid and an error will be returned.

See also

R_DRW2D_CtxBgColor, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_Color_t

4.2.5.2 R_DRW2D_CtxBgColor

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxBgColor(r_drw2d_Device_t   Device,
                                   r_drw2d_Color_t    Color)
```

Parameter

Table 4-17 Parameter of R_DRW2D_CtxBgColor

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Color	The color value (32bit packed ARGB). (see r_drw2d_Color_t)

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_EFFECT_INVALID_OPERATION - Invalid effect name.

Description

Set the background color to be used for drawing primitives.
The R_DRW2D_FramebufferClear function always uses the current background color.

Between calls to R_DRW2D_CtxEffectsSet and R_DRW2D_CtxEffectsDelete, this function is invalid and an error will be returned.

See also

R_DRW2D_CtxFgColor, r_drw2d_Error_t, r_drw2d_Color_t

4.2.5.3 R_DRW2D_CtxClipRect

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxClipRect(r_drw2d_Device_t          Device,
                                    const r_drw2d_IntRect_t  *Rect)
```

Parameter

Table 4-18 Parameter of R_DRW2D_CtxClipRect

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Rect	Rectangular clipping area (window coordinates) (see r_drw2d_IntRect_t). All elements must be positive numbers (or 0).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK	- No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE	- Invalid device handle.
R_DRW2D_ERR_INVALID_VALUE_NULLPTR	- Parameter pointer argument is NULL.

Description

Sets a global clipping rectangle for subsequent drawing operations. A clip-rect can be set by calling this function before each R_DRW2D_Draw* API and R_DRW2D_FramebufferClear.

For the setting range of parameter, see ***Table 6-2***.

See also

r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_IntRect_t

4.2.5.4 R_DRW2D_CtxFillMode

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxFillMode(r_drw2d_Device_t      Device,
                                    r_drw2d_FillMode_t    Mode)
```

Parameter

Table 4-19 Parameter of R_DRW2D_CtxFillMode

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Mode	The fill mode to be used. (see r_drw2d_FillMode_t)

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK	- No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE	- Invalid device handle.
R_DRW2D_ERR_INVALID_VALUE_FILLMODE	- Invalid fill mode.
R_DRW2D_ERR_EFFECT_INVALID_OPERATION	- Invalid effect name.

Description

Set the filling mode for drawing with primitives.
This is used for placing e.g. a texture on top of any primitive.

Between calls to R_DRW2D_CtxEffectsSet and R_DRW2D_CtxEffectsDelete, this function is invalid and an error will be returned.

See also

r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_FillMode_t

4.2.5.5 R_DRW2D_CtxCullMode

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxCullMode(r_drw2d_Device_t      Device,
                                    r_drw2d_CullMode_t    CullMode)
```

Parameter

Table 4-20 Parameter of R_DRW2D_CtxCullMode

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
CullMode	The cull mode to be used. (see r_drw2d_CullMode_t)

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_INVALID_VALUE_CULLMODE - Invalid cull mode.

Description

Set the culling mode for drawing with primitives.
This is used to discard triangles or rectangles depending on their winding order.
The default cull mode is R_DRW2D_CULLMODE_NONE.

See also

r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_CullMode_t

4.2.5.6 R_DRW2D_CtxLineStyle

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxLineStyle(r_drw2d_Device_t            Device,
                                     const r_drw2d_LineStyle_t  *Style)
```

Parameter

Table 4-21 Parameter of R_DRW2D_CtxLineStyle

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Style	Pointer to line style struct (see r_drw2d_LineStyle_t).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_INVALID_VALUE_LINEJOIN - Invalid LineJoin type.
R_DRW2D_ERR_INVALID_VALUE_LINECAP - Invalid LineCap type.
R_DRW2D_ERR_INVALID_VALUE_LINEWIDTH - Invalid line width.
R_DRW2D_ERR_INVALID_VALUE_MITERLIMIT - Invalid miter limit.
R_DRW2D_ERR_INVALID_VALUE_NULLPTR - Parameter pointer argument is NULL.

Description

Set the line drawing style (e.g. line caps, line width, ...).

See also

R_DRW2D_DrawLines, R_DRW2D_DrawPolyline, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_LineStyle_t

4.2.5.7 R_DRW2D_CtxBlendMode

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxBlendMode(r_drw2d_Device_t     Device,
                                     r_drw2d_BlendMode_t  BlendMode)
```

Parameter

Table 4-22 Parameter of R_DRW2D_CtxBlendMode

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
BlendMode	The blend mode to be used. (see r_drw2d_BlendMode_t)

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_INVALID_VALUE_BLENDMODE - Invalid blend mode.

Description

Set preset color/alpha source/destination blending equations.
When R_DRW2D_BLENDMODE_CUSTOM is selected, the color/alpha blending equations are determined by the blend factors set by R_DRW2D_CtxBlendFactors.

See also

R_DRW2D_CtxBlendFactors, r_drw2d_Error_t, r_drw2d_BlendMode_t,

4.2.5.8 R_DRW2D_CtxBlendFactors

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxBlendFactors(r_drw2d_Device_t          Device,
                                        r_drw2d_BlendFactor_t     SrcRGB,
                                        r_drw2d_BlendFactor_t     DstRGB,
                                        r_drw2d_BlendFactor_t     SrcAlpha,
                                        r_drw2d_BlendFactor_t     DstAlpha)
```

Parameter

Table 4-23 Parameter of R_DRW2D_CtxBlendFactors

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
SrcRGB	The blend factor to be used for source RGB values (see r_drw2d_BlendFactor_t).
DstRGB	The blend factor to be used for destination RGB values (see r_drw2d_BlendFactor_t).
SrcAlpha	The blend factor to be used for source alpha values (see r_drw2d_BlendFactor_t).
DstAlpha	The blend factor to be used for destination alpha values (see r_drw2d_BlendFactor_t).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCRGB - Invalid SrcRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTRGB - Invalid DstRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCALPHA - Invalid SrcAlpha blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTALPHA - Invalid DstAlpha blend factor.

Description

Set color/alpha source/destination blending factors.
In order for these to have an effect, the R_DRW2D_BLENDMODE_CUSTOM blend mode must be selected (see R_DRW2D_CtxBlendMode).
The effective color/alpha blend equation is determined by

dst_color = src_color*src_factor_color + dst_color*dst_factor_color
dst_alpha = src_alpha*src_factor_alpha + dst_alpha*dst_factor_alpha

See also

R_DRW2D_CtxBlendMode, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_BlendFactor_t

4.2.5.9 R_DRW2D_CtxImgQuality

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxImgQuality(r_drw2d_Device_t       Device,
                                      r_drw2d_ImgQuality_t   Quality)
```

Parameter

Table 4-24 Parameter of R_DRW2D_CtxImgQuality

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Quality	Image quality/antialiasing mode (see r_drw2d_ImgQuality_t).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK	- No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE	- Invalid device handle.
R_DRW2D_ERR_INVALID_VALUE_IMGQUALITY	- Invalid image quality mode.

Description

Sets for the current context a global quality value used for graphics primitives

If *Quality* is R_DRW2D_IMGQUALITY_LOW, antialiasing and some effect features are not enabled. Refer to R_DRW2D_CtxEffectsSet for details of effect features. This mode can be used as a performance optimization.
Note that the DHD platform is limited to an edge width/height of max. 2048 when using non antialiased (R_DRW2D_IMGQUALITY_LOW) edges.

No restrictions on R_DRW2D_IMGQUALITY_MIDDLE and R_DRW2D_IMGQUALITY_HIGH.

See also

r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_ImgQuality_t
R_DRW2D_CtxEffectsSet

4.2.5.10 R_DRW2D_CtxTransformMode

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxTransformMode(r_drw2d_Device_t          Device,
                                         r_drw2d_TransformMode_t  Mode)
```

Parameter

Table 4-25 Parameter of R_DRW2D_CtxTransformMode

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Mode	The transform mode to be used (see r_drw2d_TransformMode_t).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_INVALID_VALUE_TRANSFORMMODE - Invalid vertex matrix transform mode.

Description

Set the vertex transform/projection mode.
The default is R_DRW2D_TRANSFORM_2D (2D vertex matrix transformation).
When set to R_DRW2D_TRANSFORM_NONE, vertices will be used as-is, i.e. they will not be transformed by the vertex matrix. This mode can be used as a performance optimization. When no explicit UV coordinates are given, the texture is mapped to the framebuffer 1:1.
When set to R_DRW2D_TRANSFORM_3D, vertices will be transformed by the 4x4 vertex matrix and projected onto the current viewport.

See also

r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_TransformMode_t

4.2.5.11 R_DRW2D_CtxTextureTransformMode

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxTextureTransformMode(
                                    r_drw2d_Device_t                Device,
                                    r_drw2d_TextureTransformMode_t  Mode)
```

Parameter

Table 4-26 Parameter of R_DRW2D_CtxTextureTransformMode

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Mode	The transform mode to be used (see r_drw2d_TextureTransformMode_t).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK	- No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE	- Invalid device handle.
R_DRW2D_ERR_TEXTURE_TRANSFORMMODE	- Invalid texture matrix transform mode.

Description

Set the texture coordinate transformation mode.
The default is R_DRW2D_TEX_TRANSFORM_2D (2D texture matrix transformation).
When set to R_DRW2D_TEX_TRANSFORM_NONE, texture coordinates will be used as-is, i.e. they will not be transformed by the texture matrix. This mode can be used as a performance optimization. When no explicit UV coordinates are given, the texture is mapped to the framebuffer 1:1.
When set to R_DRW2D_TEX_TRANSFORM_2D, texture coordinates will be transformed by the 3x2 texture matrix.

See also

r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_TextureTransformMode_t

4.2.5.12 R_DRW2D_CtxViewport

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxViewport(r_drw2d_Device_t          Device,
                                    const r_drw2d_IntRect_t  *Rect)
```

Parameter

Table 4-27 Parameter of R_DRW2D_CtxViewport

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Rect	The viewport rectangle (see r_drw2d_IntRect_t).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_INVALID_VALUE_NULLPTR - Parameter pointer argument is NULL.
R_DRW2D_ERR_INVALID_VALUE_VIEWPORT_X - Invalid viewport Pos.X.
R_DRW2D_ERR_INVALID_VALUE_VIEWPORT_Y - Invalid viewport Pos.Y.
R_DRW2D_ERR_INVALID_VALUE_VIEWPORT_W - Invalid viewport Size.Width.
R_DRW2D_ERR_INVALID_VALUE_VIEWPORT_H - Invalid viewport Size.Height.

Description

Set viewport for drawing operation.
Viewport is a feature to restrict destination of drawing operations.
The default is to use the current framebuffer width/height, offset by (0; 0).
After setting a custom viewport, set the Rect.Size to (0; 0) to revert to the default behavior.
R_DRW2D_FramebufferClear is also affected by Viewport settings.
For the setting range of parameter, see ***Table 6-2***.

See also

r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_IntRect_t

4.2.5.13 R_DRW2D_CtxStripingEnable

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxStripingEnable(r_drw2d_Device_t   Device)
```

Parameter

Table 4-28 Parameter of R_DRW2D_CtxStripingEnable

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.
R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.

Description

Enable striped pixel enumeration (for performance reasons).
This should only be enabled if texture mapping is used and larger triangles/quads are being rendered.
The actual stripe settings are determined per-primitive.

See also

R_DRW2D_CtxStripingDisable, r_drw2d_Error_t, r_drw2d_Device_t

4.2.5.14 R_DRW2D_CtxStripingDisable

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxStripingDisable(r_drw2d_Device_t  Device)
```

Parameter

Table 4-29 Parameter of R_DRW2D_CtxStripingDisable

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.
R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.

Description

Disable striped pixel enumeration.

See also

R_DRW2D_CtxStripingEnable, r_drw2d_Error_t, r_drw2d_Device_t

4.2.6 Effect functions

For more information, help and examples concerning the Drw2D Effects API, see *Chapter 3.2.5*.

4.2.6.1 R_DRW2D_CtxEffectsSet

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxEffectsSet(r_drw2d_Device_t       Device,
                                      r_drw2d_EffectStage_t  *Effects,
                                      uint32_t               Count)
```

Parameter

Table 4-30 Parameter of R_DRW2D_CtxEffectsSet

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Effects	Effect stage contains information about one effect. Array with effects (see r_drw2d_EffectStage_t).
Count	Number of effects in the array.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK	- No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE	- Invalid device handle.
R_DRW2D_ERR_INVALID_VALUE	- Parameter/argument value is out of range or undefined.

Description

Sets an array of effects to be used for colorization and blending. The data in the array must be readable/writeable until a call to R_DRW2D_CtxEffectsDelete.
The content of the array will be modified by R_DRW2D_CtxEffectsUpdate.
Between calls to R_DRW2D_CtxEffectsSet and R_DRW2D_CtxEffectsDelete any calls to functions R_DRW2D_CtxFgColor, R_DRW2D_CtxBgColor, and R_DRW2D_CtxFillMode are invalid and an error will be returned.
The effects R_DRW2D_EFFECT_REPLACE, R_DRW2D_EFFECT_MODULATE, R_DRW2D_EFFECT_ADD, R_DRW2D_EFFECT_SUBTRACT, R_DRW2D_EFFECT_ADD_SIGNED, R_DRW2D_EFFECT_INTERPOLATE, and R_DRW2D_EFFECT_DOT3 can be combined to calculate a final color. If the input source is specified as R_DRW2D_EFFECT_SOURCE_PREV_STAGE, the result of the previous stage will be used.
If the combination of effects cannot be realized, an error will be returned by the drawing call.
The effects R_DRW2D_EFFECT_CONSTANT_ALPHA and R_DRW2D_EFFECT_GRADIENT provide a final alpha blending. It is necessary to set the image quality to R_DRW2D_IMGQUALITY_MEDIUM or R_DRW2D_IMGQUALITY_HIGH (see R_DRW2D_CtxImgQuality) for these effects to work: In the case of R_DRW2D_EFFECT_CONSTANT_ALPHA the result will be blended with a constant alpha. The R_DRW2D_EFFECT_GRADIENT can be used to specify two points and two alpha values to calculate a linear gradient, which will be used to blend the final color. The effects R_DRW2D_EFFECT_GRADIENT and R_DRW2D_EFFECT_CONSTANT_ALPHA can be combined.
See *Chapter 3.2.5* for more information on how to use effects.
For the setting range of parameter, see *Table 6-2*.

See also

R_DRW2D_CtxEffectsUpdate, R_DRW2D_CtxEffectsDelete, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_EffectStage_t, uint32_t

4.2.6.2 R_DRW2D_CtxEffectsUpdate

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxEffectsUpdate(r_drw2d_Device_t              Device,
                                         r_drw2d_EffectName_t         Name,
                                         uint32_t                     Stage,
                                         uint32_t                     NumParams,
                                         const r_drw2d_EffectParam_t  *Params)
```

Parameter

Table 4-31 Parameter of R_DRW2D_CtxEffectsUpdate

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Name	Effect (see r_drw2d_EffectName_t).
Stage	Stage of effect to be updated.
NumParams	Number of parameters to be passed.
Params	Array of parameters (see r_drw2d_EffectParam_t).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_INVALID_VALUE - Parameter/argument value is out of range or undefined.
R_DRW2D_ERR_INVALID_VALUE_NULLPTR - Parameter pointer argument is NULL.
R_DRW2D_ERR_EFFECT_INVALID_OPERATION - Invalid Effect Name.

Description

updates effect at stage.
For the setting range of parameter, see *Table 6-2*.

See also

R_DRW2D_CtxEffectsSet, R_DRW2D_CtxEffectsDelete, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_EffectName_t, uint32_t, r_drw2d_EffectParam_t

4.2.6.3 R_DRW2D_CtxEffectsDelete

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxEffectsDelete(r_drw2d_Device_t Device)
```

Parameter

Table 4-32 Parameter of R_DRW2D_CtxEffectsDelete

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.
R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.

Description

Deletes all effects.

See also

R_DRW2D_CtxEffectsSet, R_DRW2D_CtxEffectsUpdate, r_drw2d_Error_t, r_drw2d_Device_t

4.2.7 Texture functions

4.2.7.1 R_DRW2D_CtxTextureSet

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxTextureSet(r_drw2d_Device_t          Device,
                                      uint32_t                 TextureUnit,
                                      const r_drw2d_Texture_t  *Texture)
```

Parameter

Table 4-33 Parameter of R_DRW2D_CtxTextureSet

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
TextureUnit	Texture Unit Number.
Texture	Reference to texture structure (see r_drw2d_Texture_t). NULL to deselect current texture.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK	- No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE	- Invalid device handle.
R_DRW2D_ERR_TEXTURE_UNIT	- Invalid texture unit number.
R_DRW2D_ERR_TEXTURE_ADDR	- Invalid texture buffer address.
R_DRW2D_ERR_BUFFER_WIDTH	- Invalid/unsupported width.
R_DRW2D_ERR_BUFFER_HEIGHT	- Invalid/unsupported height.
R_DRW2D_ERR_BUFFER_PIXELFMT	- Invalid/unsupported pixel format.

Description

Set source texture.
The Drw2D with D/AVE HD supports the direct / zero-copy use of an application provided texture address.
The texture is used when the fill mode is set to R_DRW2D_FILLMODE_TEXTURE.
For the setting range of parameter, see ***Table 6-2***.

See also

R_DRW2D_CtxFillMode, R_DRW2D_TextureBlit, r_drw2d_Error_t, r_drw2d_Device_t, uint32_t, r_drw2d_Texture_t,

4.2.7.2 R_DRW2D_TextureBlit

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_TextureBlit(r_drw2d_Device_t          Device,
                                    const r_drw2d_Rect_t     *SrcRect,
                                    const r_drw2d_Rect_t     *DstRect)
```

Parameter

Table 4-34 Parameter of R_DRW2D_TextureBlit

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
SrcRect	Source rectangle (see r_drw2d_Rect_t).
DstRect	Destination rectangle (see r_drw2d_Rect_t).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_DEVICE_INTERNAL_ALLOCCLUT- Internal device driver error (while handling CLUT memory).
R_DRW2D_ERR_FRAMEBUFFER_HANDLE - Invalid framebuffer handle.
R_DRW2D_ERR_TEXTURE_ADDR - Invalid texture buffer address.
R_DRW2D_ERR_TEXTURE_PIXELFMT - Unsupported texel format.
R_DRW2D_ERR_TEXTURE_RLE_BPP - Bits per texel not suitable for RLE decoder (D/AVE HD specific).
R_DRW2D_ERR_TEXTURE_TRANSFORMMODE - Invalid texture matrix transform mode (R_DRW2D_CtxTextureTransformMode).
R_DRW2D_ERR_EFFECT_INVALID_TEXTURE - Invalid Texture Index.
R_DRW2D_ERR_BUFFER_PIXELFMT - Invalid/unsupported pixel format.
R_DRW2D_ERR_BUFFER_ALIGNMENT - Buffer alignment not correct (for framebuffer).
R_DRW2D_ERR_INVALID_VALUE_NULLPTR - Parameter pointer argument is NULL.
R_DRW2D_ERR_INVALID_VALUE_FILLMODE - Invalid fill mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDMODE - Invalid blend mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCRGB_UNSUPPORTED - Unsupported SrcRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTRGB_UNSUPPORTED - Unsupported DstRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCALPHA_UNSUPPORTED - Unsupported SrcAlpha blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTALPHA_UNSUPPORTED - Unsupported DstAlpha blend factor.
R_DRW2D_ERR_DRAWING_TEXTUREBLIT - Failed to draw blit texture.
R_DRW2D_ERR_EFFECT_INVALID_OPERAND - Invalid Parameters provided for effects.
R_DRW2D_ERR_EFFECT_INVALID_OPERATION - Invalid effect name.
R_DRW2D_ERR_EFFECT_OUT_OF_RESOURCES - Combination of effects cannot be realized.

Description

Blit texture from Src to Dest.
Src can be NULL, in that case the blit origin is 0, 0 inside of the texture and dimensions are taken from the DstRect.
For the setting range of parameter, see *Table 6-2*.

See also

R_DRW2D_CtxFillMode, R_DRW2D_CtxTextureSet, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_Rect_t

4.2.7.3 R_DRW2D_CtxTextureColorKeyEnable

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxTextureColorKeyEnable(r_drw2d_Device_t   Device,
                                                 r_drw2d_Color_t    ColorKey)
```

Parameter

Table 4-35 Parameter of R_DRW2D_CtxTextureColorKeyEnable

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
ColorKey	The color before replacement in RGB (alpha component is ignored)

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_NATIVEDRVHANDLE - Failed to query native driver handle.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.

Description

Enables Color Keying for the provided RGB color. The color after replacement will be fully transparent.

See also

R_DRW2D_CtxTextureColorKeyDisable, r_drw2d_Error_t, r_drw2d_Device_t

4.2.7.4 R_DRW2D_CtxTextureColorKeyDisable

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxTextureColorKeyDisable(r_drw2d_Device_t  Device)
```

Parameter

Table 4-36 Parameter of R_DRW2D_CtxTextureColorKeyDisable

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK	- No error occurred.
R_DRW2D_ERR_DEVICE_NATIVEDRVHANDLE	- Failed to query native driver handle.
R_DRW2D_ERR_DEVICE_HANDLE	- Invalid device handle.

Description

Disabled a previously set color keying.

See also

R_DRW2D_CtxTextureColorKeyEnable, r_drw2d_Error_t, r_drw2d_Device_t

4.2.8 Matrix transformation functions

4.2.8.1 R_DRW2D_CtxIdentity

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxIdentity(r_drw2d_Device_t  Device)
```

Parameter

Table 4-37 Parameter of R_DRW2D_CtxIdentity

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.
R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.

Description

Reset vertex transformation matrix.
The identity matrix maps the vertex at (0; 0) to the top/left framebuffer position.
The vertex at (framebuffer_width-1, framebuffer_height-1) is mapped to the bottom/right framebuffer position.

See also

R_DRW2D_CtxTransform, R_DRW2D_CtxRotate, R_DRW2D_CtxRotate3d, R_DRW2D_CtxScale, R_DRW2D_CtxTranslate, R_DRW2D_CtxTextureIdentity, R_DRW2D_CtxTextureTransform, R_DRW2D_CtxTextureRotate, R_DRW2D_CtxTextureScale, R_DRW2D_CtxTextureTranslate, r_drw2d_Error_t, r_drw2d_Device_t

4.2.8.2 R_DRW2D_CtxTextureIdentity

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxTextureIdentity(r_drw2d_Device_t  Device)
```

Parameter

Table 4-38 Parameter of R_DRW2D_CtxTextureIdentity

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.
R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.

Description

Reset texture matrix.
If both texture and vertex matrices are set to identity, vertices and texture coordinates will use the same coordinate system.
i.e. drawing a textured rectangle at (10;10) with size (40;30) will result in a 1:1 mapping of the respective texture area.

See also

R_DRW2D_CtxTextureTransform, R_DRW2D_CtxTextureRotate, R_DRW2D_CtxTextureScale, R_DRW2D_CtxTextureTranslate, R_DRW2D_CtxIdentity, R_DRW2D_CtxTransform, R_DRW2D_CtxRotate, R_DRW2D_CtxRotate3d, R_DRW2D_CtxScale, R_DRW2D_CtxTranslate, r_drw2d_Error_t, r_drw2d_Device_t

4.2.8.3 R_DRW2D_CtxTransform

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxTransform(r_drw2d_Device_t         Device,
                                     const r_drw2d_FixedP_t  *Matrix)
```

Parameter

Table 4-39 Parameter of R_DRW2D_CtxTransform

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Matrix	Reference to 4x4 transformation matrix.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_INVALID_VALUE_NULLPTR - Parameter pointer argument is NULL.

Description

Set 4x4 vertex transformation matrix.
Can be used for 3D perspective mapping or affine transformations (scale, translate, rotate).

The matrix is expected to be in column-major format and use the following element order:

```
[ 0 4  8 12 ]
[ 1 5  9 13 ]
[ 2 6 10 14 ]
[ 3 7 11 15 ]
```

(the translation vector is stored in elements 12, 13, and 14)

See also

R_DRW2D_CtxIdentity, R_DRW2D_CtxRotate, R_DRW2D_CtxRotate3d, R_DRW2D_CtxScale, R_DRW2D_CtxTranslate, R_DRW2D_CtxTextureIdentity, R_DRW2D_CtxTextureTransform, R_DRW2D_CtxTextureRotate, R_DRW2D_CtxTextureScale, R_DRW2D_CtxTextureTranslate, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_FixedP_t

4.2.8.4 R_DRW2D_CtxTextureTransform

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxTextureTransform(r_drw2d_Device_t        Device,
                                            const r_drw2d_FixedP_t  *Matrix)
```

Parameter

Table 4-40 Parameter of R_DRW2D_CtxTextureTransform

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Matrix	Reference to 3x2 transformation matrix.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_INVALID_VALUE_NULLPTR - Parameter pointer argument is NULL.

Description

Set 3x2 texture transformation matrix.
Can be used for e.g. affine transformations (scale, translate, rotate).

The matrix is expected to be in column-major format and use the following element order:

```
[ 0 2 4 ]
[ 1 3 5 ]
```

(the translation vector is stored in elements 4 and 5)

See also

R_DRW2D_CtxTextureIdentity, R_DRW2D_CtxTextureRotate, R_DRW2D_CtxTextureScale, R_DRW2D_CtxTextureTranslate, R_DRW2D_CtxIdentity, R_DRW2D_CtxTransform, R_DRW2D_CtxRotate, R_DRW2D_CtxRotate3d, R_DRW2D_CtxScale, R_DRW2D_CtxTranslate, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_FixedP_t

4.2.8.5 R_DRW2D_CtxRotate

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxRotate(r_drw2d_Device_t Device,
                                  r_drw2d_FixedP_t Angle)
```

Parameter

Table 4-41 Parameter of R_DRW2D_CtxRotate

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Angle	Angle of rotation about z axis. Angle is in degrees measure. When converting to fixed point value, we recommend using R_DRW2D_2X macros.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.
R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.

Description

Multiply current vertex matrix by rotation matrix.

```
[ cos(Angle)  , -sin(Angle) , 0,  0 ]
[ sin(Angle)  , cos(Angle)  , 0,  0 ]
[ 0           , 0           , 1,  0 ]
[ 0           , 0           , 0,  1 ]
```

For the setting range of parameter, see ***Table 6-2***.

See also

R_DRW2D_CtxIdentity, R_DRW2D_CtxTransform, R_DRW2D_CtxScale, R_DRW2D_CtxTranslate, R_DRW2D_CtxTextureIdentity, R_DRW2D_CtxTextureTransform, R_DRW2D_CtxTextureRotate, R_DRW2D_CtxTextureScale, R_DRW2D_CtxTextureTranslate, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_FixedP_t, R_DRW2D_2X

4.2.8.6 R_DRW2D_CtxRotate3d

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxRotate3d(r_drw2d_Device_t  Device,
                                    r_drw2d_FixedP_t  X,
                                    r_drw2d_FixedP_t  Y,
                                    r_drw2d_FixedP_t  Z,
                                    r_drw2d_FixedP_t  Angle)
```

Parameter

Table 4-42 Parameter of R_DRW2D_CtxRotate3d

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
X	x-coordinate of unit vector starting in origin (0,0,0).
Y	y-coordinate of unit vector starting in origin (0,0,0).
Z	z-coordinate of unit vector starting in origin (0,0,0).
Angle	Angle of rotation around axis of unit vector defined by (X,Y,Z). Angle is in degrees measure. When converting to fixed point value, we recommend using R_DRW2D_2X macros.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.
R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.

Description

Multiply current vertex matrix by 3d rotation matrix.

```
[ X² (1-cos(Angle))+ cos(Angle),   XY(1- cos(Angle))-Zsin(Angle),  XZ(1- cos(Angle))+Ysin(Angle),  0 ]
[ YX(1-cos(Angle))+Zsin(Angle),  Y² (1- cos(Angle))+ cos(Angle),  YZ(1- cos(Angle))-Xsin(Angle) ,  0 ]
[ XZ(1-cos(Angle))-Ysin(Angle) ,  YZ(1- cos(Angle))+Xsin(Angle), Z²(1- cos(Angle))+ cos(Angle)  ,  0 ]
[ 0                                , 0                                , 0                                , 1 ]
```

The argument (X, Y, Z) should be unit vector. $\sqrt{X^2 + Y^2 + Z^2} = 1$
For the setting range of parameter, see ***Table 6-2***.

See also

R_DRW2D_CtxIdentity, R_DRW2D_CtxTransform, R_DRW2D_CtxScale, R_DRW2D_CtxTranslate, R_DRW2D_CtxTextureIdentity, R_DRW2D_CtxTextureTransform, R_DRW2D_CtxTextureRotate, R_DRW2D_CtxTextureScale, R_DRW2D_CtxTextureTranslate, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_FixedP_t, R_DRW2D_2X

4.2.8.7 R_DRW2D_CtxTextureRotate

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxTextureRotate(r_drw2d_Device_t    Device,
                                         r_drw2d_FixedP_t    Angle)
```

Parameter

Table 4-43 Parameter of R_DRW2D_CtxTextureRotate

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Angle	Angle of rotation about z axis. Angle is in degrees measure. When converting to fixed point value, we recommend using R_DRW2D_2X macros.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.
R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.

Description

Multiply current texture matrix by rotation matrix.

```
[ cos(Angle), -sin(Angle)      , 0 ]
[ sin(Angle) ,  cos(Angle)     , 0 ]
```

For the setting range of parameter, see ***Table 6-2***.

See also

R_DRW2D_CtxTextureIdentity, R_DRW2D_CtxTextureTransform, R_DRW2D_CtxTextureScale, R_DRW2D_CtxTextureTranslate, R_DRW2D_CtxIdentity, R_DRW2D_CtxTransform, R_DRW2D_CtxRotate, R_DRW2D_CtxRotate3d, R_DRW2D_CtxScale, R_DRW2D_CtxTranslate, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_FixedP_t, R_DRW2D_2X

4.2.8.8 R_DRW2D_CtxScale

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxScale(r_drw2d_Device_t     Device,
                                 r_drw2d_FixedP_t     ScaleX,
                                 r_drw2d_FixedP_t     ScaleY,
                                 r_drw2d_FixedP_t     ScaleZ)
```

Parameter

Table 4-44 Parameter of R_DRW2D_CtxScale

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
ScaleX	Scale factor for the X axis (1.0 means no scaling).
ScaleY	Scale factor for the Y axis (1.0 means no scaling).
ScaleZ	Scale factor for the Z axis (1.0 means no scaling).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.
R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.

Description

Multiply current vertex matrix by scaling matrix.

```
[ ScaleX  ,  0      , 0      , 0 ]
[ 0       , ScaleY  , 0      , 0 ]
[ 0       , 0       , ScaleZ , 0 ]
[ 0       , 0       , 0      , 1 ]
```

See also

R_DRW2D_CtxIdentity, R_DRW2D_CtxTransform, R_DRW2D_CtxRotate, R_DRW2D_CtxRotate3d, R_DRW2D_CtxTranslate, R_DRW2D_CtxTextureIdentity, R_DRW2D_CtxTextureTransform, R_DRW2D_CtxTextureRotate, R_DRW2D_CtxTextureScale, R_DRW2D_CtxTextureTranslate, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_FixedP_t

4.2.8.9 R_DRW2D_CtxTextureScale

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxTextureScale(r_drw2d_Device_t  Device,
                                        r_drw2d_FixedP_t  ScaleX,
                                        r_drw2d_FixedP_t  ScaleY)
```

Parameter

Table 4-45 Parameter of R_DRW2D_CtxTextureScale

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
ScaleX	Scale factor for the X axis (1.0 means no scaling).
ScaleY	Scale factor for the Y axis (1.0 means no scaling).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.
R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.

Description

Multiply current texture matrix by scaling matrix.

```
[ ScaleX , 0 , 0 ]
[0 , ScaleY , 0 ]
```

See also

R_DRW2D_CtxTextureIdentity, R_DRW2D_CtxTextureTransform, R_DRW2D_CtxTextureRotate, R_DRW2D_CtxTextureTranslate, R_DRW2D_CtxIdentity, R_DRW2D_CtxTransform, R_DRW2D_CtxRotate, R_DRW2D_CtxRotate3d, R_DRW2D_CtxScale, R_DRW2D_CtxTranslate, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_FixedP_t

4.2.8.10 R_DRW2D_CtxTranslate

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxTranslate(r_drw2d_Device_t     Device,
                                     r_drw2d_FixedP_t     TransX,
                                     r_drw2d_FixedP_t     TransY,
                                     r_drw2d_FixedP_t     TransZ)
```

Parameter

Table 4-46 Parameter of R_DRW2D_CtxTranslate

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
TransX	X axis translation.
TransY	Y axis translation.
TransZ	Z axis translation.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.
R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.

Description

Multiply current vertex matrix by translation matrix.

```
[ 1, 0, 0, TransX]
[ 0, 1, 0, TransY]
[ 0, 0, 1, TransZ]
[ 0, 0, 0, 1      ]
```

The result of the matrix depends on the execution order of other functions.
e.g.

Function call order	Vertex matrix result

`R_DRW2D_CtxIdentify(g_drw2d_dev);`
`R_DRW2D_CtxTranslate(g_drw2d_dev, 100, 50, 0);`
`R_DRW2D_CtxScale(g_drw2d_dev, 2, 2, 1);`
	[2, 0, 0, 100]
	[0, 2, 0, 50]
	[0, 0, 1, 0]
	[0, 0, 0, 1]

`R_DRW2D_CtxIdentify(g_drw2d_dev);`
`R_DRW2D_CtxScale(g_drw2d_dev, 2, 2, 1);`
`R_DRW2D_CtxTranslate(g_drw2d_dev, 100, 50, 0);`
	[2, 0, 0, 200]
	[0, 2, 0, 100]
	[0, 0, 1, 0]
	[0, 0, 0, 1]

For the setting range of parameter, see ***Table 6-2***.

See also

R_DRW2D_CtxIdentity, R_DRW2D_CtxTransform, R_DRW2D_CtxRotate, R_DRW2D_CtxRotate3d, R_DRW2D_CtxScale, R_DRW2D_CtxTextureIdentity, R_DRW2D_CtxTextureTransform, R_DRW2D_CtxTextureRotate, R_DRW2D_CtxTextureScale, R_DRW2D_CtxTextureTranslate, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_FixedP_t

4.2.8.11 R_DRW2D_CtxTextureTranslate

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxTextureTranslate(r_drw2d_Device_t Device,
                                            r_drw2d_FixedP_t TransX,
                                            r_drw2d_FixedP_t TransY)
```

Parameter

Table 4-47 Parameter of R_DRW2D_CtxTextureTranslate

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
TransX	X axis translation.
TransY	Y axis translation.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.
R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.

Description

Multiply current texture matrix by translation matrix.

```
[ 1, 0, TransX ]
[ 0, 1, TransY ]
```

For the setting range of parameter, see ***Table 6-2***.

See also

R_DRW2D_CtxTextureIdentity, R_DRW2D_CtxTextureTransform, R_DRW2D_CtxTextureRotate, R_DRW2D_CtxTextureScale, R_DRW2D_CtxIdentity, R_DRW2D_CtxTransform, R_DRW2D_CtxRotate, R_DRW2D_CtxRotate3d, R_DRW2D_CtxScale, R_DRW2D_CtxTranslate, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_FixedP_t

4.2.8.12 R_DRW2D_CtxFrustum

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxFrustum(r_drw2d_Device_t      Device,
                                   r_drw2d_FixedP_t      Left,
                                   r_drw2d_FixedP_t      Right,
                                   r_drw2d_FixedP_t      Bottom,
                                   r_drw2d_FixedP_t      Top,
                                   r_drw2d_FixedP_t      ZNear,
                                   r_drw2d_FixedP_t      ZFar)
```

Parameter

Table 4-48 Parameter of R_DRW2D_CtxFrustum

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Left	Left vertical clipping plane.
Right	Right vertical clipping plane.
Bottom	Bottom horizontal clipping plane.
Top	Top horizontal clipping plane.
ZNear	Distance to near clipping plane.
ZFar	Distance to far clipping plane.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_INVALID_VALUE - Parameter/argument value is out of range or undefined.

Description

Multiply current vertex matrix by perspective matrix.

```
[ (2*ZNear) / (Right-Left) , 0                        , (Right+Left) / (Right-Left)       , 0                                                ]
[ 0                        , (2*ZNear) / (Top-Bottom), (Top+bottom) / (Top-Bottom)     , 0                                                ]
[ 0                        , 0                        , -((ZFar+ZNear) / (ZFar-ZNear)), - ((2*ZFar*ZNear) / (ZFar-ZNear) ) ]
[ 0                        , 0                        , -1                              , 0                                                ]
```

For the setting range of parameter, see ***Table 6-2***.

See also

r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_FixedP_t

4.2.8.13 R_DRW2D_VtxTransform

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_VtxTransform(r_drw2d_Device_t      Device,
                                     r_drw2d_Vec4_t       *Vertices,
                                     uint32_t              NumVertices)
```

Parameter

Table 4-49 Parameter of R_DRW2D_VtxTransform

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Vertices	Pointer to vertices (see r_drw2d_Vec4_t).
NumVertices	Number of vertices provided by "Vertices" parameter.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_INVALID_VALUE_NULLPTR - Parameter pointer argument is NULL.

Description

Transform a list of vertices by the current vertex transformation matrix.
This function applies the current transformation matrix to an arbitrary number of points supplied via the parameters Vertices and NumVertices.
The calculation result is written back to instance of Vertices.
This function can be used to get the scope of the next drawing operation before executing it. This information can be used to allocate buffers with as small as possible size, as the target scope of the drawing operation is already known.
For the setting range of parameter, see ***Table 6-2***.

See also

r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_Vec4_t, uint32_t

4.2.8.14 R_DRW2D_CtxMatrix

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxMatrix(r_drw2d_Device_t          Device,
                                  r_drw2d_FixedP_t *const   VertexMatrix,
                                  r_drw2d_FixedP_t *const   TextureMatrix)
```

Parameter

Table 4-50 Parameter of R_DRW2D_CtxMatrix

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
VertexMatrix	Reference to 4x4 transformation matrix.
TextureMatrix	Reference to 3x2 transformation matrix.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.
R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.

Description

Get the 4x4 vertex transformation matrix and the 3x2 texture transformation matrix.
Can be used to save and restore (R_DRW2D_CtxTransform, R_DRW2D_CtxTextureTransform) the current transformation matrices. This may reduce the CPU overhead of repeatedly calling almost identical transformation operations for several similar drawing operations. If just one of the matrices is required, set the other pointer to zero.
VertexMatrix Reference to 4x4 transformation matrix. The matrix is expected to be in column-major format and use the following element order:

```
[ 0,  4,   8, 12 ]
[ 1,  5,   9, 13 ]
[ 2,  6, 10, 14 ]
[ 3,  7, 11, 15 ]
```

(the translation vector is stored in elements 12, 13, and 14)

TextureMatrix - Reference to 3x2 transformation matrix. The matrix is expected to be in column-major format and use the following element order:

```
[ 0, 2, 4 ]
[ 1, 3, 5 ]
```

(the translation vector is stored in elements 4 and 5)

See also

R_DRW2D_CtxIdentity, R_DRW2D_CtxRotate, R_DRW2D_CtxRotate3d, R_DRW2D_CtxScale, R_DRW2D_CtxTranslate, R_DRW2D_CtxTextureIdentity, R_DRW2D_CtxTextureTransform, R_DRW2D_CtxTextureRotate, R_DRW2D_CtxTextureScale, R_DRW2D_CtxTextureTranslate, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_FixedP_t

4.2.8.15 R_DRW2D_ClutAlloc

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_ClutAlloc (r_drw2d_Device_t    Device,
                                   uint32_t            Size,
                                   uint32_t           *ClutBase)
```

Parameter

Table 4-51 Parameter of R_DRW2D_ClutAlloc

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Size	Size of CLUT elements. Specify the Size according to the pixel format to use. When using R_DRW2D_PIXELFORMAT_CLUT_1, specify "2" for Size. When using R_DRW2D_PIXELFORMAT_CLUT_2, specify "4" for Size. When using R_DRW2D_PIXELFORMAT_CLUT_4, specify "16" for Size. When using R_DRW2D_PIXELFORMAT_CLUT_8, specify "256" for Size.
ClutBase	This function will write the offset address of CLUT memory to store specified size. Clutbase is used as a handle value for subsequent CLUT APIs.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK	- No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE	- Invalid device handle.
R_DRW2D_ERR_DEVICE_INTERNAL_ALLOCCLUT	- Internal device driver error (while handling CLUT memory).

Description

Allocates space for a CLUT used by R_DRW2D_ClutSet and R_DRW2D_CtxClutSet.
Call this function first when using CLUT.

CLUT memory can store 512 elements in total.
Control it so that it does not become fragmented.

See also

R_DRW2D_ClutFree, R_DRW2D_CtxClutSet, R_DRW2D_ClutSet, r_drw2d_Error_t, r_drw2d_Device_t, uint32_t

4.2.8.16 R_DRW2D_ClutFree

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_ClutFree (r_drw2d_Device_t    Device,
                                  uint32_t            Size,
                                  uint32_t            ClutBase)
```

Parameter

Table 4-52 Parameter of R_DRW2D_ClutFree

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Size	Size of CLUT elements. Specify the Size according to the pixel format to use. When using R_DRW2D_PIXELFORMAT_CLUT_1, specify "2" for Size. When using R_DRW2D_PIXELFORMAT_CLUT_2, specify "4" for Size. When using R_DRW2D_PIXELFORMAT_CLUT_4, specify "16" for Size. When using R_DRW2D_PIXELFORMAT_CLUT_8, specify "256" for Size.
ClutBase	Offset address of CLUT memory.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK	- No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE	- Invalid device handle.
R_DRW2D_ERR_DEVICE_INTERNAL_ALLOCCLUT	- Internal device driver error (while handling CLUT memory).

Description

Frees CLUT memory previously allocated with R_DRW2D_ClutAlloc.

See also

R_DRW2D_ClutAlloc, R_DRW2D_CtxClutSet, R_DRW2D_ClutSet, r_drw2d_Error_t, r_drw2d_Device_t, uint32_t

4.2.8.17 R_DRW2D_CtxClutSet

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxClutSet (r_drw2d_Device_t  Device,
                                    uint32_t          ClutBase)
```

Parameter

Table 4-53 Parameter of R_DRW2D_CtxClutSet

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
ClutBase	Offset address of CLUT memory.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.
R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.

Description

Assign a previously created CLUT with the ClutBase (as returned by R_DRW2D_ClutAlloc) to the texture.
Before calling this function, call R_DRW2D_ClutAlloc and acquire ClutBase.

See also

R_DRW2D_ClutAlloc, R_DRW2D_ClutFree, R_DRW2D_ClutSet, r_drw2d_Error_t, r_drw2d_Device_t, uint32_t

4.2.8.18 R_DRW2D_ClutSet

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_ClutSet (r_drw2d_Device_t   Device,
                                 uint32_t          *Data,
                                 uint32_t           Start,
                                 uint32_t           Size)
```

Parameter

Table 4-54 Parameter of R_DRW2D_ClutSet

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Data	Pointer to the array of CLUT data. The data consists of 32 bit ARGB color data
Start	Offset address of CLUT memory. Specify the ClutBase acquired with R_DRW2D_ClutAlloc.
Size	Size of CLUT elements.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK	- No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE	- Invalid device handle.
R_DRW2D_ERR_DEVICE_INTERNAL_ALLOCCLUT	- Internal device driver error (while handling CLUT memory).
R_DRW2D_ERR_INVALID_VALUE_NULLPTR	- Parameter pointer argument is NULL.

Description

Create and set a CLUT that was previously allocated with R_DRW2D_ClutAlloc.
Before calling this function, call R_DRW2D_ClutAlloc and acquire ClutBase.
The relationship between pixel format and "Data" and "Size" are as follows.
- R_DRW2D_PIXELFORMAT_CLUT_1: "Data" will be 2 colors of data. Specify 2 for "Size".
- R_DRW2D_PIXELFORMAT_CLUT_2: "Data" will be 4 colors of data. Specify 4 for "Size".
- R_DRW2D_PIXELFORMAT_CLUT_4: "Data" will be 16 colors of data. Specify 16 for "Size".
- R_DRW2D_PIXELFORMAT_CLUT_8: "Data" will be 256 colors of data. Specify 256 for "Size".

See also

R_DRW2D_ClutAlloc, R_DRW2D_ClutFree, R_DRW2D_CtxClutSet, r_drw2d_Error_t, r_drw2d_Device_t, uint32_t

4.2.9 Framebuffer functions

4.2.9.1 R_DRW2D_FramebufferSet

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_FramebufferSet(r_drw2d_Device_t          Device,
                                       r_drw2d_Framebuffer_t    *Framebuffer)
```

Parameter

Table 4-55 Parameter of R_DRW2D_FramebufferSet

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Framebuffer	Reference to framebuffer structure (see r_drw2d_Framebuffer_t).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK	- No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE	- Invalid device handle.
R_DRW2D_ERR_FRAMEBUFFER_ADDR	- Invalid framebuffer address.
R_DRW2D_ERR_BUFFER_WIDTH	- Invalid/unsupported width.
R_DRW2D_ERR_BUFFER_HEIGHT	- Invalid/unsupported height.
R_DRW2D_ERR_BUFFER_PIXELFMT	- Invalid/unsupported pixel format.

Description

Set current destination framebuffer.
Allocation of the framebuffer is the responsibility of the application.
For the setting range of parameter, see ***Table 6-2***.

See also

r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_Framebuffer_t

4.2.9.2 R_DRW2D_FramebufferClear

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_FramebufferClear(r_drw2d_Device_t Device)
```

Parameter

Table 4-56 Parameter of R_DRW2D_FramebufferClear

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_DEVICE_INTERNAL_ALLOCCLUT- Internal device driver error (while handling CLUT memory).
R_DRW2D_ERR_FRAMEBUFFER_HANDLE - Invalid framebuffer handle.
R_DRW2D_ERR_TEXTURE_ADDR - Invalid texture buffer address.
R_DRW2D_ERR_TEXTURE_PIXELFMT - Unsupported texel format.
R_DRW2D_ERR_TEXTURE_RLE_BPP - Bits per texel not suitable for RLE decoder (D/AVE HD specific).
R_DRW2D_ERR_EFFECT_INVALID_TEXTURE - Invalid Texture Index.
R_DRW2D_ERR_BUFFER_PIXELFMT - Invalid/unsupported pixel format.
R_DRW2D_ERR_BUFFER_ALIGNMENT - Buffer alignment not correct (for framebuffer).
R_DRW2D_ERR_INVALID_VALUE_FILLMODE - Invalid fill mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDMODE - Invalid blend mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCRGB_UNSUPPORTED - Unsupported SrcRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTRGB_UNSUPPORTED - Unsupported DstRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCALPHA_UNSUPPORTED - Unsupported SrcAlpha blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTALPHA_UNSUPPORTED - Unsupported DstAlpha blend factor.
R_DRW2D_ERR_EFFECT_INVALID_OPERAND - Invalid Parameters provided for effects.
R_DRW2D_ERR_EFFECT_INVALID_OPERATION - Invalid effect name.
R_DRW2D_ERR_EFFECT_OUT_OF_RESOURCES - Combination of effects cannot be realized.

Description

Clears the current clip rectangle with the current background color (can be set with R_DRW2D_CtxBgColor). R_DRW2D_FramebufferClear will always use the SOLID fill mode and ignore current blend mode/factor settings.

See also

R_DRW2D_FramebufferSet, r_drw2d_Error_t, r_drw2d_Device_t

4.2.10 Render functions

All following render functions take the current context settings (fill mode, fg/bg colors, texture, ...) into account.

4.2.10.1 R_DRW2D_DrawTriangles

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_DrawTriangles(r_drw2d_Device_t       Device,
                                      const r_drw2d_Point_t  *Points,
                                      uint32_t               Count,
                                      const uint8_t          *EdgeFlags)
```

Parameter

Table 4-57 Parameter of R_DRW2D_DrawTriangles

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Points	Pointer to array of vertices (see r_drw2d_Point_t). The array should consist of *Count* elements.
Count	Number of vertices. *Count*/3 triangles will be drawn.
EdgeFlags	Pointer to array of edge flag. The array should consist of (*Count*/3) elements. One byte per triangle. Specifies which triangles edge will be antialiased. can be combined with OR setting from followings. R_DRW2D_EDGE_AB R_DRW2D_EDGE_BC R_DRW2D_EDGE_CA See r_drw2d_EdgeFlag_t. If NULL, do not use antialiasing.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_DEVICE_INTERNAL_ALLOCCLUT- Internal device driver error (while handling CLUT memory).
R_DRW2D_ERR_FRAMEBUFFER_HANDLE - Invalid framebuffer handle.
R_DRW2D_ERR_TEXTURE_ADDR - Invalid texture buffer address.
R_DRW2D_ERR_TEXTURE_PIXELFMT - Unsupported texel format.
R_DRW2D_ERR_TEXTURE_RLE_BPP - Bits per texel not suitable for RLE decoder (D/AVE HD specific).
R_DRW2D_ERR_EFFECT_INVALID_TEXTURE - Invalid Texture Index.
R_DRW2D_ERR_BUFFER_PIXELFMT - Invalid/unsupported pixel format.
R_DRW2D_ERR_BUFFER_ALIGNMENT - Buffer alignment not correct (for framebuffer).
R_DRW2D_ERR_INVALID_VALUE_NULLPTR - Parameter pointer argument is NULL.
R_DRW2D_ERR_INVALID_VALUE_FILLMODE - Invalid fill mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDMODE - Invalid blend mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCRGB_UNSUPPORTED - Unsupported SrcRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTRGB_UNSUPPORTED - Unsupported DstRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCALPHA_UNSUPPORTED - Unsupported SrcAlpha blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTALPHA_UNSUPPORTED - Unsupported DstAlpha blend factor.
R_DRW2D_ERR_DRAWING_DRAWTRI - Failed to draw triangle.
R_DRW2D_ERR_EFFECT_INVALID_OPERAND - Invalid Parameters provided for effects.
R_DRW2D_ERR_EFFECT_INVALID_OPERATION - Invalid effect name.
R_DRW2D_ERR_EFFECT_OUT_OF_RESOURCES - Combination of effects cannot be realized.

Description

Render an array of triangles.
Each triangle vertex will be transformed by the current vertex transformation matrix.
If texture mapping is enabled, the texture will be applied according to the current texture transformation matrix.
Note that the D/AVE HD platform is limited to an edge width/height of max. 2048 when using non antialiased edges (Edge flag set to 0).
For the setting range of parameter, see ***Table 6-2***.

See also

R_DRW2D_DrawTrianglesUV, R_DRW2D_DrawPolyline, R_DRW2D_DrawLines, R_DRW2D_DrawEllipse, R_DRW2D_DrawRect, R_DRW2D_FramebufferClear, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_Point_t, uint32_t, uint8_t

4.2.10.2 R_DRW2D_DrawTrianglesUV

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_DrawTrianglesUV(r_drw2d_Device_t         Device,
                                        const r_drw2d_Point_t   *Points,
                                        uint32_t                 Count,
                                        const uint8_t           *EdgeFlags,
                                        const r_drw2d_UVCoord_t *UVCoords)
```

Parameter

Table 4-58 Parameter of R_DRW2D_DrawTrianglesUV

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Points	Pointer to array of vertices (see r_drw2d_Point_t). The array should consist of *Count* elements.
Count	Number of vertices. (*Count*/3) triangles will be drawn.
EdgeFlags	Pointer to array of edge flag. The array should consist of (*Count*/3) elements. One byte per triangle. Specifies which triangles edge will be antialiased. can be combined with OR setting from followings. R_DRW2D_EDGE_AB R_DRW2D_EDGE_BC R_DRW2D_EDGE_CA See r_drw2d_EdgeFlag_t. If NULL, do not use antialiasing.
UVCoords	Pointer to array of UV coordinates (see r_drw2d_UVCoord_t). The array should consist of *Count* elements.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_DEVICE_INTERNAL_ALLOCCLUT- Internal device driver error (while handling CLUT memory).
R_DRW2D_ERR_FRAMEBUFFER_HANDLE - Invalid framebuffer handle.
R_DRW2D_ERR_TEXTURE_ADDR - Invalid texture buffer address.
R_DRW2D_ERR_TEXTURE_PIXELFMT - Unsupported texel format.
R_DRW2D_ERR_TEXTURE_RLE_BPP - Bits per texel not suitable for RLE decoder (D/AVE HD specific).
R_DRW2D_ERR_EFFECT_INVALID_TEXTURE - Invalid Texture Index.
R_DRW2D_ERR_BUFFER_PIXELFMT - Invalid/unsupported pixel format.
R_DRW2D_ERR_BUFFER_ALIGNMENT - Buffer alignment not correct (for framebuffer).
R_DRW2D_ERR_INVALID_VALUE_NULLPTR - Parameter pointer argument is NULL.
R_DRW2D_ERR_INVALID_VALUE_FILLMODE - Invalid fill mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDMODE - Invalid blend mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCRGB_UNSUPPORTED - Unsupported SrcRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTRGB_UNSUPPORTED - Unsupported DstRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCALPHA_UNSUPPORTED - Unsupported SrcAlpha blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTALPHA_UNSUPPORTED - Unsupported DstAlpha blend factor.
R_DRW2D_ERR_DRAWING_DRAWTRIUV - Failed to draw UV texture mapped triangle.
R_DRW2D_ERR_EFFECT_INVALID_OPERAND - Invalid Parameters provided for effects.
R_DRW2D_ERR_EFFECT_INVALID_OPERATION - Invalid effect name.
R_DRW2D_ERR_EFFECT_OUT_OF_RESOURCES - Combination of effects cannot be realized.

Description

Render an array of UV texture mapped triangles.
Each triangle vertex will be transformed by the current vertex transformation matrix.
Each UV coordinate will be transformed by the current texture transformation matrix if the texture transform mode R_DRW2D_CtxTextureTransformMode is set to R_DRW2D_TEX_TRANSFORM_2D.
If the fill mode is set to R_DRW2D_FILLMODE_SOLID, this function behaves like R_DRW2D_DrawTriangles and the UV coordinate array is ignored.
Note that the D/AVE HD platform is limited to an edge width/height of max. 2048 when using non antialiased edges (Edge flag set to 0).
For the setting range of parameter, see ***Table 6-2***.

See also

R_DRW2D_DrawTrianglesUV, R_DRW2D_DrawPolyline, R_DRW2D_DrawLines, R_DRW2D_DrawEllipse, R_DRW2D_DrawRect, R_DRW2D_FramebufferClear, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_EdgeFlag_t, r_drw2d_Point_t, uint32_t, uint8_t, r_drw2d_UVCoord_t

4.2.10.3 R_DRW2D_DrawRect

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_DrawRect(r_drw2d_Device_t          Device,
                                 const r_drw2d_Rect_t     *Rect)
```

Parameter

Table 4-59 Parameter of R_DRW2D_DrawRect

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Rect	Rectangle position and size.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_DEVICE_INTERNAL_ALLOCCLUT- Internal device driver error (while handling CLUT memory).
R_DRW2D_ERR_FRAMEBUFFER_HANDLE - Invalid framebuffer handle.
R_DRW2D_ERR_TEXTURE_ADDR - Invalid texture buffer address.
R_DRW2D_ERR_TEXTURE_PIXELFMT - Unsupported texel format.
R_DRW2D_ERR_TEXTURE_RLE_BPP - Bits per texel not suitable for RLE decoder (D/AVE HD specific).
R_DRW2D_ERR_EFFECT_INVALID_TEXTURE - Invalid Texture Index.
R_DRW2D_ERR_BUFFER_PIXELFMT - Invalid/unsupported pixel format.
R_DRW2D_ERR_BUFFER_ALIGNMENT - Buffer alignment not correct (for framebuffer).
R_DRW2D_ERR_INVALID_VALUE_NULLPTR - Parameter pointer argument is NULL.
R_DRW2D_ERR_INVALID_VALUE_FILLMODE - Invalid fill mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDMODE - Invalid blend mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCRGB_UNSUPPORTED - Unsupported SrcRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTRGB_UNSUPPORTED - Unsupported DstRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCALPHA_UNSUPPORTED - Unsupported SrcAlpha blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTALPHA_UNSUPPORTED - Unsupported DstAlpha blend factor.
R_DRW2D_ERR_DRAWING_DRAWQUAD - Failed to draw quad.
R_DRW2D_ERR_EFFECT_INVALID_OPERAND - Invalid Parameters provided for effects.
R_DRW2D_ERR_EFFECT_INVALID_OPERATION - Invalid effect name.
R_DRW2D_ERR_EFFECT_OUT_OF_RESOURCES - Combination of effects cannot be realized.

Description

Render a rectangle.
The Pos and Size fields of the Rect argument are used to construct a quad which will then be transformed by the current vertex matrix.
If texture mapping is enabled, the texture will be applied according to the current texture transformation matrix.
For the setting range of parameter, see *Table 6-2*.

See also

R_DRW2D_DrawTriangles, R_DRW2D_DrawTrianglesUV, R_DRW2D_DrawPolyline, R_DRW2D_DrawLines, R_DRW2D_DrawEllipse, R_DRW2D_FramebufferClear, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_Rect_t

4.2.10.4 R_DRW2D_DrawRectUV

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_DrawRectUV(r_drw2d_Device_t          Device,
                                   const r_drw2d_Rect_t     *Rect,
                                   const r_drw2d_UVCoord_t  *UVCoords)
```

Parameter

Table 4-60 Parameter of R_DRW2D_DrawRectUV

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Rect	Rectangle position and size.
UVCoords	Reference to an array of 4 UV coordinates (left/top, right/top, right/bottom, left/bottom order) (see r_drw2d_UVCoord_t).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_DEVICE_INTERNAL_ALLOCCLUT- Internal device driver error (while handling CLUT memory).
R_DRW2D_ERR_FRAMEBUFFER_HANDLE - Invalid framebuffer handle.
R_DRW2D_ERR_TEXTURE_ADDR - Invalid texture buffer address.
R_DRW2D_ERR_TEXTURE_PIXELFMT - Unsupported texel format.
R_DRW2D_ERR_TEXTURE_RLE_BPP - Bits per texel not suitable for RLE decoder (D/AVE HD specific).
R_DRW2D_ERR_EFFECT_INVALID_TEXTURE - Invalid Texture Index.
R_DRW2D_ERR_BUFFER_PIXELFMT - Invalid/unsupported pixel format.
R_DRW2D_ERR_BUFFER_ALIGNMENT - Buffer alignment not correct (for framebuffer).
R_DRW2D_ERR_INVALID_VALUE_NULLPTR - Parameter pointer argument is NULL.
R_DRW2D_ERR_INVALID_VALUE_FILLMODE - Invalid fill mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDMODE - Invalid blend mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCRGB_UNSUPPORTED - Unsupported SrcRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTRGB_UNSUPPORTED - Unsupported DstRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCALPHA_UNSUPPORTED - Unsupported SrcAlpha blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTALPHA_UNSUPPORTED - Unsupported DstAlpha blend factor.
R_DRW2D_ERR_DRAWING_DRAWRECTUV - Failed to draw UV texture mapped rectangle.
R_DRW2D_ERR_EFFECT_INVALID_OPERAND - Invalid Parameters provided for effects.
R_DRW2D_ERR_EFFECT_INVALID_OPERATION - Invalid effect name.
R_DRW2D_ERR_EFFECT_OUT_OF_RESOURCES - Combination of effects cannot be realized.

Description

Render a UV texture mapped rectangle.
The Pos and Size fields of the Rect argument are used to construct a quad which will then be transformed by the current vertex matrix.
Each UV coordinate will be transformed by the current texture transformation matrix if the texture transform mode R_DRW2D_CtxTextureTransformMode is set to R_DRW2D_TEX_TRANSFORM_2D.
If the fill mode is set to R_DRW2D_FILLMODE_SOLID, this function behaves like R_DRW2D_DrawRect and the UV coordinate array is ignored.
For the setting range of parameter, see ***Table 6-2***.

See also

R_DRW2D_DrawTriangles, R_DRW2D_DrawTrianglesUV, R_DRW2D_DrawPolyline, R_DRW2D_DrawLines, R_DRW2D_DrawEllipse, R_DRW2D_FramebufferClear r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_Rect_t, r_drw2d_UVCoord_t

4.2.10.5 R_DRW2D_DrawQuads

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_DrawQuads(r_drw2d_Device_t        Device,
                                  const r_drw2d_Point_t  *Points,
                                  uint32_t               Count,
                                  const uint8_t          *EdgeFlags)
```

Parameter

Table 4-61 Parameter of R_DRW2D_DrawQuads

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Points	Pointer to array of vertices (see r_drw2d_Point_t). The array should consist of *Count* elements.
Count	Number of vertices. (*Count*/4) quadrilaterals will be drawn.
EdgeFlags	Pointer to array of edge flag. The array should consist of (*Count*/4) elements. One byte per quadrilaterals. Specifies which quadrilaterals edge will be antialiased. can be combined with OR setting from followings. R_DRW2D_EDGE_AB R_DRW2D_EDGE_BC R_DRW2D_EDGE_CD R_DRW2D_EDGE_DA See r_drw2d_EdgeFlag_t. If NULL, do not use antialiasing.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_DEVICE_INTERNAL_ALLOCCLUT- Internal device driver error (while handling CLUT memory).
R_DRW2D_ERR_FRAMEBUFFER_HANDLE - Invalid framebuffer handle.
R_DRW2D_ERR_TEXTURE_ADDR - Invalid texture buffer address.
R_DRW2D_ERR_TEXTURE_PIXELFMT - Unsupported texel format.
R_DRW2D_ERR_TEXTURE_RLE_BPP - Bits per texel not suitable for RLE decoder (D/AVE HD specific).
R_DRW2D_ERR_EFFECT_INVALID_TEXTURE - Invalid Texture Index.
R_DRW2D_ERR_BUFFER_PIXELFMT - Invalid/unsupported pixel format.
R_DRW2D_ERR_BUFFER_ALIGNMENT - Buffer alignment not correct (for framebuffer).
R_DRW2D_ERR_INVALID_VALUE_NULLPTR - Parameter pointer argument is NULL.
R_DRW2D_ERR_INVALID_VALUE_FILLMODE - Invalid fill mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDMODE - Invalid blend mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCRGB_UNSUPPORTED - Unsupported SrcRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTRGB_UNSUPPORTED - Unsupported DstRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCALPHA_UNSUPPORTED - Unsupported SrcAlpha blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTALPHA_UNSUPPORTED - Unsupported DstAlpha blend factor.
R_DRW2D_ERR_DRAWING_DRAWQUAD - Failed to draw quad.
R_DRW2D_ERR_EFFECT_INVALID_OPERAND - Invalid Parameters provided for effects.
R_DRW2D_ERR_EFFECT_INVALID_OPERATION - Invalid effect name.
R_DRW2D_ERR_EFFECT_OUT_OF_RESOURCES - Combination of effects cannot be realized.

Description

Renders an array of quadrilaterals.
Each vertex will be transformed by the current vertex matrix.
If texture mapping is enabled, the texture will be applied according to the current texture transformation matrix.
Note that the D/AVE HD platform is limited to an edge width/height of max. 2048 when using non antialiased edges (Edge flag set to 0).
For the setting range of parameter, see ***Table 6-2***.

See also

R_DRW2D_DrawQuadsUV, R_DRW2D_DrawTriangles, R_DRW2D_DrawTrianglesUV, R_DRW2D_DrawPolyline, R_DRW2D_DrawLines, R_DRW2D_DrawEllipse, R_DRW2D_FramebufferClear, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_EdgeFlag_t, uint32_t, uint8_t

4.2.10.6 R_DRW2D_DrawQuadsUV

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_DrawQuadsUV(r_drw2d_Device_t          Device,
                                    const r_drw2d_Point_t    *Points,
                                    uint32_t                  Count,
                                    uint8_t                  *EdgeFlags,
                                    const r_drw2d_UVCoord_t  *UVCoords)
```

Parameter

Table 4-62 Parameter of R_DRW2D_DrawQuadsUV

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Points	Pointer to array of vertices (see r_drw2d_Point_t). The array should consist of *Count* elements.
Count	Number of vertices. (*Count*/4) quadrilaterals will be drawn.
EdgeFlags	Pointer to array of edge flag. The array should consist of (*Count*/4) elements. One byte per quadrilaterals. Specifies which quadrilaterals edge will be antialiased. can be combined with OR setting from followings. R_DRW2D_EDGE_AB R_DRW2D_EDGE_BC R_DRW2D_EDGE_CD R_DRW2D_EDGE_DA See r_drw2d_EdgeFlag_t. If NULL, do not use antialiasing.
UVCoords	Pointer to array of UV coordinates (see r_drw2d_UVCoord_t). The array should consist of *Count* elements.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_DEVICE_INTERNAL_ALLOCCLUT- Internal device driver error (while handling CLUT memory).
R_DRW2D_ERR_FRAMEBUFFER_HANDLE - Invalid framebuffer handle.
R_DRW2D_ERR_TEXTURE_ADDR - Invalid texture buffer address.
R_DRW2D_ERR_TEXTURE_PIXELFMT - Unsupported texel format.
R_DRW2D_ERR_TEXTURE_RLE_BPP - Bits per texel not suitable for RLE decoder (D/AVE HD specific).
R_DRW2D_ERR_EFFECT_INVALID_TEXTURE - Invalid Texture Index.
R_DRW2D_ERR_BUFFER_PIXELFMT - Invalid/unsupported pixel format.
R_DRW2D_ERR_BUFFER_ALIGNMENT - Buffer alignment not correct (for framebuffer).
R_DRW2D_ERR_INVALID_VALUE_NULLPTR - Parameter pointer argument is NULL.
R_DRW2D_ERR_INVALID_VALUE_FILLMODE - Invalid fill mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDMODE - Invalid blend mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCRGB_UNSUPPORTED - Unsupported SrcRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTRGB_UNSUPPORTED - Unsupported DstRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCALPHA_UNSUPPORTED - Unsupported SrcAlpha blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTALPHA_UNSUPPORTED - Unsupported DstAlpha blend factor.
R_DRW2D_ERR_DRAWING_DRAWTRIUV - Failed to draw UV texture mapped triangle.
R_DRW2D_ERR_EFFECT_INVALID_OPERAND - Invalid Parameters provided for effects.
R_DRW2D_ERR_EFFECT_INVALID_OPERATION - Invalid effect name.
R_DRW2D_ERR_EFFECT_OUT_OF_RESOURCES - Combination of effects cannot be realized.

Description

Renders an array of UV texture mapped quadrilaterals.
Each vertex will be transformed by the current vertex matrix.
Each UV coordinate will be transformed by the current texture transformation matrix if the texture transform mode R_DRW2D_CtxTextureTransformMode is set to R_DRW2D_TEX_TRANSFORM_2D.
The triangle texture mapping is performed twice internally, so texture maps correctly only for the linear transformation (e.g. from rectangle to parallelogram).
If the fill mode is set to R_DRW2D_FILLMODE_SOLID, this function behaves like R_DRW2D_DrawQuads and the UV coordinate array is ignored.
Note that the D/AVE HD platform is limited to an edge width/height of max. 2048 when using non antialiased edges (Edge flag set to 0).
For the setting range of parameter, see ***Table 6-2***.

See also

R_DRW2D_DrawQuads, R_DRW2D_DrawTriangles, R_DRW2D_DrawTrianglesUV, R_DRW2D_DrawPolyline, R_DRW2D_DrawLines, R_DRW2D_DrawEllipse, R_DRW2D_FramebufferClear, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_Point_t, r_drw2d_EdgeFlag_t, uint32_t, uint8_t, r_drw2d_UVCoord_t

4.2.10.7 R_DRW2D_DrawQuads3dUV

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_DrawQuads3dUV(r_drw2d_Device_t          Device,
                                      const r_drw2d_Vec4_t     *Points,
                                      uint32_t                 Count,
                                      uint8_t                  *EdgeFlags,
                                      const r_drw2d_UVCoord_t  *UVCoords)
```

Parameter

Table 4-63 Parameter of R_DRW2D_DrawQuads3dUV

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Points	Pointer to array of 3D vertices (see r_drw2d_Vec4_t). The array should consist of *Count* elements.
Count	Number of vertices. (*Count*/4) quadrilaterals will be drawn.
EdgeFlags	Pointer to array of edge flag. The array should consist of (*Count*/4) elements. One byte per quadrilaterals. Specifies which quadrilaterals edge will be antialiased. can be combined with OR setting from followings. R_DRW2D_EDGE_AB R_DRW2D_EDGE_BC R_DRW2D_EDGE_CD R_DRW2D_EDGE_DA See r_drw2d_EdgeFlag_t. If NULL, do not use antialiasing.
UVCoords	Pointer to array of UV coordinates (see r_drw2d_UVCoord_t). The array should consist of *Count* elements.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_DEVICE_INTERNAL_ALLOCCLUT- Internal device driver error (while handling CLUT memory).
R_DRW2D_ERR_FRAMEBUFFER_HANDLE - Invalid framebuffer handle.
R_DRW2D_ERR_TEXTURE_ADDR - Invalid texture buffer address.
R_DRW2D_ERR_TEXTURE_PIXELFMT - Unsupported texel format.
R_DRW2D_ERR_TEXTURE_RLE_BPP - Bits per texel not suitable for RLE decoder (D/AVE HD specific).
R_DRW2D_ERR_EFFECT_INVALID_TEXTURE - Invalid Texture Index.
R_DRW2D_ERR_BUFFER_PIXELFMT - Invalid/unsupported pixel format.
R_DRW2D_ERR_BUFFER_ALIGNMENT - Buffer alignment not correct (for framebuffer).
R_DRW2D_ERR_INVALID_VALUE_NULLPTR - Parameter pointer argument is NULL.
R_DRW2D_ERR_INVALID_VALUE_FILLMODE - Invalid fill mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDMODE - Invalid blend mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCRGB_UNSUPPORTED - Unsupported SrcRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTRGB_UNSUPPORTED - Unsupported DstRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCALPHA_UNSUPPORTED - Unsupported SrcAlpha blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTALPHA_UNSUPPORTED - Unsupported DstAlpha blend factor.
R_DRW2D_ERR_DRAWING_DRAWTRIUV - Failed to draw UV texture mapped triangle.
R_DRW2D_ERR_EFFECT_INVALID_OPERAND - Invalid Parameters provided for effects.
R_DRW2D_ERR_EFFECT_INVALID_OPERATION - Invalid effect name.
R_DRW2D_ERR_EFFECT_OUT_OF_RESOURCES - Combination of effects cannot be realized.

Description

Renders an array of UV texture mapped 3D-quadrilaterals.
Each vertex will be transformed by the current vertex matrix.
Each UV coordinate will be transformed by the current texture transformation matrix if the texture transform mode R_DRW2D_CtxTextureTransformMode is set to R_DRW2D_TEX_TRANSFORM_2D.
If the fill mode is set to R_DRW2D_FILLMODE_SOLID, this function behaves like R_DRW2D_DrawQuads and the UV coordinate array is ignored.
Note that the D/AVE HD platform is limited to an edge width/height of max. 2048 when using non antialiased edges (Edge flag set to 0).
For the setting range of parameter, see ***Table 6-2***.

See also

R_DRW2D_DrawQuads, R_DRW2D_DrawQuadsUV, R_DRW2D_DrawTriangles, R_DRW2D_DrawTrianglesUV, R_DRW2D_DrawPolyline, R_DRW2D_DrawLines, R_DRW2D_DrawEllipse, R_DRW2D_FramebufferClear, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_Vec4_t, r_drw2d_EdgeFlag_t, uint32_t, uint8_t, r_drw2d_UVCoord_t

4.2.10.8 R_DRW2D_DrawEllipse

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_DrawEllipse(r_drw2d_Device_t  Device,
                                    r_drw2d_Point_t   Point,
                                    r_drw2d_FixedP_t  RadiusX,
                                    r_drw2d_FixedP_t  RadiusY)
```

Parameter

Table 4-64 Parameter of R_DRW2D_DrawEllipse

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Point	Center point (see r_drw2d_Point_t).
RadiusX	Horizontal radius.
RadiusY	Vertical radius.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_DEVICE_INTERNAL_ALLOCCLUT- Internal device driver error (while handling CLUT memory).
R_DRW2D_ERR_FRAMEBUFFER_HANDLE - Invalid framebuffer handle.
R_DRW2D_ERR_TEXTURE_ADDR - Invalid texture buffer address.
R_DRW2D_ERR_TEXTURE_PIXELFMT - Unsupported texel format.
R_DRW2D_ERR_TEXTURE_RLE_BPP - Bits per texel not suitable for RLE decoder (D/AVE HD specific).
R_DRW2D_ERR_EFFECT_INVALID_TEXTURE - Invalid Texture Index.
R_DRW2D_ERR_BUFFER_PIXELFMT - Invalid/unsupported pixel format.
R_DRW2D_ERR_BUFFER_ALIGNMENT - Buffer alignment not correct (for framebuffer).
R_DRW2D_ERR_INVALID_VALUE_FILLMODE - Invalid fill mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDMODE - Invalid blend mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCRGB_UNSUPPORTED - Unsupported SrcRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTRGB_UNSUPPORTED - Unsupported DstRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCALPHA_UNSUPPORTED - Unsupported SrcAlpha blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTALPHA_UNSUPPORTED - Unsupported DstAlpha blend factor.
R_DRW2D_ERR_EFFECT_INVALID_OPERAND - Invalid Parameters provided for effects.
R_DRW2D_ERR_EFFECT_INVALID_OPERATION - Invalid effect name.
R_DRW2D_ERR_EFFECT_OUT_OF_RESOURCES - Combination of effects cannot be realized.

Description

Render an ellipse at Point with the specified x and y radius.
Only the center point will be transformed by the current vertex matrix. Radius is not transformed by current transformation matrix. So, transformation such as scaling or rotation are not reflected.
In order to draw a circle, use the same values for RadiusX and RadiusY.
If texture mapping is enabled, the texture will be applied according to the current texture transformation matrix.
It is necessary to set the image quality to R_DRW2D_IMGQUALITY_MEDIUM or R_DRW2D_IMGQUALITY_HIGH (see R_DRW2D_CtxImgQuality) for bezier curve to draw.
For the setting range of parameter, see ***Table 6-2***.

See also

R_DRW2D_DrawTriangles, R_DRW2D_DrawTrianglesUV, R_DRW2D_DrawPolyline, R_DRW2D_DrawLines, R_DRW2D_DrawRect, R_DRW2D_FramebufferClear, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_Point_t, r_drw2d_FixedP_t

4.2.10.9 R_DRW2D_DrawLines

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_DrawLines(r_drw2d_Device_t        Device,
                                  const r_drw2d_Point_t  *Points,
                                  uint32_t               Count)
```

Parameter

Table 4-65 Parameter of R_DRW2D_DrawLines

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Points	Pointer to array of Line start/end vertices (see r_drw2d_Point_t). The array should consist of *Count* elements. The order is (start of the 1st Line), (end of the 1st Line), (start of the 2nd Line), (end of the 2nd Line), ...
Count	Number of vertices. (*Count*/2) lines will be drawn.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_DEVICE_INTERNAL_ALLOCCLUT- Internal device driver error (while handling CLUT memory).
R_DRW2D_ERR_FRAMEBUFFER_HANDLE - Invalid framebuffer handle.
R_DRW2D_ERR_TEXTURE_ADDR - Invalid texture buffer address.
R_DRW2D_ERR_TEXTURE_PIXELFMT - Unsupported texel format.
R_DRW2D_ERR_TEXTURE_RLE_BPP - Bits per texel not suitable for RLE decoder (D/AVE HD specific).
R_DRW2D_ERR_EFFECT_INVALID_TEXTURE - Invalid Texture Index.
R_DRW2D_ERR_BUFFER_PIXELFMT - Invalid/unsupported pixel format.
R_DRW2D_ERR_BUFFER_ALIGNMENT - Buffer alignment not correct (for framebuffer).
R_DRW2D_ERR_INVALID_VALUE_NULLPTR - Parameter pointer argument is NULL.
R_DRW2D_ERR_INVALID_VALUE_FILLMODE - Invalid fill mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDMODE - Invalid blend mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCRGB_UNSUPPORTED - Unsupported SrcRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTRGB_UNSUPPORTED - Unsupported DstRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCALPHA_UNSUPPORTED - Unsupported SrcAlpha blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTALPHA_UNSUPPORTED - Unsupported DstAlpha blend factor.
R_DRW2D_ERR_INVALID_VALUE_DRAWLINES_ODDPOINTCOUNT - Odd number of points passed to R_DRW2D_DrawLines.
R_DRW2D_ERR_EFFECT_INVALID_OPERAND - Invalid Parameters provided for effects.
R_DRW2D_ERR_EFFECT_INVALID_OPERATION - Invalid effect name.
R_DRW2D_ERR_EFFECT_OUT_OF_RESOURCES - Combination of effects cannot be realized.

Description

Render an array of lines. The rendered lines include the start and end points.
Each line point and line width will be transformed by the current transformation matrix.
The Width field of r_drw2d_LineStyle_t specifies the line width.
The LineCap field of r_drw2d_LineStyle_t specifies how to render the line starts/ends (see r_drw2d_LineCap_t).
If R_DRW2D_LINECAP_ROUND is specified for LineCap, the scaling is not applied to the cap part.
Line color is specified by R_DRW2D_CtxFgColor. Transparency cannot be used for line color (i.e. Alpha value should be 0xFF) when R_DRW2D_LINECAP_ROUND is specified for LineCap.
If texture mapping is enabled, the texture will be applied according to the current texture transformation matrix.
For the setting range of parameter, see ***Table 6-2***.

See also

R_DRW2D_DrawTriangles, R_DRW2D_DrawTrianglesUV, R_DRW2D_DrawPolyline, R_DRW2D_DrawEllipse, R_DRW2D_DrawRect, R_DRW2D_FramebufferClear, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_Point_t, uint32_t

4.2.10.10 R_DRW2D_DrawPolyline

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_DrawPolyline(r_drw2d_Device_t         Device,
                                     const r_drw2d_Point_t   *Points,
                                     uint32_t                 Count)
```

Parameter

Table 4-66 Parameter of R_DRW2D_DrawPolyline

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Points	Pointer to array of Line start/end vertices (see r_drw2d_Point_t). The array should consist of *Count* elements.
Count	Number of vertices. (*Count*-1) or (*Count*) lines will be drawn depending on IsClosed setting.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_DEVICE_INTERNAL_ALLOCCLUT- Internal device driver error (while handling CLUT memory).
R_DRW2D_ERR_FRAMEBUFFER_HANDLE - Invalid framebuffer handle.
R_DRW2D_ERR_TEXTURE_ADDR - Invalid texture buffer address.
R_DRW2D_ERR_TEXTURE_PIXELFMT - Unsupported texel format.
R_DRW2D_ERR_TEXTURE_RLE_BPP - Bits per texel not suitable for RLE decoder (D/AVE HD specific).
R_DRW2D_ERR_EFFECT_INVALID_TEXTURE - Invalid Texture Index.
R_DRW2D_ERR_BUFFER_PIXELFMT - Invalid/unsupported pixel format.
R_DRW2D_ERR_BUFFER_ALIGNMENT - Buffer alignment not correct (for framebuffer).
R_DRW2D_ERR_INVALID_VALUE_NULLPTR - Parameter pointer argument is NULL.
R_DRW2D_ERR_INVALID_VALUE_FILLMODE - Invalid fill mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDMODE - Invalid blend mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCRGB_UNSUPPORTED - Unsupported SrcRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTRGB_UNSUPPORTED - Unsupported DstRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCALPHA_UNSUPPORTED - Unsupported SrcAlpha blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTALPHA_UNSUPPORTED - Unsupported DstAlpha blend factor.
R_DRW2D_ERR_INVALID_VALUE_DRAWLINES_ODDPOINTCOUNT - Odd number of points passed to R_DRW2D_DrawLines.
R_DRW2D_ERR_INVALID_VALUE_POLYLINE_COUNT - Invalid polyline point count (0 or 1).
R_DRW2D_ERR_DRAWING_DRAWTRI - Failed to draw triangle.
R_DRW2D_ERR_EFFECT_INVALID_OPERAND - Invalid Parameters provided for effects.
R_DRW2D_ERR_EFFECT_INVALID_OPERATION - Invalid effect name.
R_DRW2D_ERR_EFFECT_OUT_OF_RESOURCES - Combination of effects cannot be realized.

Description

Render a polyline consisting of one or many line segments.
Each line point and line width will be transformed by the current transformation matrix.
The Width field of r_drw2d_LineStyle_t specifies the line width.
If the IsClosed field of r_drw2d_LineStyle_t is set to 1 (true), the last segment will be connected to the first segment.
The line segments will be connected as specified by the LineJoin field of r_drw2d_LineStyle_t (see R_DRW2D_CtxLineStyle, r_drw2d_LineStyle_t, r_drw2d_LineJoin_t).
If the R_DRW2D_LINEJOIN_MITER join type is selected, the MiterLimit field of r_drw2d_LineStyle_t specifies the maximum distance between the line join tip and line point. If the miter limit is exceeded, a bevel joint will be drawn at the miter limit position.
If the polyline is not closed, the LineCap field of r_drw2d_LineStyle_t specifies how to render the polyline start/end (see r_drw2d_LineCap_t).
If R_DRW2D_LINEJOIN_ROUND is specified for LineJoin, the scaling is not applied to the join part.
If R_DRW2D_LINECAP_ROUND is specified for LineCap, the scaling is not applied to the cap part.
Line color is specified by R_DRW2D_CtxFgColor. Transparency cannot be used for line color (i.e. Alpha value should be 0xFF).
If texture mapping is enabled, the texture will be applied according to the current texture transformation matrix.
For the setting range of parameter, see ***Table 6-2***.

See also

R_DRW2D_DrawTriangles, R_DRW2D_DrawTrianglesUV, R_DRW2D_DrawLines, R_DRW2D_DrawEllipse, R_DRW2D_DrawRect, R_DRW2D_FramebufferClear, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_Point_t, uint32_t

4.2.10.11 R_DRW2D_DrawBezierCurves

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_DrawBezierCurves(r_drw2d_Device_t           Device,
                                         const r_drw2d_Point_t     *Points,
                                         uint32_t                  Count)
```

Parameter

Table 4-67 Parameter of R_DRW2D_DrawBezierCurves

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Points	Pointer to array of curve point vertices (see r_drw2d_Point_t). The array should consist of *Count* elements.
Count	Number of vertices. ((Count - 1) / 2) curves will be drawn

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_DEVICE_INTERNAL_ALLOCCLUT- Internal device driver error (while handling CLUT memory).
R_DRW2D_ERR_FRAMEBUFFER_HANDLE - Invalid framebuffer handle.
R_DRW2D_ERR_TEXTURE_ADDR - Invalid texture buffer address.
R_DRW2D_ERR_TEXTURE_PIXELFMT - Unsupported texel format.
R_DRW2D_ERR_TEXTURE_RLE_BPP - Bits per texel not suitable for RLE decoder (D/AVE HD specific).
R_DRW2D_ERR_EFFECT_INVALID_TEXTURE - Invalid Texture Index.
R_DRW2D_ERR_BUFFER_PIXELFMT - Invalid/unsupported pixel format.
R_DRW2D_ERR_BUFFER_ALIGNMENT - Buffer alignment not correct (for framebuffer).
R_DRW2D_ERR_INVALID_VALUE_NULLPTR - Parameter pointer argument is NULL.
R_DRW2D_ERR_INVALID_VALUE_FILLMODE - Invalid fill mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDMODE - Invalid blend mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCRGB_UNSUPPORTED - Unsupported SrcRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTRGB_UNSUPPORTED - Unsupported DstRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCALPHA_UNSUPPORTED - Unsupported SrcAlpha blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTALPHA_UNSUPPORTED - Unsupported DstAlpha blend factor.
R_DRW2D_ERR_INVALID_VALUE_DRAWLINES_ODDPOINTCOUNT - Odd number of points passed to R_DRW2D_DrawLines.
R_DRW2D_ERR_INVALID_VALUE_POLYBEZIER_COUNT - Invalid bezier curves point count (0, 1 or 2).
R_DRW2D_ERR_EFFECT_INVALID_OPERAND - Invalid Parameters provided for effects.
R_DRW2D_ERR_EFFECT_INVALID_OPERATION - Invalid effect name.
R_DRW2D_ERR_EFFECT_OUT_OF_RESOURCES - Combination of effects cannot be realized.

Description

Render a bezier curve consisting of one or more quadratic bezier segments.
Only each curve point will be transformed by the current transformation matrix.
The Width field of r_drw2d_LineStyle_t specifies the line width. The value must be less than 16.
Line width is not transformed by current transformation matrix.
If the IsClosed field of r_drw2d_LineStyle_t is set to 1 (true), the last segment will be connected to the first segment by a straight line.
The bezier segments will be connected by round joints. The ends of the curve will be flat.
Line color is specified by R_DRW2D_CtxFgColor. Transparency cannot be used for line color (i.e. Alpha value should be 0xFF) when multiple curves are drawn.
It is necessary to set the image quality to R_DRW2D_IMGQUALITY_MEDIUM or R_DRW2D_IMGQUALITY_HIGH (see R_DRW2D_CtxImgQuality) for bezier curve to draw.
For the setting range of parameter, see ***Table 6-2***.

See also

R_DRW2D_DrawTriangles, R_DRW2D_DrawTrianglesUV, R_DRW2D_DrawLines, R_DRW2D_DrawEllipse, R_DRW2D_DrawRect, R_DRW2D_FramebufferClear, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_Point_t, uint32_t

4.2.11 Convolution filter functions

The convolution filter render functions take the following context settings into account

- source texture (see R_DRW2D_CtxTextureSet)
- destination framebuffer (see R_DRW2D_FramebufferSet)
- convolution filter kernel (see R_DRW2D_CtxConvolutionKernel, R_DRW2D_CtxConvolutionKernelPreset1d, R_DRW2D_CtxConvolutionKernelPreset2d)
- clipping rectangle (see R_DRW2D_CtxClipRect)

4.2.11.1 R_DRW2D_DrawRectConvolve1dx

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_DrawRectConvolve1dx(r_drw2d_Device_t        Device,
                                            const r_drw2d_IntRect_t *Rect,
                                            uint16_t                TextureOffX,
                                            uint16_t                TextureOffY)
```

Parameter

Table 4-68 Parameter of R_DRW2D_DrawRectConvolve1dx

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Rect	Rectangle position and size (see r_drw2d_IntRect_t).
TextureOffX	Horizontal texture offset (number of texels).
TextureOffY	Vertical texture offset (number of texels).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_DEVICE_INTERNAL_SWIZZLEVT - Internal device driver error. (trying to mix swizzling + virtual tiling).
R_DRW2D_ERR_DEVICE_INTERNAL_ALLOCCLUT- Internal device driver error (while handling CLUT memory).
R_DRW2D_ERR_DEVICE_PIXELFMT - Pixel format not supported by device.
R_DRW2D_ERR_CONVOLUTION_RES_CONFLICT - Resource conflict. Color unit already in use by effect.
R_DRW2D_ERR_FRAMEBUFFER_HANDLE - Invalid framebuffer handle.
R_DRW2D_ERR_TEXTURE_ADDR - Invalid texture buffer address.
R_DRW2D_ERR_TEXTURE_PIXELFMT - Unsupported texel format.
R_DRW2D_ERR_TEXTURE_RLE_BPP - Bits per texel not suitable for RLE decoder (D/AVE HD specific).
R_DRW2D_ERR_TEXTURE_WIDTH - Invalid texture width.
R_DRW2D_ERR_TEXTURE_HEIGHT - Invalid texture height.
R_DRW2D_ERR_EFFECT_INVALID_TEXTURE - Invalid Texture Index.
R_DRW2D_ERR_BUFFER_PIXELFMT - Invalid/unsupported pixel format.
R_DRW2D_ERR_BUFFER_ALIGNMENT - Buffer alignment not correct (for framebuffer).
R_DRW2D_ERR_INVALID_VALUE_NULLPTR - Parameter pointer argument is NULL.
R_DRW2D_ERR_INVALID_VALUE_FILLMODE - Invalid fill mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDMODE - Invalid blend mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCRGB_UNSUPPORTED - Unsupported SrcRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTRGB_UNSUPPORTED - Unsupported DstRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCALPHA_UNSUPPORTED - Unsupported SrcAlpha blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTALPHA_UNSUPPORTED - Unsupported DstAlpha blend factor.
R_DRW2D_ERR_EFFECT_INVALID_OPERAND - Invalid Parameters provided for effects.
R_DRW2D_ERR_EFFECT_INVALID_OPERATION - Invalid effect name.
R_DRW2D_ERR_EFFECT_OUT_OF_RESOURCES - Combination of effects cannot be realized.

Description

Apply one dimensional convolution filter to texture and store result in framebuffer.
The currently selected 1D convolution kernel is applied in the horizontal direction, i.e. the kernel coefficients determine the weights of the pixel(s) to the left and right of the current texel while the source texture is being traversed.
The Pos field of the Rect argument is used to select the framebuffer destination position.
The Size field of the Rect argument is used to select the width and height of the convoluted area.
The TextureOffX and TextureOffY arguments are used to select the left/top texel of the convolution source area.
TextureOffX must be less than (texture width - 8). TextureOffY must be less than the (texture height - 8).
The texture width and height must be greater than 8.
R_DRW2D_TEX_WRAPU and R_DRW2D_TEX_WRAPV flags are not available.
This function does not regard the vertex and texture transformation matrices.
The fill mode must be set to R_DRW2D_FILLMODE_TEXTURE.
Note that using textures with an improper pitch or height on D/AVE HD will lead to severe performance loss. The texture's pitch should be a multiple of 256 / BPP, where BPP is the current texture format's number of bits per pixel. The texture's height should be a multiple of 4. Furthermore, the texture's start address should be 8 byte aligned.
For the setting range of parameter, see ***Table 6-2***.
Convolution filter cannot be applied when texture color format is AL1, AL2, AL4, CLUT1, CLUT2, CLUT4 or CLUT8.

See also

R_DRW2D_DrawRectConvolve1dy, R_DRW2D_DrawRectConvolve2d, R_DRW2D_CtxConvolutionKernelPreset1d, R_DRW2D_CtxConvolutionKernelPreset2d, R_DRW2D_DrawTriangles, R_DRW2D_DrawTrianglesUV, R_DRW2D_DrawPolyline, R_DRW2D_DrawLines, R_DRW2D_DrawEllipse, R_DRW2D_FramebufferClear, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_IntRect_t, uint16_t

4.2.11.2 R_DRW2D_DrawRectConvolve1dy

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_DrawRectConvolve1dy(r_drw2d_Device_t        Device,
                                            const r_drw2d_IntRect_t *Rect,
                                            uint16_t                TextureOffX,
                                            uint16_t                TextureOffY)
```

Parameter

Table 4-69 Parameter of R_DRW2D_DrawRectConvolve1dy

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Rect	Rectangle position and size (see r_drw2d_IntRect_t).
TextureOffX	Horizontal texture offset (number of texels).
TextureOffY	Vertical texture offset (number of texels).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_DEVICE_INTERNAL_SWIZZLEVT - Internal device driver error. (trying to mix swizzling + virtual tiling).
R_DRW2D_ERR_DEVICE_INTERNAL_ALLOCCLUT- Internal device driver error (while handling CLUT memory).
R_DRW2D_ERR_DEVICE_PIXELFMT - Pixel format not supported by device.
R_DRW2D_ERR_CONVOLUTION_RES_CONFLICT - Resource conflict. Color unit already in use by effect.
R_DRW2D_ERR_FRAMEBUFFER_HANDLE - Invalid framebuffer handle.
R_DRW2D_ERR_TEXTURE_ADDR - Invalid texture buffer address.
R_DRW2D_ERR_TEXTURE_PIXELFMT - Unsupported texel format.
R_DRW2D_ERR_TEXTURE_RLE_BPP - Bits per texel not suitable for RLE decoder (D/AVE HD specific).
R_DRW2D_ERR_TEXTURE_WIDTH - Invalid texture width.
R_DRW2D_ERR_TEXTURE_HEIGHT - Invalid texture height.
R_DRW2D_ERR_EFFECT_INVALID_TEXTURE - Invalid Texture Index.
R_DRW2D_ERR_BUFFER_PIXELFMT - Invalid/unsupported pixel format.
R_DRW2D_ERR_BUFFER_ALIGNMENT - Buffer alignment not correct (for framebuffer).
R_DRW2D_ERR_INVALID_VALUE_NULLPTR - Parameter pointer argument is NULL.
R_DRW2D_ERR_INVALID_VALUE_FILLMODE - Invalid fill mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDMODE - Invalid blend mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCRGB_UNSUPPORTED - Unsupported SrcRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTRGB_UNSUPPORTED - Unsupported DstRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCALPHA_UNSUPPORTED - Unsupported SrcAlpha blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTALPHA_UNSUPPORTED - Unsupported DstAlpha blend factor.
R_DRW2D_ERR_EFFECT_INVALID_OPERAND - Invalid Parameters provided for effects.
R_DRW2D_ERR_EFFECT_INVALID_OPERATION - Invalid effect name.
R_DRW2D_ERR_EFFECT_OUT_OF_RESOURCES - Combination of effects cannot be realized.

Description

Apply one dimensional convolution filter to texture and store result in framebuffer.
The currently selected 1D convolution kernel is applied in the vertical direction, i.e. the kernel coefficients determine the weights of the pixel(s) above and below the current texel while the source texture is being traversed.
The Pos field of the Rect argument is used to select the framebuffer destination position. The Size field of the Rect argument is used to select the width and height of the convoluted area.
The TextureOffX and TextureOffY arguments are used to select the left/top texel of the convolution source area. TextureOffX must be less than (texture width - 8). TextureOffY must be less than the (texture height - 8).
The texture width and height must be greater than 8.
R_DRW2D_TEX_WRAPU and R_DRW2D_TEX_WRAPV flags are not available.
This function does not regard the vertex and texture transformation matrices.
The fill mode must be set to R_DRW2D_FILLMODE_TEXTURE.
Note that using textures with an improper pitch or height on D/AVE HD will lead to severe performance loss. The texture's pitch should be a multiple of 256 / BPP, where BPP is the current texture format's number of bits per pixel. The texture's height should be a multiple of 4. Furthermore, the texture's start address should be 8 byte aligned.
For the setting range of parameter, see ***Table 6-2***.
Convolution filter cannot be applied when texture color format is AL1, AL2, AL4, CLUT1, CLUT2, CLUT4 or CLUT8.

See also

R_DRW2D_DrawRectConvolve1dx,R_DRW2D_CtxConvolutionKernelPreset1d, R_DRW2D_DrawRectConvolve2d, R_DRW2D_CtxConvolutionKernelPreset2d, R_DRW2D_DrawTriangles, R_DRW2D_DrawTrianglesUV, R_DRW2D_DrawPolyline, R_DRW2D_DrawLines, R_DRW2D_DrawEllipse, R_DRW2D_FramebufferClear, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_IntRect_t, uint16_t

4.2.11.3 R_DRW2D_DrawRectConvolve2d

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_DrawRectConvolve2d(r_drw2d_Device_t         Device,
                                           const r_drw2d_IntRect_t *Rect,
                                           uint16_t                 TextureOffX,
                                           uint16_t                 TextureOffY)
```

Parameter

Table 4-70 Parameter of R_DRW2D_DrawRectConvolve2d

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Rect	Rectangle position and size (see r_drw2d_IntRect_t).
TextureOffX	Horizontal texture offset (number of texels).
TextureOffY	Vertical texture offset (number of texels).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_DEVICE_INTERNAL_SWIZZLEVT - Internal device driver error. (trying to mix swizzling + virtual tiling).
R_DRW2D_ERR_DEVICE_INTERNAL_ALLOCCLUT- Internal device driver error (while handling CLUT memory).
R_DRW2D_ERR_DEVICE_PIXELFMT - Pixel format not supported by device.
R_DRW2D_ERR_CONVOLUTION_RES_CONFLICT - Resource conflict. Color unit already in use by effect.
R_DRW2D_ERR_FRAMEBUFFER_HANDLE - Invalid framebuffer handle.
R_DRW2D_ERR_TEXTURE_ADDR - Invalid texture buffer address.
R_DRW2D_ERR_TEXTURE_PIXELFMT - Unsupported texel format.
R_DRW2D_ERR_TEXTURE_RLE_BPP - Bits per texel not suitable for RLE decoder (D/AVE HD specific).
R_DRW2D_ERR_TEXTURE_WIDTH - Invalid texture width.
R_DRW2D_ERR_TEXTURE_HEIGHT - Invalid texture height.
R_DRW2D_ERR_EFFECT_INVALID_TEXTURE - Invalid Texture Index.
R_DRW2D_ERR_BUFFER_PIXELFMT - Invalid/unsupported pixel format.
R_DRW2D_ERR_BUFFER_ALIGNMENT - Buffer alignment not correct (for framebuffer).
R_DRW2D_ERR_INVALID_VALUE_NULLPTR - Parameter pointer argument is NULL.
R_DRW2D_ERR_INVALID_VALUE_FILLMODE - Invalid fill mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDMODE - Invalid blend mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCRGB_UNSUPPORTED - Unsupported SrcRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTRGB_UNSUPPORTED - Unsupported DstRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCALPHA_UNSUPPORTED - Unsupported SrcAlpha blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTALPHA_UNSUPPORTED - Unsupported DstAlpha blend factor.
R_DRW2D_ERR_EFFECT_INVALID_OPERAND - Invalid Parameters provided for effects.
R_DRW2D_ERR_EFFECT_INVALID_OPERATION - Invalid effect name.
R_DRW2D_ERR_EFFECT_OUT_OF_RESOURCES - Combination of effects cannot be realized.

Description

Apply two dimensional convolution filter to texture and store result in framebuffer.The currently selected 2D convolution kernel is applied in both x and y directions, i.e. the kernel coefficients determine the weights of the pixel(s) above and below resp. to the left and right of the current texel while the source texture is being traversed.
The Pos field of the Rect argument is used to select the framebuffer destination position.
The Size field of the Rect argument is used to select the width and height of the convoluted area.
The TextureOffX and TextureOffY arguments are used to select the left/top texel of the convolution source area. TextureOffX must be less than (texture width - 8). TextureOffY must be less than the (texture height - 8).
The texture width and height must be greater than 8.
R_DRW2D_TEX_WRAPU and R_DRW2D_TEX_WRAPV flags are not available.
This function does not regard the vertex and texture transformation matrices.
If the selected convolution kernel is separable, i.e. if the 2d kernel matrix can be expressed as a product of a row and column vector (e.g. Gaussian blur), applications should convolute the texture in two passes using R_DRW2D_DrawRectConvolve1dx and R_DRW2D_DrawRectConvolve1dy. Doing so will decrease computational complexity from $O(n^2)$ to $O(n)$. The drawback of this approach is that a temporary buffer is required to store the result of the first pass.
Due to HW restrictions, not all separable kernels can be implemented using aforementioned two-pass rendering technique. In particular, the D/AVE HD cannot process separable kernels that contain negative weights (e.g. Sobel edge detect), since the result pixels are clamped to the 0..255 range after the first pass.
The fill mode must be set to R_DRW2D_FILLMODE_TEXTURE.
Note that using textures with an improper pitch or height on D/AVE HD will lead to severe performance loss. The texture's pitch should be a multiple of 256 / BPP, where BPP is the current texture format's number of bits per pixel. The texture's height should be a multiple of 4. Furthermore, the texture's start address should be 8 byte aligned.
For the setting range of parameter, see ***Table 6-2***.
Convolution filter cannot be applied when texture color format is AL1, AL2, AL4, CLUT1, CLUT2, CLUT4 or CLUT8.

See also

R_DRW2D_CtxConvolutionKernelPreset2d, R_DRW2D_DrawRectConvolve1dx, R_DRW2D_DrawRectConvolve1dy, R_DRW2D_CtxConvolutionKernelPreset1d, R_DRW2D_DrawTriangles, R_DRW2D_DrawTrianglesUV, R_DRW2D_DrawPolyline, R_DRW2D_DrawLines, R_DRW2D_DrawEllipse, R_DRW2D_FramebufferClear, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_IntRect_t, uint16_t

4.2.11.4 R_DRW2D_DrawRectConvolve

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_DrawRectConvolve(r_drw2d_Device_t           Device,
                                         const r_drw2d_IntRect_t   *Rect,
                                         uint16_t                  TextureOffX,
                                         uint16_t                  TextureOffY)
```

Parameter

Table 4-71 Parameter of R_DRW2D_DrawRectConvolve

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Rect	Rectangle position and size (see r_drw2d_IntRect_t).
TextureOffX	Horizontal texture offset (number of texels).
TextureOffY	Vertical texture offset (number of texels).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_DEVICE_INTERNAL_SWIZZLEVT - Internal device driver error. (trying to mix swizzling + virtual tiling).
R_DRW2D_ERR_DEVICE_INTERNAL_ALLOCCLUT- Internal device driver error (while handling CLUT memory).
R_DRW2D_ERR_DEVICE_PIXELFMT - Pixel format not supported by device.
R_DRW2D_ERR_CONVOLUTION_RES_CONFLICT - Resource conflict. Color unit already in use by effect.
R_DRW2D_ERR_FRAMEBUFFER_HANDLE - Invalid framebuffer handle.
R_DRW2D_ERR_TEXTURE_ADDR - Invalid texture buffer address.
R_DRW2D_ERR_TEXTURE_PIXELFMT - Unsupported texel format.
R_DRW2D_ERR_TEXTURE_RLE_BPP - Bits per texel not suitable for RLE decoder (D/AVE HD specific).
R_DRW2D_ERR_TEXTURE_WIDTH - Invalid texture width.
R_DRW2D_ERR_TEXTURE_HEIGHT - Invalid texture height.
R_DRW2D_ERR_EFFECT_INVALID_TEXTURE - Invalid Texture Index.
R_DRW2D_ERR_BUFFER_PIXELFMT - Invalid/unsupported pixel format.
R_DRW2D_ERR_BUFFER_ALIGNMENT - Buffer alignment not correct (for framebuffer).
R_DRW2D_ERR_INVALID_VALUE_NULLPTR - Parameter pointer argument is NULL.
R_DRW2D_ERR_INVALID_VALUE_FILLMODE - Invalid fill mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDMODE - Invalid blend mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCRGB_UNSUPPORTED - Unsupported SrcRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTRGB_UNSUPPORTED - Unsupported DstRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCALPHA_UNSUPPORTED - Unsupported SrcAlpha blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTALPHA_UNSUPPORTED - Unsupported DstAlpha blend factor.
R_DRW2D_ERR_EFFECT_INVALID_OPERAND - Invalid Parameters provided for effects.
R_DRW2D_ERR_EFFECT_INVALID_OPERATION - Invalid effect name.
R_DRW2D_ERR_EFFECT_OUT_OF_RESOURCES - Combination of effects cannot be realized.

Description

Apply two dimensional convolution filter to texture and store result in framebuffer.
The currently selected 2D convolution kernel is applied in both x and y directions, i.e. the kernel coefficients determine the weights of the pixel(s) above and below resp. to the left and right of the current texel while the source texture is being traversed.
The Pos field of the Rect argument is used to select the framebuffer destination position.
The Size field of the Rect argument is used to select the width and height of the convoluted area.
The TextureOffX and TextureOffY arguments are used to select the left/top texel of the convolution source area. TextureOffX must be less than (texture width - 8). TextureOffY must be less than the (texture height - 8).
The texture width and height must be greater than 8.
R_DRW2D_TEX_WRAPU and R_DRW2D_TEX_WRAPV flags are not available.
This function does not regard the vertex and texture transformation matrices.
If the selected convolution kernel is separable, i.e. if the 2d kernel matrix can be expressed as a product of a row and column vector (e.g. Gaussian blur), applications should convolute the texture in two passes using R_DRW2D_DrawRectConvolve1dx and R_DRW2D_DrawRectConvolve1dy. Doing so will decrease computational complexity from $O(n^2)$ to $O(n)$. The drawback of this approach is that a temporary buffer is required to store the result of the first pass.
Due to HW restrictions, not all separable kernels can be implemented using aforementioned two-pass rendering technique. In particular, the D/AVE HD cannot process separable kernels that contain negative weights (e.g. Sobel edge detect), since the result pixels are clamped to the 0..255 range after the first pass.
The fill mode must be set to R_DRW2D_FILLMODE_TEXTURE.
Note that using textures with an improper pitch or height on D/AVE HD will lead to severe performance loss. The texture's pitch should be a multiple of 256 / BPP, where BPP is the current texture format's number of bits per pixel. The texture's height should be a multiple of 4. Furthermore, the texture's start address should be 8 byte aligned.
For the setting range of parameter, see ***Table 6-2***.
Convolution filter cannot be applied when texture color format is AL1, AL2, AL4, CLUT1, CLUT2, CLUT4 or CLUT8.

See also

R_DRW2D_CtxConvolutionKernelPreset2d, R_DRW2D_DrawRectConvolve1dx, R_DRW2D_DrawRectConvolve1dy, R_DRW2D_CtxConvolutionKernelPreset1d, R_DRW2D_DrawTriangles, R_DRW2D_DrawTrianglesUV, R_DRW2D_DrawPolyline, R_DRW2D_DrawLines, R_DRW2D_DrawEllipse, R_DRW2D_FramebufferClear, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_IntRect_t, uint16_t

4.2.11.5 R_DRW2D_CtxConvolutionKernelPreset1d

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxConvolutionKernelPreset1d(
                                    r_drw2d_Device_t                    Device,
                                    r_drw2d_ConvolutionKernelPreset1d_t Preset)
```

Parameter

Table 4-72 Parameter of R_DRW2D_CtxConvolutionKernelPreset1d

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Preset	Kernel preset (see r_drw2d_ConvolutionKernelPreset1d_t).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK	- No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE	- Invalid device handle.
R_DRW2D_ERR_CONVOLUTION_ADDR	- Invalid coefficient array address.
R_DRW2D_ERR_CONVOLUTION_DIMENSION	- Invalid kernel dimensions.
R_DRW2D_ERR_INVALID_VALUE	- Parameter/argument value is out of range or undefined.
R_DRW2D_ERR_INVALID_VALUE_CONVKERNELPRESET1D	- Invalid 1D convolution preset.

Description

Select 1d convolution kernel size and weights.

See also

R_DRW2D_CtxConvolutionKernelPreset2d, R_DRW2D_DrawRectConvolve1dx, R_DRW2D_DrawRectConvolve1dy, R_DRW2D_DrawRectConvolve2d, R_DRW2D_DrawTriangles, R_DRW2D_DrawTrianglesUV, R_DRW2D_DrawPolyline, R_DRW2D_DrawLines, R_DRW2D_DrawEllipse, R_DRW2D_FramebufferClear, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_ConvolutionKernelPreset1d_t

4.2.11.6 R_DRW2D_CtxConvolutionKernelPreset2d

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxConvolutionKernelPreset2d(
                                    r_drw2d_Device_t                    Device,
                                    r_drw2d_ConvolutionKernelPreset2d_t Preset)
```

Parameter

Table 4-73 Parameter of R_DRW2D_CtxConvolutionKernelPreset2d

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Preset	Kernel preset (see r_drw2d_ConvolutionKernelPreset2d_t).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK	- No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE	- Invalid device handle.
R_DRW2D_ERR_CONVOLUTION_ADDR	- Invalid coefficient array address.
R_DRW2D_ERR_CONVOLUTION_DIMENSION	- Invalid kernel dimensions.
R_DRW2D_ERR_INVALID_VALUE	- Parameter/argument value is out of range or undefined.
R_DRW2D_ERR_INVALID_VALUE_CONVKERNELPRESET2D	- Invalid 2D convolution preset.

Description

Select 2d convolution kernel size and weights.

See also

R_DRW2D_CtxConvolutionKernelPreset1d, R_DRW2D_DrawRectConvolve1dx, R_DRW2D_DrawRectConvolve1dy, R_DRW2D_DrawRectConvolve2d, R_DRW2D_DrawTriangles, R_DRW2D_DrawTrianglesUV, R_DRW2D_DrawPolyline, R_DRW2D_DrawLines, R_DRW2D_DrawEllipse, R_DRW2D_FramebufferClear, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_ConvolutionKernelPreset2d_t

4.2.11.7 R_DRW2D_GetGaussKernel

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_GetGaussKernel(r_drw2d_Device_t    Device,
                                       r_drw2d_FixedP_t   *Kernel,
                                       int32_t             Width,
                                       int32_t             Height,
                                       r_drw2d_FixedP_t    Sigma)
```

Parameter

Table 4-74 Parameter of R_DRW2D_GetGaussKernel

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Kernel	Pointer to an array that has a size of (Width*Height).
Width	Width of the kernel (needs to be an odd value!).
Height	Height of the kernel (needs to be an odd value!).
Sigma	The sigma used to compute the gauss coefficients.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK	- No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE	- Invalid device handle.
R_DRW2D_ERR_CONVOLUTION_ADDR	- Invalid coefficient array address.
R_DRW2D_ERR_CONVOLUTION_DIMENSION	- Invalid kernel dimensions.

Description

Computes a gauss kernel with the given size and sigma.
For the setting range of parameter, see ***Table 6-2***.

See also

R_DRW2D_CtxConvolutionKernel, R_DRW2D_DrawRectConvolve1dx,R_DRW2D_DrawRectConvolve1dy, R_DRW2D_DrawRectConvolve2d, R_DRW2D_DrawRectConvolve, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_FixedP_t, int32_t

4.2.11.8 R_DRW2D_CtxConvolutionKernel

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxConvolutionKernel(
                                                r_drw2d_Device_t           Device,
                                                const r_drw2d_ConvKernel_t *Kernel)
```

Parameter

Table 4-75 Parameter of R_DRW2D_CtxConvolutionKernel

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Kernel	Convolution kernel (see r_drw2d_ConvKernel_t). - kernel width and height needs to be an odd value

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK	- No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE	- Invalid device handle.
R_DRW2D_ERR_CONVOLUTION_ADDR	- Invalid coefficient array address.
R_DRW2D_ERR_CONVOLUTION_DIMENSION	- Invalid kernel dimensions.
R_DRW2D_ERR_INVALID_VALUE	- Parameter/argument value is out of range or undefined.

Description

Select 2d convolution kernel size and weights.
For the setting range of parameter, see ***Table 6-2***.

See also

R_DRW2D_CtxConvolutionKernelPreset1d, R_DRW2D_DrawRectConvolve1dx, R_DRW2D_DrawRectConvolve1dy, R_DRW2D_DrawRectConvolve2d, R_DRW2D_DrawTriangles, R_DRW2D_DrawTrianglesUV, R_DRW2D_DrawPolyline, R_DRW2D_DrawLines, R_DRW2D_DrawEllipse, R_DRW2D_FramebufferClear, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_ConvKernel_t

4.2.11.9 R_DRW2D_CtxConvolutionMode

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_CtxConvolutionMode(r_drw2d_Device_t     Device,
                                           r_drw2d_ConvMode_t   Mode)
```

Parameter

Table 4-76 Parameter of R_DRW2D_CtxConvolutionMode

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Mode	Convolution mode (see r_drw2d_ConvMode_t).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.
R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.

Description

Sets the convolution mode.

See also

R_DRW2D_CtxConvolutionKernelPreset1d, R_DRW2D_DrawRectConvolve1dx, R_DRW2D_DrawRectConvolve1dy, R_DRW2D_DrawRectConvolve2d, R_DRW2D_DrawTriangles, R_DRW2D_DrawTrianglesUV, R_DRW2D_DrawPolyline, R_DRW2D_DrawLines, R_DRW2D_DrawEllipse, R_DRW2D_FramebufferClear, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_ConvMode_t

4.2.12 Display list control functions

4.2.12.1 R_DRW2D_GpuFinish

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_GpuFinish(r_drw2d_Device_t    Device,
                                  r_drw2d_Finish_t    Block)
```

Parameter

Table 4-77 Parameter of R_DRW2D_GpuFinish

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Block	Specify to wait for execution or not (see r_drw2d_Finish_t).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK	- No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE	- Invalid device handle.
R_DRW2D_ERR_DEVICE_OUTOFVIDMEM	- Failed to allocate video memory.
R_DRW2D_ERR_DEVICE_INTERNAL_FINISH	- Internal device driver error (during finish).
R_DRW2D_ERR_DEVICE_INTERNAL_FLUSH	- Internal device driver error (during flush).
R_DRW2D_ERR_INVALID_VALUE_GPUFINISH	- Invalid finish type.

Description

Tell the driver to explicitly trigger the finishing of the current drawing scene operation (display list execution). This function can block. i.e. wait for all commands in the GPU to be processed if the *Block* parameter is R_DRW2D_FINISH_WAIT or return immediately if the parameter is R_DRW2D_FINISH_NOWAIT.

Alternatively, a non-blocking approach can be used: Call once R_DRW2D_FINISH_NOWAIT_MARK and later query with R_DRW2D_GpuFinished.
Use of R_DRW2D_FINISH_NOWAIT_MARK and confirmation of *RetFinshed* = R_TRUE by R_DRW2D_GpuFinished are one to one relationship. If it doesn't confirm completion for the previous R_DRW2D_FINISH_NOWAIT_MARK, calling this function with R_DRW2D_FINISH_NOWAIT_MARK has no effect.

See also

R_DRW2D_GpuFinished, R_DRW2D_GpuCmdListCreate, R_DRW2D_GpuCmdListGenerate, R_DRW2D_GpuCmdListExec, R_DRW2D_GpuCmdListCopy, R_DRW2D_GpuCmdListDelete, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_Finish_t

4.2.12.2 R_DRW2D_GpuFinished

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_GpuFinished(r_drw2d_Device_t       Device,
                                    r_drw2d_Boolean_t     *RetFinished)
```

Parameter

Table 4-78 Parameter of R_DRW2D_GpuFinished

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
RetFinished	Result whether the GPU is finished R_TRUE : already finished R_FALSE: not finished

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK	- No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE	- Invalid device handle.
R_DRW2D_ERR_INVALID_VALUE_GPUFINISH	- Invalid finish type.

Description

Queries the driver for a yes/no whether there are still pending jobs in its pipeline.
Can be used for single-threaded non-blocking use cases. R_DRW2D_GpuFinish with R_DRW2D_FINISH_NOWAIT_MARK must be called once before this function.
RetFinished = R_TRUE can only be checked once per R_DRW2D_FINISH_NOWAIT_MARK.

See also

R_DRW2D_GpuFinish, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_Boolean_t

4.2.12.3 R_DRW2D_GpuCmdListCreate

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_GpuCmdListCreate(r_drw2d_Device_t        Device,
                                         r_drw2d_GpuCmdList_t   *RetCmdList)
```

Parameter

Table 4-79 Parameter of R_DRW2D_GpuCmdListCreate

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
RetCmdList	Receives the allocated command list address (see r_drw2d_GpuCmdList_t).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_DEVICE_OUTOFVIDMEM - Failed to allocate video memory.
R_DRW2D_ERR_DEVICE_INTERNAL_CMDLIST - Internal device driver error (during cmdlist create/..).
R_DRW2D_ERR_COMMANDLIST_RETHANDLE - Invalid command list handle return address.

Description

Allocate empty command list.
The application must call R_DRW2D_GpuCmdListDelete to delete the command list when it is no longer used.

See also

R_DRW2D_GpuFinish, R_DRW2D_GpuCmdListGenerate, R_DRW2D_GpuCmdListExec, R_DRW2D_GpuCmdListCopy, R_DRW2D_GpuCmdListDelete, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_GpuCmdList_t

4.2.12.4 R_DRW2D_GpuCmdListGenerate

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_GpuCmdListGenerate(
                                        r_drw2d_Device_t              Device,
                                        r_drw2d_GpuCmdList_t          CmdList,
                                        r_drw2d_GpuCmdListCallback_t  Cbk,
                                        void                         *UserData)
```

Parameter

Table 4-80 Parameter of R_DRW2D_GpuCmdListGenerate

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
CmdList	Command list handle (see r_drw2d_GpuCmdList_t).
Cbk	Callback function (see r_drw2d_GpuCmdListCallback_t).
UserData	Arbitrary user data which will be passed to the callback function.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_DEVICE_OUTOFVIDMEM - Failed to allocate video memory.
R_DRW2D_ERR_DEVICE_INTERNAL_CMDLIST - Internal device driver error (during cmdlist create/..).
R_DRW2D_ERR_DEVICE_INTERNAL_ALLOCCLUT- Internal device driver error (while handling CLUT memory).
R_DRW2D_ERR_FRAMEBUFFER_HANDLE - Invalid framebuffer handle.
R_DRW2D_ERR_TEXTURE_ADDR - Invalid texture buffer address.
R_DRW2D_ERR_TEXTURE_PIXELFMT - Unsupported texel format.
R_DRW2D_ERR_TEXTURE_RLE_BPP - Bits per texel not suitable for RLE decoder (D/AVE HD specific).
R_DRW2D_ERR_EFFECT_INVALID_TEXTURE - Invalid Texture Index.
R_DRW2D_ERR_BUFFER_PIXELFMT - Invalid/unsupported pixel format.
R_DRW2D_ERR_BUFFER_ALIGNMENT - Buffer alignment not correct (for framebuffer).
R_DRW2D_ERR_INVALID_VALUE_FILLMODE - Invalid fill mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDMODE - Invalid blend mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCRGB_UNSUPPORTED - Unsupported SrcRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTRGB_UNSUPPORTED - Unsupported DstRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCALPHA_UNSUPPORTED - Unsupported SrcAlpha blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTALPHA_UNSUPPORTED - Unsupported DstAlpha blend factor.
R_DRW2D_ERR_COMMANDLIST_HANDLE - Invalid command list handle (not allocated, faulty, can not record, ..).
R_DRW2D_ERR_COMMANDLIST_CBKNULLPTR - Command list callback function null ptr.
R_DRW2D_ERR_EFFECT_INVALID_OPERAND - Invalid Parameters provided for effects.
R_DRW2D_ERR_EFFECT_INVALID_OPERATION - Invalid effect name.
R_DRW2D_ERR_EFFECT_OUT_OF_RESOURCES - Combination of effects cannot be realized.

Description

Record command list by calling an application provided function that invokes render commands.
Any previously recorded command list data will be discarded.
The command list must have been created using R_DRW2D_GpuCmdListCreate.
Please notice that not all API functions may be called in the callback functions. Drawing API functions are supported.

See also

R_DRW2D_GpuFinish, R_DRW2D_GpuCmdListCreate, R_DRW2D_GpuCmdListExec, R_DRW2D_GpuCmdListCopy, R_DRW2D_GpuCmdListDelete, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_GpuCmdList_t, r_drw2d_GpuCmdListCallback_t

4.2.12.5 R_DRW2D_GpuCmdListExec

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_GpuCmdListExec(r_drw2d_Device_t         Device,
                                       r_drw2d_GpuCmdList_t     CmdList)
```

Parameter

Table 4-81 Parameter of R_DRW2D_GpuCmdListExec

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
CmdList	Command list handle (see r_drw2d_GpuCmdList_t).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_DEVICE_OUTOFVIDMEM - Failed to allocate video memory.
R_DRW2D_ERR_DEVICE_INTERNAL_CMDLIST - Internal device driver error (during cmdlist create/..).
R_DRW2D_ERR_DEVICE_INTERNAL_ALLOCCLUT- Internal device driver error (while handling CLUT memory).
R_DRW2D_ERR_FRAMEBUFFER_HANDLE - Invalid framebuffer handle.
R_DRW2D_ERR_TEXTURE_ADDR - Invalid texture buffer address.
R_DRW2D_ERR_TEXTURE_PIXELFMT - Unsupported texel format.
R_DRW2D_ERR_TEXTURE_RLE_BPP - Bits per texel not suitable for RLE decoder (D/AVE HD specific).
R_DRW2D_ERR_EFFECT_INVALID_TEXTURE - Invalid Texture Index.
R_DRW2D_ERR_BUFFER_PIXELFMT - Invalid/unsupported pixel format.
R_DRW2D_ERR_BUFFER_ALIGNMENT - Buffer alignment not correct (for framebuffer).
R_DRW2D_ERR_INVALID_VALUE_FILLMODE - Invalid fill mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDMODE - Invalid blend mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCRGB_UNSUPPORTED - Unsupported SrcRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTRGB_UNSUPPORTED - Unsupported DstRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCALPHA_UNSUPPORTED - Unsupported SrcAlpha blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTALPHA_UNSUPPORTED - Unsupported DstAlpha blend factor.
R_DRW2D_ERR_COMMANDLIST_HANDLE - Invalid command list handle (not allocated, faulty, can not record, ..).
R_DRW2D_ERR_EFFECT_INVALID_OPERAND - Invalid Parameters provided for effects.
R_DRW2D_ERR_EFFECT_INVALID_OPERATION - Invalid effect name.
R_DRW2D_ERR_EFFECT_OUT_OF_RESOURCES - Combination of effects cannot be realized.

Description

Execute previously recorded command list.

See also

R_DRW2D_GpuFinish, R_DRW2D_GpuCmdListCreate, R_DRW2D_GpuCmdListGenerate, R_DRW2D_GpuCmdListCopy, R_DRW2D_GpuCmdListDelete, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_GpuCmdList_t

4.2.12.6 R_DRW2D_GpuCmdListCopy

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_GpuCmdListCopy(r_drw2d_Device_t       Device,
                                       r_drw2d_GpuCmdList_t  CmdList,
                                       void                 *DestAddr,
                                       uint32_t             *Size,
                                       void                 *RelocBaseAddr)
```

Parameter

Table 4-82 Parameter of R_DRW2D_GpuCmdListCopy

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
CmdList	Command list handle (see r_drw2d_GpuCmdList_t).
DestAddr	Where to copy the command list. NULL to query required size.
Size	If DestAddr is null, returns the required size. Otherwise this parameter determines the maximum number of bytes that DestAddr can hold.
RelocBaseAddr	If not null, specifies the start address from where the command list can be executed later on (e.g. a flash memory address).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK	- No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE	- Invalid device handle.
R_DRW2D_ERR_DEVICE_OUTOFVIDMEM	- Failed to allocate video memory.
R_DRW2D_ERR_INVALID_VALUE_NULLPTR	- Parameter pointer argument is NULL.
R_DRW2D_ERR_COMMANDLIST_HANDLE	- Invalid command list handle (not allocated, faulty, can not record, ..).
R_DRW2D_ERR_DEVICE_INTERNAL_CMDLIST	- Internal device driver error (during cmdlist create/..).

Description

Copy command list data to memory area.
If DestAddr is NULL, Size returns the required size (in bytes) and no command list data is copied.
If RelocBaseAddr is != NULL, relocate jump commands so that the command list can later be executed from the given address (e.g. in flash memory).
The copied command list may not be re-recorded or deleted using R_DRW2D_GpuCmdListDelete.

See also

R_DRW2D_GpuFinish, R_DRW2D_GpuCmdListCreate, R_DRW2D_GpuCmdListGenerate, R_DRW2D_GpuCmdListExec, R_DRW2D_GpuCmdListDelete, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_GpuCmdList_t, uint32_t

4.2.12.7 R_DRW2D_GpuCmdListDelete

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_GpuCmdListDelete(r_drw2d_Device_t       Device,
                                         r_drw2d_GpuCmdList_t  CmdList)
```

Parameter

Table 4-83 Parameter of R_DRW2D_GpuCmdListDelete

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
CmdList	Command list handle (see r_drw2d_GpuCmdList_t).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK	- No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE	- Invalid device handle.
R_DRW2D_ERR_DEVICE_OUTOFVIDMEM	- Failed to allocate video memory.
R_DRW2D_ERR_DEVICE_INTERNAL_CMDLIST	- Internal device driver error (during cmdlist create/..).
R_DRW2D_ERR_COMMANDLIST_HANDLE	- Invalid command list handle (not allocated, faulty, can not record, ..).

Description

Delete command list. The command list must have been created using R_DRW2D_GpuCmdListCreate.

See also

R_DRW2D_GpuFinish, R_DRW2D_GpuCmdListCreate, R_DRW2D_GpuCmdListGenerate, R_DRW2D_GpuCmdListExec, R_DRW2D_GpuCmdListCopy, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_GpuCmdList_t

4.2.13 Performance counter functions

4.2.13.1 R_DRW2D_PerfCountersAlloc

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_PerfCountersAlloc(r_drw2d_Device_t   Device)
```

Parameter

Table 4-84 Parameter of R_DRW2D_PerfCountersAlloc

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_SYS_MUTEX_LOCK - Failed to lock mutex.
R_DRW2D_ERR_SYS_MUTEX_UNLOCK - Failed to unlock mutex.
R_DRW2D_ERR_PERF_ALLOC - Failed to allocate performance counters.

Description

Allocate hardware performance counter resources for this device context.

See also

r_drw2d_Performance_t, R_DRW2D_PerfCountersFree, R_DRW2D_PerfValueGet, R_DRW2D_PerfValueReset, r_drw2d_Error_t, r_drw2d_Device_t

4.2.13.2 R_DRW2D_PerfCountersFree

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_PerfCountersFree(r_drw2d_Device_t    Device)
```

Parameter

Table 4-85 Parameter of R_DRW2D_PerfCountersFree

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK	- No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE	- Invalid device handle.
R_DRW2D_ERR_SYS_MUTEX_LOCK	- Failed to lock mutex.
R_DRW2D_ERR_SYS_MUTEX_UNLOCK	- Failed to unlock mutex.
R_DRW2D_ERR_PERF_FREE	- Failed to free performance counters.

Description

Free hardware performance counter resources for this device context.

See also

R_DRW2D_PerfCountersAlloc, R_DRW2D_PerfValueGet, R_DRW2D_PerfValueReset, r_drw2d_Error_t, r_drw2d_Device_t

4.2.13.3 R_DRW2D_PerfValueGet

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_PerfValueGet(r_drw2d_Device_t         Device,
                                     r_drw2d_Performance_t    Type,
                                     uint32_t                *RetValue)
```

Parameter

Table 4-86 Parameter of R_DRW2D_PerfValueGet

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Type	Performance type to query (see r_drw2d_Performance_t).
RetValue	Cycle count is stored here (must not be NULL).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK	- No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE	- Invalid device handle.
R_DRW2D_ERR_INVALID_VALUE_PERF_TYPE	- Invalid performance counter type.
R_DRW2D_ERR_INVALID_VALUE_NULLPTR	- Parameter pointer argument is NULL.
R_DRW2D_ERR_PERF_READ	- Failed to query performance counter value.

Description

Query the driver for HW cycles specified by Type and return the value in RetValue.
The cycle counts are reset when R_DRW2D_PerfValueReset is called.
R_DRW2D_PerfCountersAlloc must have been called to allocate the hardware performance counters for the Drw2D device context.

See also

R_DRW2D_PerfValueReset, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_Performance_t, uint32_t

4.2.13.4 R_DRW2D_PerfValueReset

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_PerfValueReset(r_drw2d_Device_t         Device,
                                       r_drw2d_Performance_t    Type)
```

Parameter

Table 4-87 Parameter of R_DRW2D_PerfValueReset

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Type	Performance type to query (see r_drw2d_Performance_t).

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.

R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.
R_DRW2D_ERR_INVALID_VALUE_PERF_TYPE - Invalid performance counter type.
R_DRW2D_ERR_PERF_RESET - Failed to reset performance counter.

Description

Reset the HW cycles of the given performance type to 0.

See also

R_DRW2D_PerfValueGet, r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_Performance_t

4.2.14 Error handling functions

If the driver detects any fatal error, it will call the driver internal error handler function set by R_DRW2D_ErrCallbackSet.
The error handler itself is not part of the API and must not be called by an application.

4.2.14.1 R_DRW2D_ErrCallbackSet

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_ErrCallbackSet(r_drw2d_Device_t          Device,
                                       r_drw2d_ErrorCallback_t   ErrorCb,
                                       void                     *UserData)
```

Parameter

Table 4-88 Parameter of R_DRW2D_ErrCallbackSet

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
ErrorCb	Reference to error handler callback function (see r_drw2d_ErrorCallback_t).
UserData	Arbitrary user data that is passed on to the error callback function.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.
R_DRW2D_ERR_OK - No error occurred.
R_DRW2D_ERR_DEVICE_HANDLE - Invalid device handle.

Description

Install a device context / thread specific application error handler for the driver.
If ErrorCb is zero, no callback function will be used.
If the application has set an error handler callback function, the central error handler will call it and then return to its caller.
To uninstall a device context / thread specific error handler, ErrorCb should be set to NULL.

See also

r_drw2d_Error_t, r_drw2d_Device_t, r_drw2d_ErrorCallback_t

4.2.14.2 R_DRW2D_GlobalErrCallbackSet

Function Prototypes

```
r_drw2d_Error_t R_DRW2D_GlobalErrCallbackSet(
                                r_drw2d_GlobalErrorCallback_t  GlobalErrorCb,
                                void                           *UserData)
```

Parameter

Table 4-89 Parameter of R_DRW2D_GlobalErrCallbackSet

Parameter	Description
GlobalErrorCb	Reference to error handler callback function (see r_drw2d_GlobalErrorCallback_t).
UserData	Arbitrary user data that is passed on to the error callback function.

Return Codes

Error code. See r_drw2d_Error_t for the list of error codes.
R_DRW2D_ERR_OK - No error occurred.

Description

Install a global error handler for the driver.
If GlobalErrorCb is zero, no callback function will be used.
If a valid device context (r_drw2d_Device_t) is available when an error occurs, and the application has set an error callback for that device context, the device context error handler will have precedence over the global error handler.
If no valid device context is available, only the global error handler will be called.
If the application has set an error handler callback function, the central error handler will call it and then return to its caller.
To uninstall the global error handler, GlobalErrorCb should be set to NULL.

See also

r_drw2d_Error_t, r_drw2d_GlobalErrorCallback_t

5. Types

5.1 Basic type

This section shows the basic type used on this library.

Table 5-1 Basic type

Types	Definition		Basic types
char_t	typedef char	char_t;	signed char
int8_t	typedef signed char	int8_t	signed char
int16_t	typedef signed short	int16_t	signed short
int32_t	typedef signed int	int32_t	signed int
int64_t	typedef signed long long	int64_t	signed long long
uint8_t	typedef unsigned char	uint8_t	unsigned char
uint16_t	typedef unsigned short	uint16_t	unsigned short
uint32_t	typedef unsigned int	uint32_t	unsigned int
uint64_t	typedef unsigned long long	uint64_t	unsigned long long
float32_t	typedef float	float32_t	float
float64_t	typedef double	float64_t	double

5.2 Drw2D API Type

This section shows the Drw2D API function type used on this library.

Table 5-2 Drw2D API Type

Types	Definition	Description
r_drw2d_Device_t	typedef void *r_drw2d_Device_t	Device handle. See also R_DRW2D_Init
r_drw2d_Context_t	typedef void *r_drw2d_Context_t	Opaque render context handle.
r_drw2d_FixedP_t	typedef int32_t r_drw2d_FixedP_t	A 1:15:16 fixed point number (signed 2s complement format. MSB=sign bit,15 integer bits, 16 fractional bits).
r_drw2d_Boolean_t	typedef int32_t r_drw2d_Boolean_t	A Boolean (R_TRUE or R_FALSE, 1 or 0).
r_drw2d_Unit_t	typedef uint32_t r_drw2d_Unit_t;	Device enumeration type.
r_drw2d_Color_t	typedef uint32_t r_drw2d_Color_t	Packed 32bit ARGB color.

5.2.1 r_drw2d_ErrorCallback_t

Description

Device-context specific error callback function type.
This callback function is used when a valid Device handle is available.
Therefore, a different callback function can be set for each thread that uses the Drw2D API.
Return R_TRUE if the error was handled.

Definition

```
typedef r_drw2d_Boolean_t (*r_drw2d_ErrorCallback_t)(
                                          r_drw2d_Device_t     Device,
                                          r_drw2d_Error_t      Error,
                                          void                *UserData);
```

Table 5-3 Parameter of r_drw2d_ErrorCallback_t type

Parameter	Description
Device	Device handle (see r_drw2d_Device_t).
Error	Error information (see r_drw2d_Error_t).
UserData	Arbitrary user data that is passed on to the error callback function.

See also

r_drw2d_Error_t, R_DRW2D_GlobalErrCallbackSet

5.2.2 r_drw2d_GlobalErrorCallback_t

Description

Global error callback function type.
This callback function is used when no Device handle is available
(e.g. when the application passed a NULL pointer handle).
The callback is shared by all threads that use the Drw2D API.
Return R_TRUE if the error was handled.

Definition

```
typedef r_drw2d_Boolean_t (*r_drw2d_GlobalErrorCallback_t)(
                                                r_drw2d_Error_t      Error,
                                                void                *UserData);
```

Table 5-4 Parameter of r_drw2d_GlobalErrorCallback_t type

Parameter	Description
Error	Error information (see r_drw2d_Error_t).
UserData	Arbitrary user data that is passed on to the error callback function.

See also

r_drw2d_Error_t, R_DRW2D_GlobalErrCallbackSet

5.2.3 r_drw2d_GpuCmdList_t

Description

GPU command list handle/address.

Definition

```
typedef void *r_drw2d_GpuCmdList_t;
```

See also

R_DRW2D_GpuCmdListCreate,R_DRW2D_GpuCmdListGenerate, R_DRW2D_GpuCmdListExec, R_DRW2D_GpuCmdListCopy

5.2.4 r_drw2d_GpuCmdListCallback_t

Description

GPU command list callback used by R_DRW2D_GpuCmdListGenerate.

Definition

```
typedef r_drw2d_Error_t (*r_drw2d_GpuCmdListCallback_t) (void *UserData);
```

Table 5-5 Parameter of r_drw2d_GpuCmdListCallback_t type

Parameter	Description
UserData	Arbitrary user data that is set to R_DRW2D_GpuCmdListGenerate.

See also

R_DRW2D_GpuCmdListCreate,R_DRW2D_GpuCmdListGenerate, R_DRW2D_GpuCmdListExec, R_DRW2D_GpuCmdListCopy

5.2.5 r_drw2d_EdgeFlag_t

Description

Specifies which edge(s) of a triangle or quad will be antialiased.

Definition

```
typedef uint32_t r_drw2d_EdgeFlag_t;
#define R_DRW2D_EDGE_AB    ((r_drw2d_EdgeFlag_t)((uint8_t)(((uint8_t)1u) << ((uint8_t)0))))
#define R_DRW2D_EDGE_BC    ((r_drw2d_EdgeFlag_t)((uint8_t)(((uint8_t)1u) << ((uint8_t)1))))
#define R_DRW2D_EDGE_CA    ((r_drw2d_EdgeFlag_t)((uint8_t)(((uint8_t)1u) << ((uint8_t)2))))
#define R_DRW2D_EDGE_CD    ((r_drw2d_EdgeFlag_t)((uint8_t)(((uint8_t)1u) << ((uint8_t)3))))
#define R_DRW2D_EDGE_DA    ((r_drw2d_EdgeFlag_t)((uint8_t)(((uint8_t)1u) << ((uint8_t)4))))
```

Table 5-6 Value of r_drw2d_EdgeFlag_t type

Value	Description
R_DRW2D_EDGE_AB	Antialias edge between first and second vertex.
R_DRW2D_EDGE_BC	Antialias edge between second and third vertex.
R_DRW2D_EDGE_CA	Antialias edge between third and first triangle vertex (triangle only).
R_DRW2D_EDGE_CD	Antialias edge between third and fourth vertex (quad only).
R_DRW2D_EDGE_DA	Antialias edge between fourth and first vertex (quad only).

See also

none

5.3 Definition

This section shows the definition value used in Drw2D API.

Table 5-7 Definition of Drw2D API

Name	Values	Description
R_DRW2D_VERSION_MAJOR	*	Major version number. This value is changed with release version.
R_DRW2D_VERSION_MINOR	*	Minor version number. This value is changed with release version.
R_DRW2D_VERSION_PATCH	*	Patch level. This value is changed with release version.

5.4 Enumerated type

This section shows the enumerated value used in Drw2D API Function.

5.4.1 r_drw2d_Error_t

Description

Return codes used in almost all API functions.
The lower 16 bits of an error code are used to encode detailed, possibly device-specific, error information.
The R_DRW2D_ERROR_CLASS macro can be used to mask out the detailed error sub-code.

Definition

```
typedef enum
{
  R_DRW2D_ERR_OK                                          = 0,
  R_DRW2D_ERR_SYS                                         = 0x00010000,
  R_DRW2D_ERR_SYS_MUTEX_LOCK                              = 0x00010010,
  R_DRW2D_ERR_SYS_MUTEX_UNLOCK                            = 0x00010011,
  R_DRW2D_ERR_SYS_MUTEX_CREATE                            = 0x00010012,
  R_DRW2D_ERR_SYS_MUTEX_DESTROY                           = 0x00010013,
  R_DRW2D_ERR_NOT_SUPPORTED                               = 0x00020000,
  R_DRW2D_ERR_INVALID_VALUE                               = 0x00030000,
  R_DRW2D_ERR_INVALID_VALUE_NULLPTR                       = 0x00030010,
  R_DRW2D_ERR_INVALID_VALUE_FILLMODE                      = 0x00030020,
  R_DRW2D_ERR_INVALID_VALUE_CULLMODE                      = 0x00030030,
  R_DRW2D_ERR_INVALID_VALUE_LINEJOIN                      = 0x00030040,
  R_DRW2D_ERR_INVALID_VALUE_LINECAP                       = 0x00030050,
  R_DRW2D_ERR_INVALID_VALUE_LINEWIDTH                     = 0x00030060,
  R_DRW2D_ERR_INVALID_VALUE_MITERLIMIT                    = 0x00030070,
  R_DRW2D_ERR_INVALID_VALUE_IMGQUALITY                    = 0x00030080,
  R_DRW2D_ERR_INVALID_VALUE_BLENDMODE                     = 0x00030090,
  R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCRGB            = 0x000300A1,
  R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTRGB            = 0x000300A2,
  R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCALPHA          = 0x000300A3,
  R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTALPHA          = 0x000300A4,
  R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCRGB_UNSUPPORTED   = 0x000300A5,
  R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTRGB_UNSUPPORTED   = 0x000300A6,
  R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_SRCALPHA_UNSUPPORTED = 0x000300A7,
  R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_DSTALPHA_UNSUPPORTED = 0x000300A8,
  R_DRW2D_ERR_INVALID_VALUE_GPUFINISH                     = 0x000300B0,
  R_DRW2D_ERR_INVALID_VALUE_TRANSFORMMODE                 = 0x000300C0,
  R_DRW2D_ERR_INVALID_VALUE_VIEWPORT_X                    = 0x000300D1,
  R_DRW2D_ERR_INVALID_VALUE_VIEWPORT_Y                    = 0x000300D2,
  R_DRW2D_ERR_INVALID_VALUE_VIEWPORT_W                    = 0x000300D3,
  R_DRW2D_ERR_INVALID_VALUE_VIEWPORT_H                    = 0x000300D4,
```

```
    R_DRW2D_ERR_INVALID_VALUE_DRAWLINES_ODDPOINTCOUNT          = 0x000300E0,
    R_DRW2D_ERR_INVALID_VALUE_PERF_TYPE                        = 0x000300F0,
    R_DRW2D_ERR_INVALID_VALUE_POLYLINE_COUNT                   = 0x00030100,
    R_DRW2D_ERR_INVALID_VALUE_POLYBEZIER_COUNT                 = 0x00030110,
    R_DRW2D_ERR_INVALID_VALUE_CONVKERNELPRESET1D               = 0x00030120,
    R_DRW2D_ERR_INVALID_VALUE_CONVKERNELPRESET2D               = 0x00030130,
    R_DRW2D_ERR_UNIT                                           = 0x00050000,
    R_DRW2D_ERR_UNIT_OUTOFBOUNDS                               = 0x00050010,
    R_DRW2D_ERR_DEVICE                                         = 0x00060000,
    R_DRW2D_ERR_DEVICE_INIT                                    = 0x00060010,
    R_DRW2D_ERR_DEVICE_HANDLE                                  = 0x00060020,
    R_DRW2D_ERR_DEVICE_INTERNAL                                = 0x00060030,
    R_DRW2D_ERR_DEVICE_INTERNAL_SWIZZLEVT                      = 0x00060031,
    R_DRW2D_ERR_DEVICE_INTERNAL_FINISH                         = 0x00060032,
    R_DRW2D_ERR_DEVICE_INTERNAL_SHUTDOWN                       = 0x00060033,
    R_DRW2D_ERR_DEVICE_INTERNAL_ALLOCCLUT                      = 0x00060034,
    R_DRW2D_ERR_DEVICE_INTERNAL_FLUSH                          = 0x00060037,
    R_DRW2D_ERR_DEVICE_INTERNAL_CMDLIST                        = 0x00060038,
    R_DRW2D_ERR_DEVICE_HWINSTANCENR                            = 0x00060100,
    R_DRW2D_ERR_DEVICE_PIXELFMT                                = 0x00060110,
    R_DRW2D_ERR_DEVICE_OUTOFVIDMEM                             = 0x00060120,
    R_DRW2D_ERR_DEVICE_NATIVEDRVHANDLE                         = 0x00060130,
    R_DRW2D_ERR_DEVICE_SAVESTATEALLOC                          = 0x00060140,
    R_DRW2D_ERR_DEVICE_SAVESTATE                               = 0x00060150,
    R_DRW2D_ERR_DEVICE_RESTORESTATE                            = 0x00060160,
    R_DRW2D_ERR_CONTEXT                                        = 0x00070000,
    R_DRW2D_ERR_CONTEXT_NOTINUSE                               = 0x00070010,
    R_DRW2D_ERR_FRAMEBUFFER                                    = 0x00080000,
    R_DRW2D_ERR_FRAMEBUFFER_HANDLE                             = 0x00080010,
    R_DRW2D_ERR_FRAMEBUFFER_ADDR                               = 0x00080020,
    R_DRW2D_ERR_BUFFER                                         = 0x00090000,
    R_DRW2D_ERR_BUFFER_PIXELFMT                                = 0x00090010,
    R_DRW2D_ERR_BUFFER_WIDTH                                   = 0x00090020,
    R_DRW2D_ERR_BUFFER_HEIGHT                                  = 0x00090030,
    R_DRW2D_ERR_BUFFER_ALLINUSE                                = 0x00090040,
    R_DRW2D_ERR_BUFFER_ALIGNMENT                               = 0x00090050,
    R_DRW2D_ERR_TEXTURE                                        = 0x000A0000,
    R_DRW2D_ERR_TEXTURE_ADDR                                   = 0x000A0020,
    R_DRW2D_ERR_TEXTURE_PIXELFMT                               = 0x000A0030,
    R_DRW2D_ERR_TEXTURE_RLE_BPP                                = 0x000A0040,
    R_DRW2D_ERR_TEXTURE_TRANSFORMMODE                          = 0x000A0060,
    R_DRW2D_ERR_TEXTURE_WIDTH                                  = 0x000A0070,
    R_DRW2D_ERR_TEXTURE_HEIGHT                                 = 0x000A0080,
    R_DRW2D_ERR_TEXTURE_UNIT                                   = 0x000A0090,
    R_DRW2D_ERR_DRAWING                                        = 0x000B0000,
    R_DRW2D_ERR_DRAWING_DRAWTRI                                = 0x000B0010,
    R_DRW2D_ERR_DRAWING_DRAWTRIUV                              = 0x000B0020,
    R_DRW2D_ERR_DRAWING_DRAWRECT                               = 0x000B0030,
    R_DRW2D_ERR_DRAWING_DRAWRECTUV                             = 0x000B0040,
    R_DRW2D_ERR_DRAWING_DRAWQUAD                               = 0x000B0050,
    R_DRW2D_ERR_DRAWING_TEXTUREBLIT                            = 0x000B0060,
    R_DRW2D_ERR_COMMANDLIST                                    = 0x000C0000,
    R_DRW2D_ERR_COMMANDLIST_RETHANDLE                          = 0x000C0010,
    R_DRW2D_ERR_COMMANDLIST_HANDLE                             = 0x000C0020,
    R_DRW2D_ERR_COMMANDLIST_CBKNULLPTR                         = 0x000C0030,
    R_DRW2D_ERR_PERF                                           = 0x000D0000,
    R_DRW2D_ERR_PERF_ALLOC                                     = 0x000D0010,
    R_DRW2D_ERR_PERF_FREE                                      = 0x000D0020,
```

```
    R_DRW2D_ERR_PERF_NOTAVAIL                                = 0x000D0030,
    R_DRW2D_ERR_PERF_READ                                    = 0x000D0040,
    R_DRW2D_ERR_PERF_RESET                                   = 0x000D0050,
    R_DRW2D_ERR_EFFECT                                       = 0x000E0000,
    R_DRW2D_ERR_EFFECT_INVALID_OPERAND                       = 0x000E0010,
    R_DRW2D_ERR_EFFECT_OUT_OF_RESOURCES                      = 0x000E0020,
    R_DRW2D_ERR_EFFECT_INVALID_OPERATION                     = 0x000E0030,
    R_DRW2D_ERR_EFFECT_DIV_BY_ZERO                           = 0x000E0040,
    R_DRW2D_ERR_EFFECT_INVALID_TEXTURE                       = 0x000E0050,
    R_DRW2D_ERR_CONVOLUTION                                  = 0x000F0000,
    R_DRW2D_ERR_CONVOLUTION_ADDR                             = 0x000F0010,
    R_DRW2D_ERR_CONVOLUTION_DIMENSION                        = 0x000F0020,
    R_DRW2D_ERR_CONVOLUTION_RES_CONFLICT                     = 0x000F0030,
    R_DRW2D_ERR_CONVOLUTION_INVALID_PARAM                    = 0x000F0040,
    R_DRW2D_NUM_ERR_CODES
  } r_drw2d_Error_t;
```

Table 5-8 Enumerator of r_drw2d_Error_t

Name	Description
R_DRW2D_ERR_OK	No error occurred.
R_DRW2D_ERR_SYS	General system failure.
R_DRW2D_ERR_SYS_MUTEX_LOCK	Failed to lock mutex.
R_DRW2D_ERR_SYS_MUTEX_UNLOCK	Failed to unlock mutex.
R_DRW2D_ERR_SYS_MUTEX_CREATE	Failed to create mutex.
R_DRW2D_ERR_SYS_MUTEX_DESTROY	Failed to destroy mutex.
R_DRW2D_ERR_NOT_SUPPORTED	Parameter/argument value or function not supported.
R_DRW2D_ERR_INVALID_VALUE	Parameter/argument value is out of range or undefined.
R_DRW2D_ERR_INVALID_VALUE_NULLPTR	Parameter pointer argument is NULL.
R_DRW2D_ERR_INVALID_VALUE_FILLMODE	Invalid fill mode.
R_DRW2D_ERR_INVALID_VALUE_CULLMODE	Invalid cull mode.
R_DRW2D_ERR_INVALID_VALUE_LINEJOIN	Invalid LineJoin type.
R_DRW2D_ERR_INVALID_VALUE_LINECAP	Invalid LineCap type.
R_DRW2D_ERR_INVALID_VALUE_LINEWIDTH	Invalid line width (<= 0).
R_DRW2D_ERR_INVALID_VALUE_MITERLIMIT	Invalid miter limit (<= 0).
R_DRW2D_ERR_INVALID_VALUE_IMGQUALITY	Invalid image quality mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDMODE	Invalid blend mode.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_ SRCRGB	Invalid SrcRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_ DSTRGB	Invalid DstRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_ SRCALPHA	Invalid SrcAlpha blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_ DSTALPHA	Invalid DstAlpha blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_ SRCRGB_UNSUPPORTED	Unsupported SrcRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_ DSTRGB_UNSUPPORTED	Unsupported DstRGB blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_ SRCALPHA_UNSUPPORTED	Unsupported SrcAlpha blend factor.
R_DRW2D_ERR_INVALID_VALUE_BLENDFACTOR_ DSTALPHA_UNSUPPORTED	Unsupported DstAlpha blend factor.
R_DRW2D_ERR_INVALID_VALUE_GPUFINISH	Invalid finish type (R_DRW2D_GpuFinish).
R_DRW2D_ERR_INVALID_VALUE_TRANSFORMMODE	Invalid vertex matrix transform mode (R_DRW2D_CtxTransformMode).
R_DRW2D_ERR_INVALID_VALUE_VIEWPORT_X	Invalid viewport Pos.X.
R_DRW2D_ERR_INVALID_VALUE_VIEWPORT_Y	Invalid viewport Pos.Y.
R_DRW2D_ERR_INVALID_VALUE_VIEWPORT_W	Invalid viewport Size.Width.
R_DRW2D_ERR_INVALID_VALUE_VIEWPORT_H	Invalid viewport Size.Height.
R_DRW2D_ERR_INVALID_VALUE_DRAWLINES_ ODDPOINTCOUNT	Odd number of points passed to R_DRW2D_DrawLines.
R_DRW2D_ERR_INVALID_VALUE_PERF_TYPE	Invalid performance counter type.
R_DRW2D_ERR_INVALID_VALUE_POLYLINE_COUNT	Invalid polyline point count (0 or 1).
R_DRW2D_ERR_INVALID_VALUE_POLYBEZIER_COUNT	Invalid bezier curves point count (0, 1 or 2).
R_DRW2D_ERR_INVALID_VALUE_CONVKERNELPRESET1D	Invalid 1D convolution preset
R_DRW2D_ERR_INVALID_VALUE_CONVKERNELPRESET2D	Invalid 2D convolution preset
R_DRW2D_ERR_UNIT	General unit error
R_DRW2D_ERR_UNIT_OUTOFBOUNDS	Invalid unit number
R_DRW2D_ERR_DEVICE	General device error
R_DRW2D_ERR_DEVICE_INIT	Failed to initialize device context

Name	Description
R_DRW2D_ERR_DEVICE_HANDLE	Invalid device handle
R_DRW2D_ERR_DEVICE_INTERNAL Internal	device driver error
R_DRW2D_ERR_DEVICE_INTERNAL_SWIZZLEVT	Internal device driver error (trying to mix swizzling + virtual tiling).
R_DRW2D_ERR_DEVICE_INTERNAL_FINISH	Internal device driver error (during finish).
R_DRW2D_ERR_DEVICE_INTERNAL_SHUTDOWN	Internal device driver error (during shutdown).
R_DRW2D_ERR_DEVICE_INTERNAL_ALLOCCLUT	Internal device driver error (while handling CLUT memory).
R_DRW2D_ERR_DEVICE_INTERNAL_FLUSH	Internal device driver error (during flush).
R_DRW2D_ERR_DEVICE_INTERNAL_CMDLIST	Internal device driver error (during cmdlist create/..).
R_DRW2D_ERR_DEVICE_HWINSTANCENR	Invalid instance (hw unit) nr.
R_DRW2D_ERR_DEVICE_PIXELFMT	Pixel format not supported by device.
R_DRW2D_ERR_DEVICE_OUTOFVIDMEM	Failed to allocate video memory.
R_DRW2D_ERR_DEVICE_NATIVEDRVHANDLE	Failed to query native driver handle.
R_DRW2D_ERR_DEVICE_SAVESTATEALLOC	Failed to create save state.
R_DRW2D_ERR_DEVICE_SAVESTATE	Failed to backup low level driver state.
R_DRW2D_ERR_DEVICE_RESTORESTATE	Failed to restore low level driver state.
R_DRW2D_ERR_CONTEXT	General render context error.
R_DRW2D_ERR_CONTEXT_NOTINUSE	Context not in use (while calling R_DRW2D_ContextSelect, R_DRW2D_ContextInit has to be called first).
R_DRW2D_ERR_FRAMEBUFFER	General framebuffer error.
R_DRW2D_ERR_FRAMEBUFFER_HANDLE	Invalid framebuffer handle.
R_DRW2D_ERR_FRAMEBUFFER_ADDR	Invalid framebuffer address.
R_DRW2D_ERR_BUFFER	General (pixel-) buffer error (framebuffer or texture).
R_DRW2D_ERR_BUFFER_PIXELFMT	Invalid/unsupported pixel format,
R_DRW2D_ERR_BUFFER_WIDTH	Invalid/unsupported width.
R_DRW2D_ERR_BUFFER_HEIGHT	Invalid/unsupported height.
R_DRW2D_ERR_BUFFER_ALLINUSE	All buffers in use.
R_DRW2D_ERR_BUFFER_ALIGNMENT	Buffer alignment not correct (for framebuffer).
R_DRW2D_ERR_TEXTURE	General texture error.
R_DRW2D_ERR_TEXTURE_ADDR	Invalid texture buffer address.
R_DRW2D_ERR_TEXTURE_PIXELFMT	Unsupported texel format.
R_DRW2D_ERR_TEXTURE_RLE_BPP	Bits per texel not suitable for RLE decoder.
R_DRW2D_ERR_TEXTURE_TRANSFORMMODE	Invalid texture matrix transform mode (R_DRW2D_CtxTextureTransformMode).
R_DRW2D_ERR_TEXTURE_WIDTH	Invalid texture width.
R_DRW2D_ERR_TEXTURE_HEIGHT	Invalid texture height.
R_DRW2D_ERR_TEXTURE_UNIT	Invalid texture unit number.
R_DRW2D_ERR_DRAWING	Drawing error.
R_DRW2D_ERR_DRAWING_DRAWTRI	Failed to draw triangle.
R_DRW2D_ERR_DRAWING_DRAWTRIUV	Failed to draw UV texture mapped triangle.
R_DRW2D_ERR_DRAWING_DRAWRECT	Failed to draw rectangle.
R_DRW2D_ERR_DRAWING_DRAWRECTUV	Failed to draw UV texture mapped rectangle.
R_DRW2D_ERR_DRAWING_DRAWQUAD	Failed to draw quad.
R_DRW2D_ERR_COMMANDLIST	Command list error.
R_DRW2D_ERR_COMMANDLIST_RETHANDLE	Invalid command list handle return address.
R_DRW2D_ERR_COMMANDLIST_HANDLE	Invalid command list handle (not allocated, faulty, cannot record, ...).
R_DRW2D_ERR_COMMANDLIST_CBKNULLPTR	Command list callback function null ptr.
R_DRW2D_ERR_PERF	Performance counter error.
R_DRW2D_ERR_PERF_ALLOC	Failed to allocate performance counters.

Name	Description
R_DRW2D_ERR_PERF_FREE	Failed to free performance counters
R_DRW2D_ERR_PERF_NOTAVAIL	Performance counter is not available (due to hw-limit or sw-config).
R_DRW2D_ERR_PERF_READ	Failed to query performance counter value.
R_DRW2D_ERR_PERF_RESET	Failed to reset performance counter.
R_DRW2D_ERR_EFFECT	Effect API error.
R_DRW2D_ERR_EFFECT_INVALID_OPERAND	Invalid Parameters provided for effects.
R_DRW2D_ERR_EFFECT_OUT_OF_RESOURCES	Combination of effects cannot be realized.
R_DRW2D_ERR_EFFECT_INVALID_OPERATION	Invalid Effect Name.
R_DRW2D_ERR_EFFECT_DIV_BY_ZERO	Attempted Division by zero (Two identical points provided to Gradient Effect).
R_DRW2D_ERR_EFFECT_INVALID_TEXTURE	Invalid Texture Index.
R_DRW2D_ERR_CONVOLUTION	Convolution API error.
R_DRW2D_ERR_CONVOLUTION_ADDR	Invalid coefficient array address.
R_DRW2D_ERR_CONVOLUTION_DIMENSION	Invalid kernel dimensions.
R_DRW2D_ERR_CONVOLUTION_RES_CONFLICT	Resource conflict. Color unit already in use by effect.
R_DRW2D_ERR_CONVOLUTION_INVALID_PARAM	Invalid parameter.

See also

none

5.4.2 r_drw2d_PixelFormat_t

Description

Describes the color model and pixel storage format of an r_drw2d_Buffer_t.

Definition

```
typedef enum
{
  R_DRW2D_PIXELFORMAT_NONE      =  0,
  R_DRW2D_PIXELFORMAT_ALPHA     =  1,
  R_DRW2D_PIXELFORMAT_LUM8      =  2,
  R_DRW2D_PIXELFORMAT_AL17      =  3,
  R_DRW2D_PIXELFORMAT_AL44      =  4,
  R_DRW2D_PIXELFORMAT_AL88      =  5,
  R_DRW2D_PIXELFORMAT_AL1       =  6,
  R_DRW2D_PIXELFORMAT_AL2       =  7,
  R_DRW2D_PIXELFORMAT_AL4       =  8,
  R_DRW2D_PIXELFORMAT_AL8       =  9,
  R_DRW2D_PIXELFORMAT_RGB565    = 10,
  R_DRW2D_PIXELFORMAT_ARGB1555  = 11,
  R_DRW2D_PIXELFORMAT_RGBA5551  = 12,
  R_DRW2D_PIXELFORMAT_ARGB4444  = 13,
  R_DRW2D_PIXELFORMAT_RGBA4444  = 14,
  R_DRW2D_PIXELFORMAT_ARGB8888  = 15,
  R_DRW2D_PIXELFORMAT_RGBA8888  = 16,
  R_DRW2D_PIXELFORMAT_CLUT_8    = 17,
  R_DRW2D_PIXELFORMAT_CLUT_4    = 18,
  R_DRW2D_PIXELFORMAT_CLUT_2    = 19,
  R_DRW2D_PIXELFORMAT_CLUT_1    = 20,
  R_DRW2D_NUM_PIXELFORMATS
} r_drw2d_PixelFormat_t;
```

Table 5-9 Enumerator of r_drw2d_PixelFormat_t

Name	Description
R_DRW2D_PIXELFORMAT_NONE	Unspecified format (e.g. unallocated buffers).
R_DRW2D_PIXELFORMAT_ALPHA8	8 alpha bits per pixel (only available as framebuffer format: input is read to all 4 channels, alpha channel is written out).
R_DRW2D_PIXELFORMAT_LUM8	8 luminance bits per pixel (only available as framebuffer format: alpha channel set to 1.0 on read, blue channel is written out).
R_DRW2D_PIXELFORMAT_AL17	1 alpha bit, 7 luminance bits per pixel.
R_DRW2D_PIXELFORMAT_AL44	4 alpha bits, 4 luminance bits per pixel.
R_DRW2D_PIXELFORMAT_AL88	8 alpha bits, 8 luminance bits per pixel.
R_DRW2D_PIXELFORMAT_AL1	1 alpha/luminance bit per pixel (only available as texture format).
R_DRW2D_PIXELFORMAT_AL2	2 alpha/luminance bits per pixel (only available as texture format).
R_DRW2D_PIXELFORMAT_AL4	4 alpha/luminance bits per pixel (only available as texture format).
R_DRW2D_PIXELFORMAT_AL8	8 alpha/luminance bits per pixel (only available as texture format).
R_DRW2D_PIXELFORMAT_RGB565	5 green, 6 red, 5 blue bits per pixel, packed into a 16bit word.
R_DRW2D_PIXELFORMAT_ARGB1555	1 alpha, 5 green, 5 red, 5 blue bits per pixel, packed into a 16bit word (alpha=MSB).
R_DRW2D_PIXELFORMAT_RGBA5551	5 green, 5 red, 5 blue, 1 alpha bits per pixel, packed into a 16bit word (alpha=LSB).
R_DRW2D_PIXELFORMAT_ARGB4444	4 alpha, 4 green, 4 red, 4 blue bits per pixel, packed into a 16bit word (alpha=MSB).
R_DRW2D_PIXELFORMAT_RGBA4444	4 green, 4 red, 4 blue, 4 alpha bits per pixel, packed into a 16bit word (alpha=LSB).
R_DRW2D_PIXELFORMAT_ARGB8888	8 alpha, 8 red, 8 green, 8 blue bits per pixel, packed into a 32bit word (alpha=MSB).
R_DRW2D_PIXELFORMAT_RGBA8888	8 red, 8 green, 8 blue, 8 alpha bits per pixel, packed into a 32bit word (alpha=LSB).
R_DRW2D_PIXELFORMAT_CLUT_8	8 bpp color lookup format (only available as texture format).
R_DRW2D_PIXELFORMAT_CLUT_4	4 bpp color lookup format (only available as texture format).

Name	Description
R_DRW2D_PIXELFORMAT_CLUT_2	2 bpp color lookup format (only available as texture format).
R_DRW2D_PIXELFORMAT_CLUT_1	1 bpp color lookup format (only available as texture format).

See also

r_drw2d_Buffer_t, r_drw2d_Texture_t, R_DRW2D_FramebufferSet, R_DRW2D_CtxTextureSet

5.4.3 r_drw2d_FramebufferFlags_t

Description

Flags used for R_DRW2D_FramebufferSet call.
reserved for future extensions.

Definition

```
typedef enum
{
  __R_DRW2D_FRAMEBUFFERFLAGS_PLACEHOLDER__
} r_drw2d_FramebufferFlags_t;
```

See also

R_DRW2D_FramebufferSet

5.4.4 r_drw2d_TextureFlags_t

Description

Flags used for R_DRW2D_CtxTextureSet call.
The upper 8 bits are reserved for internal purposes (texture lock flags).

Definition

```
typedef uint32_t r_drw2d_TextureFlags_t;
#define R_DRW2D_TEX_NONE          ((r_drw2d_TextureFlags_t)(0u))
#define R_DRW2D_TEX_WRAPU         ((r_drw2d_TextureFlags_t)((uint8_t)(((uint8_t)1u) << ((uint8_t)0))))
#define R_DRW2D_TEX_WRAPV         ((r_drw2d_TextureFlags_t)((uint8_t)(((uint8_t)1u) << ((uint8_t)1))))
#define R_DRW2D_TEX_BILINEAR      ((r_drw2d_TextureFlags_t)((uint8_t)(((uint8_t)1u) << ((uint8_t)2))))
#define R_DRW2D_TEX_PERSPECTIVE   ((r_drw2d_TextureFlags_t)((uint8_t)(((uint8_t)1u) << ((uint8_t)3))))
#define R_DRW2D_TEX_RLE           ((r_drw2d_TextureFlags_t)((uint8_t)(((uint8_t)1u) << ((uint8_t)4))))
#define R_DRW2D_TEX_SWIZZLE       ((r_drw2d_TextureFlags_t)((uint8_t)(((uint8_t)1u) << ((uint8_t)5))))
#define R_DRW2D_TEX_VT            ((r_drw2d_TextureFlags_t)((uint8_t)(((uint8_t)1u) << ((uint8_t)6))))
```

Table 5-10 Enumerator of r_drw2d_TextureFlags_t

Name	Description
R_DRW2D_TEX_NONE	No-op texture flag.
R_DRW2D_TEX_WRAPU	Wrap bitmap on U axis (x-direction).
R_DRW2D_TEX_WRAPV	Wrap bitmap on V axis (y-direction).
R_DRW2D_TEX_BILINEAR	Enable bilinear filtering.
R_DRW2D_TEX_PERSPECTIVE	Enable perspective texture mapping.
R_DRW2D_TEX_RLE	Enable run length encoding (RLE) texture compression.
R_DRW2D_TEX_SWIZZLE	Enable texture swizzling. This flag cannot be combined with R_DRW2D_TEX_VT. The textures' pitch & height have to be aligned to the swizzle mode value (by default it is 4x4 which means the pitch & height need to be aligned to 4).
R_DRW2D_TEX_VT	Enable virtual tiling. This flag cannot be combined with R_DRW2D_TEX_SWIZZLE. Virtual tiling settings is defined in r_config_drw2d.h (R_DRW2D_DHD_VT_BPP_*). This feature requires the texture pitch to be a multiple of the virtual tile width.

See also

R_DRW2D_CtxTextureSet

5.4.5 r_drw2d_BlendMode_t

Description

Enumeration of preset blending equations.

Definition

```
typedef enum
{
  R_DRW2D_BLENDMODE_CUSTOM      =  0,
  R_DRW2D_BLENDMODE_SRC         =  1,
  R_DRW2D_BLENDMODE_SRC_OVER    =  2,
  R_DRW2D_BLENDMODE_DST_OVER    =  3,
  R_DRW2D_BLENDMODE_SRC_IN      =  4,
  R_DRW2D_BLENDMODE_DST_IN      =  5,
  R_DRW2D_BLENDMODE_MULTIPLY    =  6,
  R_DRW2D_BLENDMODE_SCREEN      =  7,
  R_DRW2D_BLENDMODE_DARKEN      =  8,
  R_DRW2D_BLENDMODE_LIGHTEN     =  9,
  R_DRW2D_BLENDMODE_ADDITIVE    = 10,
  R_DRW2D_NUM_BLENDMODES
} r_drw2d_BlendMode_t;
```

Table 5-11 Enumerator of r_drw2d_BlendMode_t

Name	Description
R_DRW2D_BLENDMODE_CUSTOM	User defined blend mode specified by R_DRW2D_CtxBlendFactors. dst_color = src_color * src_factor_color + dst_color * dst_factor_color dst_alpha = src_alpha * src_factor_alpha + dst_alpha* dst_factor_alpha
R_DRW2D_BLENDMODE_SRC	Porter-Duff Src. dst_color = src_color dst_alpha = src_alpha
R_DRW2D_BLENDMODE_SRC_OVER	Porter-Duff Src-over-Dst. st_color = src_color + dst_color* (1-src_alpha) dst_alpha = src_alpha + dst_alpha * (1-src_alpha)
R_DRW2D_BLENDMODE_DST_OVER	Porter-Duff Dst-over-Src. dst_color = src_color * (1-dst_alpha) + dst_color dst_alpha = src_alpha * (1-dst_alpha) + dst_alpha
R_DRW2D_BLENDMODE_SRC_IN	Porter-Duff Src-in-Dst. dst_color = src_color * dst_alpha dst_alpha = src_alpha * dst_alpha
R_DRW2D_BLENDMODE_DST_IN	Porter-Duff Dst-in-Src dst_color = dst_color * src_alpha dst_alpha = dst_alpha * src_alpha
R_DRW2D_BLENDMODE_MULTIPLY	Multiply the source and destination colors together, producing the effect of placing a transparent filter over a background. dst_color = (src_alpha * src_color)*(1-dst_alpha) + (dst_alpha * dst_color) * (1-src_alpha) + (src_alpha * src_color * dst_alpha * dst_color) dst_alpha = src_alpha + dst_alpha * (1-src_alpha)
R_DRW2D_BLENDMODE_SCREEN	The opposite of multiplication, producing the effect of projecting a slide over a background. dst_color = (src_alpha * src_color) + (dst_alpha * dst_color) - (src_alpha * src_color * dst_alpha * dst_color) dst_alpha = src_alpha + dst_alpha * (1-src_alpha)
R_DRW2D_BLENDMODE_DARKEN	Compute (Src over Dst) and (Dst over Src) and take the smaller (darker) value for each channel. dst_color = min(src_alpha * src_color + dst_alpha * dst_color * (1-src_alpha), dst_alpha * dst_color + src_alpha * src_color * (1-dst_alpha)) dst_alpha = src_alpha + dst_alpha * (1-src_alpha)

Name	Description
R_DRW2D_BLENDMODE_LIGHTEN	Compute (Src over Dst) and (Dst over Src) and take the larger (lighter) value for each channel. dst_color = max(src_alpha * src_color + dst_alpha * dst_color * (1-src_alpha), dst_alpha * dst_color + src_alpha * src_color * (1-dst_alpha)) dst_alpha = src_alpha + dst_alpha * (1-src_alpha)
R_DRW2D_BLENDMODE_ADDITIVE	Add alpha and color channels. dst_color = src_alpha * src_color + dst_alpha * dst_color dst_alpha = min(src_alpha+dst_alpha, 1)

See also

R_DRW2D_CtxBlendMode, R_DRW2D_CtxBlendFactors, r_drw2d_BlendFactor_t

5.4.6 r_drw2d_BlendFactor_t

Description

Source/destination blend factors used in R_DRW2D_CtxBlendFactors.
When R_DRW2D_BLENDMODE_CUSTOM is selected, the effective color/alpha blend equation is
dst_color = src_color * src_factor_color + dst_color * dst_factor_color
dst_alpha = src_alpha * src_factor_alpha + dst_alpha * dst_factor_alpha

Definition

```
typedef enum
{
  R_DRW2D_BLEND_ZERO                    =  0,
  R_DRW2D_BLEND_ONE                     =  1,
  R_DRW2D_BLEND_SRC_COLOR               =  2,
  R_DRW2D_BLEND_ONE_MINUS_SRC_COLOR     =  3,
  R_DRW2D_BLEND_DST_COLOR               =  4,
  R_DRW2D_BLEND_ONE_MINUS_DST_COLOR     =  5,
  R_DRW2D_BLEND_SRC_ALPHA               =  6,
  R_DRW2D_BLEND_ONE_MINUS_SRC_ALPHA     =  7,
  R_DRW2D_BLEND_DST_ALPHA               =  8,
  R_DRW2D_BLEND_ONE_MINUS_DST_ALPHA     =  9,
  R_DRW2D_BLEND_CONSTANT_FGCOLOR        = 10,
  R_DRW2D_BLEND_ONE_MINUS_CONST_FGCOLOR = 11,
  R_DRW2D_BLEND_CONSTANT_FGALPHA        = 12,
  R_DRW2D_BLEND_ONE_MINUS_CONST_FGALPHA = 13,
  R_DRW2D_BLEND_CONSTANT_BGCOLOR        = 14,
  R_DRW2D_BLEND_ONE_MINUS_CONST_BGCOLOR = 15,
  R_DRW2D_BLEND_CONSTANT_BGALPHA        = 16,
  R_DRW2D_BLEND_ONE_MINUS_CONST_BGALPHA = 17,
  R_DRW2D_BLEND_SRC_ALPHA_SATURATE      = 18,
  R_DRW2D_BLEND_SRC1_COLOR              = 19,
  R_DRW2D_BLEND_ONE_MINUS_SRC1_COLOR    = 20,
  R_DRW2D_BLEND_SRC1_ALPHA              = 21,
  R_DRW2D_BLEND_ONE_MINUS_SRC1_ALPHA    = 22,
  R_DRW2D_BLEND_SRC1_DST1               = 23
} r_drw2d_BlendFactor_t;
```

Table 5-12 Enumerator of r_drw2d_BlendFactor_t

Name	Description
R_DRW2D_BLEND_ZERO	Multiply by 0 (discard).
R_DRW2D_BLEND_ONE	Multiply by 1.
R_DRW2D_BLEND_SRC_COLOR	Multiply by source color.
R_DRW2D_BLEND_ONE_MINUS_SRC_COLOR	Multiply by inverse source color.
R_DRW2D_BLEND_DST_COLOR	Multiply by destination color.
R_DRW2D_BLEND_ONE_MINUS_DST_COLOR	Multiply by inverse destination color.
R_DRW2D_BLEND_SRC_ALPHA	Multiply by source alpha.
R_DRW2D_BLEND_ONE_MINUS_SRC_ALPHA	Multiply by inverse source alpha.
R_DRW2D_BLEND_DST_ALPHA	Multiply by destination alpha.
R_DRW2D_BLEND_ONE_MINUS_DST_ALPHA	Multiply by inverse destination alpha.
R_DRW2D_BLEND_CONSTANT_FGCOLOR	(not supported)
R_DRW2D_BLEND_ONE_MINUS_CONST_FGCOLOR	(not supported)
R_DRW2D_BLEND_CONSTANT_FGALPHA	(not supported)
R_DRW2D_BLEND_ONE_MINUS_CONST_FGALPHA	(not supported)
R_DRW2D_BLEND_CONSTANT_BGCOLOR	(not supported)
R_DRW2D_BLEND_ONE_MINUS_CONST_BGCOLOR	(not supported)

Name	Description
R_DRW2D_BLEND_CONSTANT_BGALPHA	(not supported)
R_DRW2D_BLEND_ONE_MINUS_CONST_FGALPHA	(not supported)
R_DRW2D_BLEND_SRC_ALPHA_SATURATE	(not supported)
R_DRW2D_BLEND_SRC1_COLOR	(not supported)
R_DRW2D_BLEND_ONE_MINUS_SRC1_COLOR	(not supported)
R_DRW2D_BLEND_SRC1_ALPHA	(not supported)
R_DRW2D_BLEND_ONE_MINUS_SRC1_ALPHA	(not supported)
R_DRW2D_BLEND_SRC1_DST1	(not supported)

See also

R_DRW2D_CtxBlendFactors, R_DRW2D_CtxBlendMode, r_drw2d_BlendMode_t

5.4.7 r_drw2d_FillMode_t

Description

Enumeration of primitive fill modes.
The fill mode influences the drawing mode of all render functions.
The initial fill mode is R_DRW2D_FILLMODE_SOLID.

Definition

```
typedef enum
{
  R_DRW2D_FILLMODE_SOLID     = 1,
  R_DRW2D_FILLMODE_TEXTURE   = 2,
  R_DRW2D_NUM_FILLMODES
} r_drw2d_FillMode_t;
```

Table 5-13 Enumerator of r_drw2d_FillMode_t

Name	Description
R_DRW2D_FILLMODE_SOLID	Fill primitive with foreground color.
R_DRW2D_FILLMODE_TEXTURE	Fill primitive with current texture.

See also

R_DRW2D_CtxFillMode

5.4.8 r_drw2d_CullMode_t

Description

Enumeration of winding order culling modes.
The initial cull mode is R_DRW2D_CULLMODE_NONE.

Definition

```
typedef enum
{
    R_DRW2D_CULLMODE_NONE   = 0,
    R_DRW2D_CULLMODE_CCW    = 1,
    R_DRW2D_CULLMODE_CW     = 2,
    R_DRW2D_NUM_CULLMODES
} r_drw2d_CullMode_t;
```

Table 5-14 Enumerator of r_drw2d_CullMode_t

Name	Description
R_DRW2D_CULLMODE_NONE	Never cull primitives [default].
R_DRW2D_CULLMODE_CCW	Cull primitives that have a counter clock wise winding order.
R_DRW2D_CULLMODE_CW	Cull primitives that have a clock wise winding order.

See also

none

5.4.9 r_drw2d_LineCap_t

Description

Enumeration of line start/end drawing styles.

Definition

```
typedef enum
{
  R_DRW2D_LINECAP_ROUND  = 0,
  R_DRW2D_LINECAP_SQUARE = 1,
  R_DRW2D_LINECAP_BUTT   = 2
} r_drw2d_LineCap_t;
```

Table 5-15 Enumerator of r_drw2d_LineCap_t

Name	Description
R_DRW2D_LINECAP_ROUND	Draw round line endings.
R_DRW2D_LINECAP_SQUARE	Draw flat line ending, offset by half the line width.
R_DRW2D_LINECAP_BUTT	Draw flat line ending.

See also

R_DRW2D_CtxLineStyle,r_drw2d_LineJoin_t,r_drw2d_LineStyle_t,
R_DRW2D_DrawPolyline,R_DRW2D_DrawLines

5.4.10 r_drw2d_LineJoin_t

Description

Enumeration of line connection drawing styles for multi-segment lines (see R_DRW2D_DrawPolyline).

Definition

```
typedef enum
{
  R_DRW2D_LINEJOIN_NONE  = 0,
  R_DRW2D_LINEJOIN_MITER = 1,
  R_DRW2D_LINEJOIN_ROUND = 2,
  R_DRW2D_LINEJOIN_BEVEL = 3,
  R_DRW2D_NUM_LINEJOIN_TYPES
} r_drw2d_LineJoin_t;
```

Table 5-16 Enumerator of r_drw2d_LineJoin_t

Name	Description
R_DRW2D_LINEJOIN_NONE	Do not connect lines (gaps may appear at sharp angles).
R_DRW2D_LINEJOIN_MITER	Connect lines using sharp edges.
R_DRW2D_LINEJOIN_ROUND	Connect lines using round edges.
R_DRW2D_LINEJOIN_BEVEL	Connect lines using flat edges.

See also

R_DRW2D_CtxLineStyle, r_drw2d_LineCap_t, r_drw2d_LineStyle_t, R_DRW2D_DrawPolyline

5.4.11 r_drw2d_ImgQuality_t

Description

Enumeration of anti-aliasing / quality modes.

Definition

```
typedef enum
{
  R_DRW2D_IMGQUALITY_LOW    = 0,
  R_DRW2D_IMGQUALITY_MEDIUM = 1,
  R_DRW2D_IMGQUALITY_HIGH   = 2
} r_drw2d_ImgQuality_t;
```

Table 5-17 Enumerator of r_drw2d_ImgQuality_t

Name	Description
R_DRW2D_IMGQUALITY_LOW	No antialiasing.
R_DRW2D_IMGQUALITY_MEDIUM	Medium quality antialiasing.
R_DRW2D_IMGQUALITY_HIGH	High quality antialiasing (default).

See also

R_DRW2D_CtxImgQuality

5.4.12 r_drw2d_TransformMode_t

Description

Vertex matrix transform mode.

Definition

```
typedef enum
{
  R_DRW2D_TRANSFORM_NONE = 0,
  R_DRW2D_TRANSFORM_2D   = 1,
  R_DRW2D_TRANSFORM_3D   = 2
} r_drw2d_TransformMode_t;
```

Table 5-18 Enumerator of r_drw2d_TransformMode_t

Name	Description
R_DRW2D_TRANSFORM_NONE	No vertex transformation.
R_DRW2D_TRANSFORM_2D	2D vertex transformation (default).
R_DRW2D_TRANSFORM_3D	3D vertex transformation and viewport projection.

See also

R_DRW2D_CtxTransformMode

5.4.13 r_drw2d_TextureTransformMode_t

Description

Texture matrix transform mode.

Definition

```
typedef enum
{
  R_DRW2D_TEX_TRANSFORM_NONE = 0,
  R_DRW2D_TEX_TRANSFORM_2D   = 2,
} r_drw2d_TextureTransformMode_t;
```

Table 5-19 Enumerator of r_drw2d_TextureTransformMode_t

Name	Description
R_DRW2D_TEX_TRANSFORM_NONE	No texture coordinate transformation.
R_DRW2D_TEX_TRANSFORM_2D	2D texture transformation (default) See also: R_DRW2D_CtxTextureTransformMode.

See also

none

5.4.14 r_drw2d_Performance_t

Description

Used in R_DRW2D_PerfValueGet and R_DRW2D_PerfValueReset to query/reset performance values.

Definition

```
typedef enum
{
  R_DRW2D_PERF_GPU_TIME  = 0,
  R_DRW2D_PERF_HW_READS  = 1,
  R_DRW2D_PERF_HW_WRITES = 2
} r_drw2d_Performance_t;
```

Table 5-20 Enumerator of r_drw2d_Performance_t

Name	Description
R_DRW2D_PERF_GPU_TIME	Total GPU time spent.
R_DRW2D_PERF_HW_READS	Total GPU time spent for texture/framebuffer reads.
R_DRW2D_PERF_HW_WRITES	Total GPU time spent for texture/framebuffer writes.

See also

none

5.4.15 r_drw2d_Finish_t

Description

Whether R_DRW2D_GpuFinish will block or not can be specified with this option.

Definition

```
typedef enum
{
  R_DRW2D_FINISH_NOWAIT      = 0,
  R_DRW2D_FINISH_WAIT        = 1,
  R_DRW2D_FINISH_NOWAIT_MARK = 2
} r_drw2d_Finish_t;
```

Table 5-21 Enumerator of r_drw2d_Finish_t

Name	Description
R_DRW2D_FINISH_NOWAIT	Send current command list to GPU and do not wait for completion.
R_DRW2D_FINISH_WAIT	Send current command list to GPU and wait for completion.
R_DRW2D_FINISH_NOWAIT_MARK	Send current command list to GPU non-waiting and mark this list to be queried with R_DRW2D_GpuFinish.

See also

none

5.4.16 r_drw2d_ConvolveMode_t

Description

Convolution filter mode (1d horizontal, 1d vertical, 2d)

Definition

```
typedef enum
{
  R_DRW2D_SYS_CONVOLVEMODE_NONE = 0,
  R_DRW2D_SYS_CONVOLVEMODE_1DX  = 1,
  R_DRW2D_SYS_CONVOLVEMODE_1DY  = 2,
  R_DRW2D_SYS_CONVOLVEMODE_2D   = 3,
  R_DRW2D_SYS_CONVOLVEMODE_USER = 4
} r_drw2d_ConvolveMode_t;
```

Table 5-22 Enumerator of r_drw2d_ConvolveMode_t

Name	Description
R_DRW2D_SYS_CONVOLVEMODE_NONE	No convolution filter mode.
R_DRW2D_SYS_CONVOLVEMODE_1DX	1d horizontal convolution filter.
R_DRW2D_SYS_CONVOLVEMODE_1DY	1d vertical convolution filter.
R_DRW2D_SYS_CONVOLVEMODE_2D	2d convolution filter.
R_DRW2D_SYS_CONVOLVEMODE_USER	user defined convolution filter.

See also

none

5.4.17 r_drw2d_ConvolutionKernelPreset1d_t

Description

1D convolution filter kernel presets.
Also, see R_DRW2D_CtxConvolutionKernelPreset1d,
R_DRW2D_DrawRectConvolve1dx, R_DRW2D_DrawRectConvolve1dy.

Definition

```
typedef enum
{
  R_DRW2D_CONV1D_GAUSSIAN_BLUR_3   = 1,
  R_DRW2D_CONV1D_GAUSSIAN_BLUR_5   = 2,
  R_DRW2D_CONV1D_GAUSSIAN_BLUR_7   = 3,
  R_DRW2D_CONV1D_SOBEL_DIFF        = 4,
  R_DRW2D_CONV1D_SOBEL_AVG         = 5,
  R_DRW2D_CONV1D_NUM_PRESETS       = 6
} r_drw2d_ConvolutionKernelPreset1d_t;
```

Table 5-23 Enumerator of r_drw2d_ConvolutionKernelPreset1d_t

Name	Description
R_DRW2D_CONV1D_GAUSSIAN_BLUR_3	3 coefficient gaussian blur.
R_DRW2D_CONV1D_GAUSSIAN_BLUR_5	5 coefficient gaussian blur.
R_DRW2D_CONV1D_GAUSSIAN_BLUR_7	7 coefficient gaussian blur.
R_DRW2D_CONV1D_SOBEL_DIFF	3x1 Sobel differentiation (edge detection).
R_DRW2D_CONV1D_SOBEL_AVG	3x1 Sobel averaging (edge detection).
R_DRW2D_CONV1D_NUM_PRESETS	Number of 1D convolution filter kernel presets.

See also

none

5.4.18 r_drw2d_ConvolutionKernelPreset2d_t

Description

2D convolution filter kernel presets.
Also see R_DRW2D_CtxConvolutionKernelPreset2d, R_DRW2D_DrawRectConvolve2d.

Definition

```
typedef enum
{
  R_DRW2D_CONV2D_GAUSSIAN_BLUR_3x3 = 1,
  R_DRW2D_CONV2D_GAUSSIAN_BLUR_5x5 = 2,
  R_DRW2D_CONV2D_GAUSSIAN_BLUR_7x7 = 3,
  R_DRW2D_CONV2D_SOBEL_H_3x3       = 4,
  R_DRW2D_CONV2D_SOBEL_V_3x3       = 5,
  R_DRW2D_CONV2D_SHARPEN_3x3       = 6,
  R_DRW2D_CONV2D_EMBOSS_3x3        = 7,
  R_DRW2D_CONV2D_NUM_PRESETS       = 8
} r_drw2d_ConvolutionKernelPreset2d_t;
```

Table 5-24 Enumerator of r_drw2d_ConvolutionKernelPreset2d_t

Name	Description
R_DRW2D_CONV1D_GAUSSIAN_BLUR_3x3	3x3 gaussian blur.
R_DRW2D_CONV1D_GAUSSIAN_BLUR_5x5	5x5 gaussian blur.
R_DRW2D_CONV1D_GAUSSIAN_BLUR_7x7	7x7 gaussian blur.
R_DRW2D_CONV2D_SOBEL_H_3x3	3x3 Sobel horizontal (edge detection).
R_DRW2D_CONV2D_SOBEL_V_3x3	3x3 Sobel vertical (edge detection).
R_DRW2D_CONV2D_SHARPEN_3x3	3x3 sharpen.
R_DRW2D_CONV2D_EMBOSS_3x3	3x3 emboss.
R_DRW2D_CONV2D_NUM_PRESETS	Number of 2D convolution filter kernel presets.

See also

none

5.4.19 r_drw2d_NativeDrvFlags_t

Description

Flags that can be passed to R_DRW2D_NativeDriverBegin.

Definition

```
typedef enum
{
  R_DRW2D_NATIVEDRV_APPCONTEXT  = 0,
  R_DRW2D_NATIVEDRV_SAVESTATE   = 1,
  R_DRW2D_NATIVEDRV_KEEPSTATE   = 2
} r_drw2d_NativeDrvFlags_t;
```

Table 5-25 Enumerator of r_drw2d_NativeDrvFlags_t

Name	Description
R_DRW2D_NATIVEDRV_APPCONTEXT	Application uses a different device context to access the low-level driver. No Drw2D state flush or backup will be done. The device context has to be created/destroyed by the application by calling the respective low level driver functions.
R_DRW2D_NATIVEDRV_SAVESTATE	Application uses the Drw2D device context to access the low-level driver. All pending Drw2D state updates will be sent to the low-level driver and the current driver state is backed up. It will automatically be restored when R_DRW2D_NativeDriverEnd is called.
R__NATIVEDRV_KEEPSTATE	Application uses the Drw2D device context to access the low-level driver. All pending Drw2D state updates will be sent to the low-level driver. The current driver state will not be backed up so that state updates done using low level driver access will potentially have an effect on subsequent Drw2D render calls.

See also

none

5.4.20 r_drw2d_EffectName_t

Description

Available effects for the effect stage. Also, see R_DRW2D_CtxEffectsSet.

Definition

```
typedef enum
{
  R_DRW2D_EFFECT_REPLACE,
  R_DRW2D_EFFECT_MODULATE,
  R_DRW2D_EFFECT_ADD,
  R_DRW2D_EFFECT_ADD_SIGNED,
  R_DRW2D_EFFECT_SUBTRACT,
  R_DRW2D_EFFECT_INTERPOLATE,
  R_DRW2D_EFFECT_DOT3,
  R_DRW2D_EFFECT_CONSTANT_ALPHA,
  R_DRW2D_EFFECT_GRADIENT
} r_drw2d_EffectName_t;
```

Table 5-26 Enumerator of r_drw2d_EffectName_t

Name	Description
R_DRW2D_EFFECT_REPLACE	Replace by color value (single ‘Color’ argument required).
R_DRW2D_EFFECT_MODULATE	Product of two color values (two ‘Color’ arguments required).
R_DRW2D_EFFECT_ADD	Sum of two color values (two ‘Color’ arguments required).
R_DRW2D_EFFECT_ADD_SIGNED	Sum of two color values minus 0.5 (two ‘Color’ arguments required).
R_DRW2D_EFFECT_SUBTRACT	Difference of two color values (two ‘Color’ arguments required).
R_DRW2D_EFFECT_INTERPOLATE	Interpolation of two color values by a third color value (three ‘Color’ arguments required).
R_DRW2D_EFFECT_DOT3	Dot product of two color values. Result is a scalar present in all color channels (two ‘Color’ arguments required).
R_DRW2D_EFFECT_CONSTANT_ALPHA	Blending by constant alpha (single ‘Constant’ argument required); requires R_DRW2D_IMGQUALITY_MEDIUM or R_DRW2D_IMGQUALITY_HIGH.
R_DRW2D_EFFECT_GRADIENT	Blending by alpha gradient (two ‘Point’ and two ‘Constant’ arguments required); requires R_DRW2D_IMGQUALITY_MEDIUM or R_DRW2D_IMGQUALITY_HIGH.

See also

none

5.4.21 r_drw2d_EffectParamSource_t

Description

Specifies where the type of the parameters is coming from.

Definition

```
typedef enum
{
  R_DRW2D_EFFECT_SOURCE_TEXTURE_UNIT,
  R_DRW2D_EFFECT_SOURCE_CONSTANT_COLOR,
  R_DRW2D_EFFECT_SOURCE_CONSTANT,
  R_DRW2D_EFFECT_SOURCE_POINT,
  R_DRW2D_EFFECT_SOURCE_PREV_STAGE
} r_drw2d_EffectParamSource_t;
```

Table 5-27 Enumerator of r_drw2d_EffectParamSource_t

Name	Description
R_DRW2D_EFFECT_SOURCE_TEXTURE_UNIT	Texture unit: Set texture unit index via Param.Color.Source.TextureUnit.
R_DRW2D_EFFECT_SOURCE_CONSTANT_COLOR	32 bit constant color value: Set color value via Param.Color.Source.ConstantColor.
R_DRW2D_EFFECT_SOURCE_CONSTANT	16.16 signed fixpoint value (e.g. for constant alpha effect): Set value via Param.Constant.
R_DRW2D_EFFECT_SOURCE_POINT	2D position (e.g. for gradient effect): Set value via Param.Point.
R_DRW2D_EFFECT_SOURCE_PREV_STAGE	The result of previous effects stage: Nothing in ‘Param’ is to be set.

See also

none

5.4.22 r_drw2d_EffectColorParamOperand_t

Description

Specifies how the color source parameters shall be accessed.

Definition

```
typedef enum
{
  R_DRW2D_EFFECT_COLOR_OPERAND_RGBA,
  R_DRW2D_EFFECT_COLOR_OPERAND_ONE_MINUS_RGBA,
  R_DRW2D_EFFECT_COLOR_OPERAND_ALPHA,
  R_DRW2D_EFFECT_COLOR_OPERAND_ONE_MINUS_ALPHA,
  R_DRW2D_EFFECT_COLOR_OPERAND_111A,
} r_drw2d_EffectColorParamOperand_t;
```

Table 5-28 Enumerator of r_drw2d_EffectColorParamOperand_t

Name	Description
R_DRW2D_EFFECT_COLOR_OPERAND_RGBA	Uses color channels directly.
R_DRW2D_EFFECT_COLOR_OPERAND_ONE_MINUS_RGBA	Uses inverted color channels.
R_DRW2D_EFFECT_COLOR_OPERAND_ALPHA	Uses alpha value for all color channels.
R_DRW2D_EFFECT_COLOR_OPERAND_ONE_MINUS_ALPHA	Uses inverted alpha value for all color channels.
R_DRW2D_EFFECT_COLOR_OPERAND_111A	Uses Alpha value. Color channels is set to 1.

See also

none

5.4.23 r_drw2d_ConvKernelColorChannel_t

Description

Specifies the channels which are being processed by a convolution kernel.

Definition

```
typedef enum
{
    R_DRW2D_CONVKERNEL_COLOR_CHANNEL_RGBA,
    R_DRW2D_CONVKERNEL_COLOR_CHANNEL_RGB,
} r_drw2d_ConvKernelColorChannel_t;
```

Table 5-29 Enumerator of r_drw2d_ConvKernelColorChannel_t

Name	Description
R_DRW2D_CONVKERNEL_COLOR_CHANNEL_RGBA	All color and alpha channels are being processed by the kernel.
R_DRW2D_CONVKERNEL_COLOR_CHANNEL_RGB	Only RGB channels are being processed by the kernel.

See also

none

5.4.24 r_drw2d_ConvMode_t

Description

Specifies how the texture is convoluted.

Definition

```
typedef enum
{
    R_DRW2D_CONVMODE_TRIMMED,
    R_DRW2D_CONVMODE_BLEEDING,
} r_drw2d_ConvMode_t;
```

Table 5-30 Enumerator of r_drw2d_ConvMode_t

Name	Description
R_DRW2D_CONVMODE_TRIMMED	Convolution is trimmed to the texture's size.
R_DRW2D_CONVMODE_BLEEDING	Convolution is also applied to pixels outside of the texture, which leads to a "bleeding" effect when using a blur kernel. The number of pixels affected outside depends on the used kernel's size.

See also

none

5.5 Structure

This section shows the structure used in Drw2D API function.

5.5.1 r_drw2d_LineStyle_t

Description

Describes the drawing style for (poly-)lines.

Definition

```
typedef struct
{
  r_drw2d_LineJoin_t  LineJoin;
  r_drw2d_LineCap_t   LineCap;
  r_drw2d_FixedP_t    Width;
  r_drw2d_FixedP_t    MiterLimit;
  r_drw2d_Boolean_t   IsClosed;
} r_drw2d_LineStyle_t;
```

Table 5-31 Member of r_drw2d_LineStyle_t structure

Member	Description
LineJoin	Line join style. One of R_DRW2D_LINEJOIN_NONE, R_DRW2D_LINEJOIN_MITER, R_DRW2D_LINEJOIN_BEVEL, R_DRW2D_LINEJOIN_ROUND.
LineCap	Line cap style. One of R_DRW2D_LINECAP_ROUND, R_DRW2D_LINECAP_SQUARE, R_DRW2D_LINECAP_BUTT.
Width	Line width.
MiterLimit	Maximum distance between miter line join tip and line point. Value must be greater than zero. If the miter limit is exceeded, a bevel joint will be drawn at the miter limit position.
IsClosed	If true, draw closed polyline (last vertex is connected to first one).

See also

R_DRW2D_CtxLineStyle,r_drw2d_LineCap_t,r_drw2d_LineJoin_t,
R_DRW2D_DrawPolyline,R_DRW2D_DrawLines

5.5.2 r_drw2d_Point_t

Description

A 2D point / vertex (fixed point).
The vertex matrix (see R_DRW2D_CtxTransform) can be used to setup custom coordinate systems.

Definition

```
typedef struct
{
  r_drw2d_FixedP_t X;
  r_drw2d_FixedP_t Y;
} r_drw2d_Point_t;
```

Table 5-32 Member of r_drw2d_Point_t structure

Member	Description
X	Horizontal position (see r_drw2d_FixedP_t).
Y	Vertical position (see r_drw2d_FixedP_t).

See also

R_DRW2D_DrawRect

5.5.3 r_drw2d_Vec4_t

Description

A 3D point / vertex (4 component fixed point vector).
This vertex type is used for the assessment of vertex transformations (see R_DRW2D_VtxTransform).

Definition

```
typedef struct
{
  r_drw2d_FixedP_t X;
  r_drw2d_FixedP_t Y;
  r_drw2d_FixedP_t Z;
  r_drw2d_FixedP_t W;
} r_drw2d_Vec4_t;
```

Table 5-33 Member of r_drw2d_Vec4_t structure

Member	Description
X	Horizontal position (see r_drw2d_FixedP_t).
Y	Vertical position (see r_drw2d_FixedP_t).
Z	'Stacked' position (see r_drw2d_FixedP_t).
W	Perspective information (see r_drw2d_FixedP_t).

See also

R_DRW2D_VtxTransform

5.5.4 r_drw2d_Size_t

Description

Specifies the size of a rectangle (fixed point).

Definition

```
typedef struct
{
  r_drw2d_FixedP_t Width;
  r_drw2d_FixedP_t Height;
} r_drw2d_Size_t;
```

Table 5-34 Member of r_drw2d_Size_t structure

Member	Description
Width	Horizontal size (see r_drw2d_FixedP_t).
Height	Vertical size (see r_drw2d_FixedP_t).

See also

R_DRW2D_DrawRect

5.5.5 r_drw2d_Rect_t

Description

A 2D rectangle, described by position and dimension (fixed point coordinates).

Definition

```
typedef struct
{
  r_drw2d_Point_t  Pos;
  r_drw2d_Size_t   Size;
} r_drw2d_Rect_t;
```

Table 5-35 Member of r_drw2d_Rect_t structure

Member	Description
Pos	Position (see r_drw2d_Point_t).
Size	Size (see r_drw2d_Size_t).

See also

R_DRW2D_DrawRect

5.5.6 r_drw2d_IntPoint_t

Description

A 2D point (integer).

Definition

```
typedef struct
{
  int32_t   X;
  int32_t   Y;
} r_drw2d_IntPoint_t;
```

Table 5-36 Member of r_drw2d_IntPoint_t structure

Member	Description
X	Horizontal position.
Y	Vertical position.

See also

r_drw2d_IntRect_t

5.5.7 r_drw2d_IntSize_t

Description

Specifies the size of a rectangle (integer coordinates).

Definition

```
typedef struct
{
  int32_t   Width;
  int32_t   Height;
} r_drw2d_IntSize_t;
```

Table 5-37 Member of r_drw2d_IntSize_t structure

Member	Description
Width	Horizontal size.
Height	Vertical size.

See also

r_drw2d_IntRect_t, r_drw2d_Buffer_t

5.5.8 r_drw2d_IntRect_t

Description

A 2D rectangle, described by position and dimension (integer coordinates).

Definition

```
typedef struct
{
  r_drw2d_IntPoint_t  Pos;
  r_drw2d_IntSize_t   Size;
} r_drw2d_IntRect_t;
```

Table 5-38 Member of r_drw2d_IntRect_t structure

Member	Description
Pos	Position (see r_drw2d_IntPoint_t).
Size	Size (see r_drw2d_IntSize_t).

See also

R_DRW2D_CtxClipRect

5.5.9 r_drw2d_UVCoord_t

Description

A 2D, normalized U/V coordinate (fixed point).

Note:
U/V are pre-scaled by 256, i.e. (256.0, 256.0) always maps to the bottom right texture corner.
R_DRW2D_2U and R_DRW2D_2V are supported macros to make this value.
(U,V) = (R_DRW2D_2U(0.0), R_DRW2D_2V(0.0)) is top-left.
(U,V) = (R_DRW2D_2U(1.0), R_DRW2D_2V(1.0)) is bottom-right.

The texture matrix (see R_DRW2D_CtxTextureTransform) can be used to setup custom texture coordinate systems.

Definition

```
typedef struct
{
  r_drw2d_FixedP_t U;
  r_drw2d_FixedP_t V;
} r_drw2d_UVCoord_t;
```

Table 5-39 Member of r_drw2d_UVCoord_t structure

Member	Description
U	Normalized horizontal texel position, scaled by 256 (see r_drw2d_FixedP_t).
V	Normalized vertical texel position, scaled by 256 (see r_drw2d_FixedP_t).

See also

R_DRW2D_DrawRectUV, R_DRW2D_DrawTrianglesUV

5.5.10 r_drw2d_Buffer_t

Description

Buffers are used with R_DRW2D_FramebufferSet and textures (R_DRW2D_CtxTextureSet).
The buffer starts at the top/left corner.

Note: Please consult target-specific documentation regarding alignment rules (4bits-per-pixel, scanlines, start address).

Definition

```
typedef struct
{
  void                    *Data;
  int32_t                  Pitch;
  r_drw2d_IntSize_t        Size;
  r_drw2d_PixelFormat_t    PixelFormat;
} r_drw2d_Buffer_t;
```

Table 5-40 Member of r_drw2d_Buffer_t structure

Member	Description
Data	Reference to pixel data (Specify the physical address that GPU can access).
Pitch	Total number of pixels per line (including alignment/padding).
Size	Buffer width and height.
PixelFormat	Pixel storage format. See r_drw2d_PixelFormat_t.

See also

none

5.5.11 r_drw2d_Framebuffer_t

Description

Framebuffer handle and attributes.

Definition

```
typedef struct
{
  void                        *Handle;
  r_drw2d_Buffer_t             Buffer;
  r_drw2d_FramebufferFlags_t  Flags;
} r_drw2d_Framebuffer_t;
```

Table 5-41 Member of r_drw2d_Framebuffer_t structure

Member	Description
Handle	Internal framebuffer handle (must not be modified by application).
Buffer	Stores framebuffer geometry and pixel format.
Flags	reserved for future extensions (see r_drw2d_FramebufferFlags_t).

See also

R_DRW2D_FramebufferSet

5.5.12 r_drw2d_Texture_t

Description

Texture handle and attributes.

Definition

```
typedef struct
{
  void                    *Handle;
  r_drw2d_Buffer_t         Buffer;
  r_drw2d_TextureFlags_t   Flags;
} r_drw2d_Texture_t;
```

Table 5-42 Member of r_drw2d_Texture_t structure

Member	Description
Handle	Internal texture handle. (must not be modified by application).
Buffer	Stores texture geometry and pixel format.
Flags	Texture flags, see r_drw2d_TextureFlags_t.

See also

r_drw2d_Buffer_t, R_DRW2D_CtxTextureSet

5.5.13 r_drw2d_EffectParam_t

Description

Effect parameter contains parameter information for one parameter of one effect.

Definition

```
typedef struct
{
 r_drw2d_EffectParamSource_t   Source;
 union
 {
    struct
    {
       union
       {
          uint32_t            TextureUnit;
          r_drw2d_Color_t     ConstantColor;
       } Source;
       r_drw2d_EffectColorParamOperand_t   Operand;
    } Color;
    r_drw2d_FixedP_t           Constant;
    r_drw2d_Point_t            Point;
 } Param;
} r_drw2d_EffectParam_t;
```

Table 5-43 Member of r_drw2d_EffectParam_t structure

Member	Description
Source	Source of the parameter (type r_drw2d_EffectParamSource_t), declares what 'Param' is.
Param	The parameter itself as a union. Can be .Color, .Constant or .Point, as indicated by 'Source'. See below for details.
Param.Color	A struct with the fields .Operand and .Source. See below for details.
Param.Color.Operand	Defines how the color value is to be used, e.g. inverted or not (type r_drw2d_EffectColorParamOperand_t).
Param.Color.Source	A union, either .TextureUnit or .ConstantColor, as indicated by 'Source' above.
Param.Color.Source.TextureUnit	Index of a texture unit.
Param.Color.Source.ConstantColor	32bit ARGB8888 color value (type r_drw2d_Color_t).
Param.Constant	16.16 signed fixed point constant value (type r_drw2d_FixedP_t).
Param.Point	16.16 signed 2D fixed point coordinate (type r_drw2d_Point_t).

See also

r_drw2d_EffectName_t, r_drw2d_EffectParam_t, R_DRW2D_CtxEffectsSet, R_DRW2D_CtxEffectsDelete, R_DRW2D_CtxEffectsUpdate

5.5.14 r_drw2d_EffectStage_t

Description

Effect stage contains information about one effect.

Definition

```
typedef struct
{
  r_drw2d_EffectName_t   Name;
  r_drw2d_EffectParam_t  Args[4];
} r_drw2d_EffectStage_t;
```

Table 5-44 Member of r_drw2d_EffectStage_t structure

Member	Description
Name	Name of the effect, type r_drw2d_EffectName_t.
Args	Parameters of the effect, array of type r_drw2d_EffectParam_t (used length of this array depends on 'Name' of the effect, maximum is 4).

See also

r_drw2d_EffectName_t, r_drw2d_EffectParam_t, R_DRW2D_CtxEffectsSet, R_DRW2D_CtxEffectsDelete, R_DRW2D_CtxEffectsUpdate

5.5.15 r_drw2d_ConvKernel_t

Description

Convolution kernel containing dimensions and coefficients of a kernel.

Definition

```
typedef struct
{
  const r_drw2d_FixedP_t           *Coeff;
  r_drw2d_ConvKernelColorChannel_t  Channel;
  int32_t                           Width;
  int32_t                           Height;
  r_drw2d_FixedP_t                  Bias;
} r_drw2d_ConvKernel_t;
```

Table 5-45 Member of r_drw2d_ConvKernel_t structure

Member	Description
Coeff	The coefficients of the kernel (row by row).
Channel	The color channels being processed by the kernel.
Width	Width of the kernel.
Height	Height of the kernel.
Bias	Bias value that is added to the resulting color channel values (range: -1.0 to 1.0).

See also

R_DRW2D_CtxConvolutionKernel, R_DRW2D_DrawRectConvolve2d

5.5.16 r_drw2d_DeviceBase_t

Description

Common base class type for all Drw2D-internal device contexts.

Definition

```
struct r_drw2d_DeviceBase_s;
typedef struct r_drw2d_DeviceBase_s r_drw2d_DeviceBase_t;
```

See also

none

5.5.17 r_drw2d_RenderContext_s

Description

Render context attributes.

Definition

```
struct r_drw2d_RenderContext_s
{
 r_drw2d_DeviceBase_t                *DeviceBase;
 r_drw2d_IntRect_t                   ClipRect;
 r_drw2d_IntRect_t                   Viewport;
 r_drw2d_Color_t                     FgColor;
 r_drw2d_Color_t                     BgColor;
 r_drw2d_FillMode_t                  FillMode;
 r_drw2d_LineStyle_t                 LineStyle;
 r_drw2d_ImgQuality_t                ImgQuality;
 r_drw2d_TransformMode_t             TransformMode;
 r_drw2d_TextureTransformMode_t      TextureTransformMode;
 r_drw2d_BlendMode_t                 BlendMode;
 r_drw2d_CullMode_t                  CullMode;
 r_drw2d_Boolean_t                   EnableStriping;
 r_drw2d_ConvolutionKernelPreset1d_t ConvKernelPreset1d;
 r_drw2d_ConvolutionKernelPreset2d_t ConvKernelPreset2d;
 const r_drw2d_ConvKernel_t*         ConvKernel;
 r_drw2d_ConvMode_t                  ConvMode;
 r_drw2d_EffectStage_t*              EffectStages;
 uint32_t                            NumberOfStages;
 uint32_t                            ClutBase;
 struct
 {
    r_drw2d_BlendFactor_t            SrcRGB;
    r_drw2d_BlendFactor_t            DstRGB;
    r_drw2d_BlendFactor_t            SrcAlpha;
    r_drw2d_BlendFactor_t            DstAlpha;
 } BlendFactors;
 r_drw2d_FixedP_t                    TextureMatrix[3*2];
 r_drw2d_FixedP_t                    VertexMatrix[4*4];
 r_drw2d_Texture_t                   Texture[2];
}
```

Table 5-46 Member of r_drw2d_RenderContext_s structure

Member	Description
DeviceBase	Parent device context or NULL if render context is unused
ClipRect	Clipping rectangle (not clipped to current framebuffer).
Viewport	View port.
FgColor	Foreground color (ARGB32).
BgColor	Background color (ARGB32).
FillMode	Fill mode.
LineStyle	Line join/cap style, line width, miter limit and "closed" flag.
ImgQuality	Current image quality (antialiasing mode).
TransformMode	Vertex transform mode (see r_drw2d_TransformMode_t).
TextureTransformMode	Texture transform mode (See r_drw2d_TextureTransformMode_t).
BlendMode	Blend mode.
CullMode	Cull mode (See r_drw2d_CullMode_t).
EnableStriping	Enable stripping.
ConvKernelPreset1d	1D convolution filter kernel presets.
ConvKernelPreset2d	2D convolution filter kernel presets.
ConvKernel	Convolution kernel (see r_drw2d_ConvKernel_t). - kernel width and height needs to be an odd value
ConvMode	Convolution mode (see r_drw2d_ConvMode_t).
EffectStages	Effect stages.
NumberOfStages	Number of effect stages.
BlendFactors	Used when BlendMode is set to R_DRW2D_BLENDMODE_CUSTOM.
BlendFactors.SrcRGB	The blend factor to be used for source RGB values (see r_drw2d_BlendFactor_t).
BlendFactors.DstRGB	The blend factor to be used for destination RGB values (see r_drw2d_BlendFactor_t).
BlendFactors.SrcAlpha	The blend factor to be used for source alpha values (see r_drw2d_BlendFactor_t).
BlendFactors.DstAlpha	The blend factor to be used for destination alpha values (see r_drw2d_BlendFactor_t).
TextureMatrix	Current texture matrix.
VertexMatrix	Current vertex matrix.
Texture	Current texture.

See also

r_drw2d_RenderContext_t

5.5.18 r_drw2d_RenderContext_t

Description

Structure type for the render context.

Definition

```
typedef struct r_drw2d_RenderContext_s r_drw2d_RenderContext_t;
```

See also

r_drw2d_RenderContext_s

5.5.19 r_drw2d_DeviceBase_s

Description

Common base class for all Drw2D-internal device contexts.
This structure must be the first field of all driver-specific device context structures.

Definition

```
struct r_drw2d_DeviceBase_s
{
 uint32_t                       NumBytes;
 r_drw2d_Unit_t                 Unit;
 r_drw2d_OS_Mutex_t             Mutex;
 struct
 {
    void                       *UserData;
    r_drw2d_ErrorCallback_t     Callback;
 } Error;
 r_drw2d_FixedP_t               DepthRangeNear;
 r_drw2d_FixedP_t               DepthRangeFar;
 r_drw2d_RenderContext_t        DefaultRenderContext;
 struct
 {
    r_drw2d_RenderContext_t    *CurrentRenderContext;
    r_drw2d_Framebuffer_t       CurrentFramebuffer;
    uint32_t                    APIDirtyFlags;
    uint32_t                    SysDirtyFlags;
    r_drw2d_IntRect_t           EffectiveClipRect;
    r_drw2d_Point_t             TexCoordsTrans[3];
    r_drw2d_ConvolveMode_t      ConvolveMode;
 } State;
}
```

Table 5-47 Member of r_drw2d_DeviceBase_s structure

Member	Description
NumBytes	Total size (in bytes) of driver specific device context (including DeviceBase structure).
Unit	Parent unit# of device context (Drw2D unit number.).
Mutex	Used to synchronize access to this device context.
Error	Error callback function pointer and user data (see R_DRW2D_ErrCallbackSet, r_drw2d_ErrorCallback_t).
Error.UserData	Arbitrary user data that is passed on to the error callback function.
Error.Callback	Error callback function pointer.
DepthRangeNear	Depth range of near clipping plane.
DepthRangeFar	Depth range of far clipping plane.
DefaultRenderContext	Default render context.
State.CurrentRenderContext	Reference to the current render context. Points whether to the DefaultRenderContext or a user-set context.
State.CurrentFramebuffer	Reference to the current framebuffer. Points whether to the default FB or a user-set FB.
State.APIDirtyFlags	Bitmask that indicates what to update on API-side when a Draw*() function is called.
State.SysDirtyFlags	Bitmask that indicates what to update on Sys-side when a Draw*() function is called.
State.EffectiveClipRect	Current clipping rectangle.
State.TexCoordsTrans	Transformed texture coordinate cache for static texture mapping.
State.ConvolveMode	Convolution filter mode (1d horizontal, 1d vertical, 2d).

See also

r_drw2d_Device_t, r_drw2d_DeviceBase_t

5.6 Macros

This section shows the macros used in Drw2D API function.

5.6.1 R_DRW2D_ERROR_CLASS

Description

Mask out error sub-code (lower 16bits of error code).

Definition

```
#define R_DRW2D_ERROR_CLASS(a) ((a) & 0x7FFF0000)
```

Arguments

Table 5-48 Parameter of R_DRW2D_ERROR_CLASS macro

Parameter	Description
a	Error code.

Returns

Error class (lower 16bits of error code set to 0).

See also

r_drw2d_Error_t

5.6.2 R_DRW2D_2X

Description

Convert integer or float to fixed point.

Definition

```
#define R_DRW2D_2X(v) ((r_drw2d_FixedP_t)((v) * 65536))
```

Arguments

Table 5-49 Parameter of R_DRW2D_2X macro

Parameter	Description
v	Integer or float value.

Returns

Fixed point value.

See also

r_drw2d_FixedP_t, R_DRW2D_2I, R_DRW2D_2F, R_DRW2D_2U

5.6.3 R_DRW2D_2I

Description

Convert fixed point to int.

Definition

```
#define R_DRW2D_2I(x)  (((int32_t)(x)) / 65536)
```

Arguments

Table 5-50 Parameter of R_DRW2D_2I macro

Parameter	Description
x	Fixed point value.

Returns

Integer value.

See also

r_drw2d_FixedP_t, R_DRW2D_2X, R_DRW2D_2F

5.6.4 R_DRW2D_2F

Description

Convert fixed point to float.

Definition

```
#define R_DRW2D_2F(x) (((float32_t)(x)) * (1.0f / 65536.0f))
```

Arguments

Table 5-51 Parameter of R_DRW2D_2F macro

Parameter	Description
x	Fixed point value.

Returns

Float value.

See also

r_drw2d_FixedP_t, R_DRW2D_2I, R_DRW2D_2X

5.6.5 R_DRW2D_2U

Description

Convert integer or float to fixed point normalized texture U coordinate.

Note: Texture coordinates are prescaled by 256.

Definition

```
#define R_DRW2D_2U(v) ((r_drw2d_FixedP_t)((v) * (256 * 65536)))
```

Arguments

Table 5-52 Parameter of R_DRW2D_2U macro

Parameter	Description
v	Integer or float value.

Returns

Fixed point value.

See also

r_drw2d_FixedP_t, R_DRW2D_2I, R_DRW2D_2F

5.6.6 R_DRW2D_2V

Description

Convert integer or float to fixed point normalized texture V coordinate.

Note: Texture coordinates are prescaled by 256.

Definition

```
#define R_DRW2D_2V(v) ((r_drw2d_FixedP_t)((v) * (256 * 65536)))
```

Arguments

Table 5-53 Parameter of R_DRW2D_2V macro

Parameter	Description
v	Integer or float value.

Returns

Fixed point value.

See also

r_drw2d_FixedP_t, R_DRW2D_2I, R_DRW2D_2F

6. Appendix

6.1 Optimization hints

- Avoid blitting rotated textures specified in Flash-ROM by Drw2D, copy it first to the RAM.
- When rotating, use texture flag R_DRW2D_TEX_VT or R_DRW2D_TEX_SWIZZLE; For swizzling you first need to prepare your textures ('swizzle' them) by an appropriate tool.
- In case of non-rotated textures, take care when using R_DRW2D_TEX_VT or R_DRW2D_TEX_SWIZZLE texture flags, because they can improve as well as worsen the performance, depending on the use-case.
- Using R_DRW2D_TEX_BILINEAR introduces some performance penalty, so use judiciously, i.e. make sure to turn it off if the output quality suffices without it.
- Align the texture address to 128 byte boundary (due to bus cache-line).
- For blitting one texture several times: if not already there, make sure to copy it to the RAM first.
- Use clipping rectangle feature to restrain the area needed to be updated by the GPU.
- Using transformation (rotate, wrap, scale, etc.) or filtering (bilinear filter) on compressed textures (like RLE) will cause a performance penalty.

6.2 Programming alignment table

Table 6-1 Programming alignment

Action	Alignment	Constraint	Directly affected APIs	Comment
Drw2D framebuffer address	128 bytes	D/AVE HD	R_DRW2D_FramebufferSet	
Drw2D framebuffer pitch	None	D/AVE HD	R_DRW2D_FramebufferSet	Recommended:128 bytes
Drw2D texture address	any	D/AVE HD	R_DRW2D_CtxTextureSet	Recommended:128 bytes
Drw2D RLE texture address	8 bytes	D/AVE HD	R_DRW2D_CtxTextureSet	Recommended:128 bytes

6.3 Setting range of parameter

In this section, the setting range of parameters of each Drw2DAPI is shown in *Table 6-2*.

Table 6-2 Setting range of parameter List

Function Name	Parameter	Setting Range
R_DRW2D_FixSin	Angle (see r_drw2d_FixedP_t)	0.0~4.0 (0-degree ~ 360-degree)
R_DRW2D_FixCos	Angle (see r_drw2d_FixedP_t)	0.0~4.0 (0-degree ~ 360-degree)
R_DRW2D_FixTan	Angle (see r_drw2d_FixedP_t)	0.0~4.0 (0-degree ~ 360-degree)
R_DRW2D_Open	Unit (see r_drw2d_Unit_t)	0
	DriverUnit (see int32_t)	0
R_DRW2D_NativeDriverEnd	Flags (See r_drw2d_NativeDrvFlags_t)	0
R_DRW2D_CtxClipRect	Rect (see r_drw2d_IntRect_t).	Rect.Pos.X: 0~4095 Rect.Pos.Y: 0~4095 Rect.Size.Width: 0~4095 Rect.Size.Height: 0~4095
R_DRW2D_CtxViewport	Rect (see r_drw2d_IntRect_t).	Rect.Pos.X: 0~4095 Rect.Pos.Y: 0~4095 Rect.Size.Width: 0~4095 Rect.Size.Height: 0~4095
R_DRW2D_CtxEffectsSet	Effects (see r_drw2d_EffectStage_t)	Effects.Args.Param.Color.Source.TextuteUnit: 0~1
	Count (see uint32_t)	1~4294967295

Function Name	Parameter	Setting Range
R_DRW2D_CtxEffectsUpdate	Count (see uint32_t)	0~65536
	Params (see r_drw2d_EffectParam_t)	Params.Param.Color.Source.TextureUnit: 0~1
R_DRW2D_CtxTextureSet	TextureUnit (see uint32_t)	0~1
	Texture (see r_drw2d_Texture_t)	Texture.Buffer.Pitch:0~4095 Texture.Buffer.Size.Width: 1~4095 Texture.Buffer.Size.Height: 1~4095
R_DRW2D_TextureBlit	SrcRect (see r_drw2d_Rect_t)	SrcRect.Pos.X: -4096.0~4095.0 SrcRect.Pos.Y: -4096.0~4095.0 SrcRect.Size.Width: 0.0~4095.0 SrcRect.Size.Height: 0.0~4095.0
	DstRect (see r_drw2d_Rect_t)	DstRect.Pos.X: -4096.0~4095.0 DstRect.Pos.Y: -4096.0~4095.0 DstRect.Size.Width: 0.0~4095.0 DstRect.Size.Height: 0.0~4095.0
R_DRW2D_CtxRotate	Angle (see r_drw2d_FixedP_t)	0~360
R_DRW2D_CtxRotate3d	X (see r_drw2d_FixedP_t)	-1.0~1.0
	Y (see r_drw2d_FixedP_t)	-1.0~1.0
	Z (see r_drw2d_FixedP_t)	-1.0~1.0
	Angle (see r_drw2d_FixedP_t)	0~360
R_DRW2D_CtxTextureRotate	Angle (see r_drw2d_FixedP_t)	0~360
R_DRW2D_CtxTranslate	TransX (see r_drw2d_FixedP_t)	-4096.0~4095.0
	TransY (see r_drw2d_FixedP_t)	-4096.0~4095.0
	TransZ (see r_drw2d_FixedP_t)	-4096.0~4095.0
R_DRW2D_CtxTextureTranslate	TransX (see r_drw2d_FixedP_t)	-4096.0~4095.0
	TransY (see r_drw2d_FixedP_t)	-4096.0~4095.0
R_DRW2D_CtxFrustum	Left (see r_drw2d_FixedP_t)	-4096.0~4095.0
	Right (see r_drw2d_FixedP_t)	-4096.0~4095.0
	Bottom (see r_drw2d_FixedP_t)	-4096.0~4095.0
	Top (see r_drw2d_FixedP_t)	-4096.0~4095.0
R_DRW2D_VtxTransform	Vertices (see r_drw2d_Vec4_t)	Vertices.X: -4096.0~4095.0 Vertices.Y: -4096.0~4095.0 Vertices.Z: -4096.0~4095.0 Vertices.W: -4096.0~4095.0
	NumVertices (see uint32_t)	0~4294967295
R_DRW2D_FramebufferSet	Framebuffer (see r_drw2d_Framebuffer_t)	Framebuffer.Buffer.Pitch: 0~4095 Framebuffer.Buffer.Size.Width: 1~4095 Framebuffer.Buffer.Size.Height: 1~4095 Framebuffer.Flags: 0
R_DRW2D_DrawTriangles	Points (see r_drw2d_Point_t)	Points.X: -4096.0~4095.0 Points.Y: -4096.0~4095.0
	Count (see uint32_t)	3~65535

Function Name	Parameter	Setting Range
R_DRW2D_DrawTrianglesUV	Points (see r_drw2d_Point_t)	Points.X: -4096.0~4095.0 Points.Y: -4096.0~4095.0
	Count (see uint32_t)	3~65535
R_DRW2D_DrawRect	Rect (see r_drw2d_Rect_t)	Rect.Pos.X: -4096.0~4095.0 Rect.Pos.Y: -4096.0~4095.0 Rect.Size.Width: 0.0~4095.0 Rect.Size.Height: 0.0~4095.0
R_DRW2D_DrawRectUV	Rect (see r_drw2d_Rect_t)	Rect.Pos.X: -4096.0~4095.0 Rect.Pos.Y: -4096.0~4095.0 Rect.Size.Width: 0.0~4095.0 Rect.Size.Height: 0.0~4095.0
R_DRW2D_DrawQuads	Points (see r_drw2d_Point_t)	Points.X: -4096.0~4095.0 Points.Y: -4096.0~4095.0
	Count (see uint32_t)	4~65536
R_DRW2D_DrawQuadsUV	Points (see r_drw2d_Point_t)	Points.X: -4096.0~4095.0 Points.Y: -4096.0~4095.0
	Count (see uint32_t)	4~65536
R_DRW2D_DrawQuads3dUV	Points (see r_drw2d_Point_t)	Points.X: -4096.0~4095.0 Points.Y: -4096.0~4095.0
	Count (see uint32_t)	4~65536
R_DRW2D_DrawEllipse	Points (see r_drw2d_Point_t)	Points.X: -4096.0~4095.0 Points.Y: -4096.0~4095.0
R_DRW2D_DrawLines	Points (see r_drw2d_Point_t)	Points.X: -4096.0~4095.0 Points.Y: -4096.0~4095.0
	Count (see uint32_t)	2~65536
R_DRW2D_DrawPolyline	Points (see r_drw2d_Point_t)	Points.X: -4096.0~4095.0 Points.Y: -4096.0~4095.0
	Count (see uint32_t)	2~65536
R_DRW2D_DrawBezierCurves	Points (see r_drw2d_Point_t)	Points.X: -4096.0~4095.0 Points.Y: -4096.0~4095.0
	Count (see uint32_t)	3~65535
R_DRW2D_DrawRectConvolve1dx	Rect (see r_drw2d_IntRect_t)	Rect.Pos.X: -4096~4095 Rect.Pos.Y: -4096~4095 Rect.Size.Width: 0~4095 Rect.Size.Height: 0~4095
	TextureOffX (see uint16_t)	0~4095
	TextureOffY (see uint16_t)	0~4095
R_DRW2D_DrawRectConvolve1dy	Rect (see r_drw2d_IntRect_t)	Rect.Pos.X: -4096~4095 Rect.Pos.Y: -4096~4095 Rect.Size.Width: 0~4095 Rect.Size.Height: 0~4095
	TextureOffX (see uint16_t)	0~4095
	TextureOffY (see uint16_t)	0~4095
R_DRW2D_DrawRectConvolve2d	Rect (see r_drw2d_IntRect_t)	Rect.Pos.X: -4096~4095 Rect.Pos.Y: -4096~4095 Rect.Size.Width: 0~4095 Rect.Size.Height: 0~4095
	TextureOffX (see uint16_t)	0~4095

Function Name	Parameter	Setting Range
	TextureOffY (see uint16_t)	0~4095
R_DRW2D_DrawRectConvolve	Rect (see r_drw2d_IntRect_t)	Rect.Pos.X: -4096~4095 Rect.Pos.Y: -4096~4095 Rect.Size.Width: 0~4095 Rect.Size.Height: 0~4095
	TextureOffX (see uint16_t)	0~4095
	TextureOffY (see uint16_t)	0~4095
R_DRW2D_GetGaussKernel	Width (see int32_t)	1~4095
	Height (see int32_t)	1~4095
R_DRW2D_CtxConvolutionKernel	Kernel (see r_drw2d_ConvKernel_t)	Kernel.Width: 1~7 Kernel.Height: 1~7 Kernel.Bias: -1.0~1.0

Revision History	Renesas Graphics Library 2D Graphics (DRW2D) Driver User's Manual: Software

Rev.	Date	Description	
		Page	Summary
0.1	Dev 03, 2019	-	First edition.
1.0	April 24, 2020	-	Update Revision.
1.1	Nov 05, 2020	14	Add the description of Texture Unit1.
		16, 26, 38, 39, 40,	Fix typo.
		19, 22	Change the suitable effect type for fade in/out.
		22	Fix the formular of Constant Alpha effect.
		23	Add the restriction of Gradient effect depending on the transform mode.
		27	Add the description of pre-multiplied alpha.
		32, 64	Fix the description of R_DRW2D_CtxViewport.
		83	Add the behavior example depending on execution order.
		103	Add the description of R_DRW2D_DrawQuadsUV.
		106, 107, 110, 112	Add the transform specification by vertex matrix.
		106, 112	Add the restriction of image quality mode.
		107, 110, 112	Add the restriction of transparency.
		114, 116, 118, 120	Add the restriction of parameter and texture color format.
		156	Fix the formular of R_DRW2D_BLENDMODE_ADDITIVE.
		205	Fix the range of Texture.Buffer.Pitch.
1.2	July 16, 2021	-	Update Revision.
2.0	Dec 01, 2021	-	Update Revision.

Renesas Graphics Library
2D Graphics (DRW2D) Driver
User's Manual: Software
Publication Date: Rev.0.1 Dev 03, 2019
Rev.2.0 Dec 01, 2021

Published by: Renesas Electronics Corporation

Renesas Graphics Library
2D Graphics (DRW2D) Driver

Renesas Electronics Corporation