

# INGROUP\_BUG

Generated by Doxygen 1.5.9

Mon May 17 18:54:48 2010



# Contents

<b>1</b>	<b>Shapes API Documentation</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	Another Section . . . . .	1
1.2.1	A Sub Section . . . . .	1
<b>2</b>	<b>Module Index</b>	<b>3</b>
2.1	Modules . . . . .	3
<b>3</b>	<b>Data Structure Index</b>	<b>5</b>
3.1	Data Structures . . . . .	5
<b>4</b>	<b>File Index</b>	<b>7</b>
4.1	File List . . . . .	7
<b>5</b>	<b>Module Documentation</b>	<b>9</b>
5.1	Shapes APIs . . . . .	9
5.1.1	Detailed Description . . . . .	9
5.2	Mandatory APIs . . . . .	10
5.2.1	Detailed Description . . . . .	10
5.2.2	Function Documentation . . . . .	10
5.2.2.1	Shapes_DeInit . . . . .	10
5.2.2.2	Shapes_Init . . . . .	10
5.3	Configuration APIs . . . . .	12
5.3.1	Detailed Description . . . . .	12
5.3.2	Function Documentation . . . . .	12
5.3.2.1	Shapes_EnableGPU . . . . .	12
5.3.2.2	Shapes_GetDbgLvl . . . . .	12
5.3.2.3	Shapes_GetGPUState . . . . .	13
5.3.2.4	Shapes_SetDbgLvl . . . . .	13
5.4	Operational APIs . . . . .	14

---

5.4.1	Detailed Description	14
5.4.2	Function Documentation	14
5.4.2.1	Shapes_DrawPoint	14
5.4.2.2	Shapes_DrawRect	14
5.4.2.3	Shapes_GetRectPerimeter	15
5.5	Enumerated Constants	16
5.5.1	Detailed Description	16
5.5.2	Enumeration Type Documentation	16
5.5.2.1	Shapes_DbgLvlType	16
5.5.2.2	Shapes_ShapeType	16
5.6	Data Structures	17
5.6.1	Detailed Description	17
<b>6</b>	<b>Data Structure Documentation</b>	<b>19</b>
6.1	Shapes_PtType Struct Reference	19
6.1.1	Detailed Description	19
6.1.2	Field Documentation	19
6.1.2.1	x	19
6.1.2.2	y	19
6.2	Shapes_RectType Struct Reference	20
6.2.1	Detailed Description	20
6.2.2	Field Documentation	20
6.2.2.1	mBottomRight	20
6.2.2.2	mTopLeft	20
<b>7</b>	<b>File Documentation</b>	<b>21</b>
7.1	Shapes.h File Reference	21
7.1.1	Detailed Description	22
7.1.2	Define Documentation	22
7.1.2.1	SHAPES_ERROR	22
7.1.2.2	SHAPES_SUCCESS	22

# Chapter 1

## Shapes API Documentation

### 1.1 Introduction

[This section should talk briefly about the API itself. Should give a brief overview of what the API enables.] Shapes API provides a framework to manipulate various geometrical shapes like Rectangles,circles etc. It also has the ability to use the underlying GPU to accelerate the processing of the shapes.

### 1.2 Another Section

Multiple sections can be added using the "section" doxygen tag.

#### 1.2.1 A Sub Section

To better document the introduction, one can also use subsection tags. This will appear under the "Another Section" Section.



# Chapter 2

## Module Index

### 2.1 Modules

Here is a list of all modules:

Shapes APIs . . . . .	9
Mandatory APIs . . . . .	10
Configuration APIs . . . . .	12
Operational APIs . . . . .	14
Enumerated Constants . . . . .	16
Data Structures . . . . .	17



# Chapter 3

## Data Structure Index

### 3.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">Shapes_PtType</a> . . . . .	19
<a href="#">Shapes_RectType</a> . . . . .	20



# Chapter 4

## File Index

### 4.1 File List

Here is a list of all documented files with brief descriptions:

<a href="#">Shapes.h</a> (This file contains the definitions of the constants, data structures, and interfaces that comprise the Shapes API. Only the interfaces declared shall be used by the client for accessing the Shapes API) . . . . .	21
<b>Shapes_MainPage.h</b> . . . . .	??



# Chapter 5

## Module Documentation

### 5.1 Shapes APIs

#### Modules

- [Mandatory APIs](#)
- [Configuration APIs](#)
- [Operational APIs](#)
- [Enumerated Constants](#)
- [Data Structures](#)

#### 5.1.1 Detailed Description

This group groups all the APIs in the Shapes API module.

## 5.2 Mandatory APIs

- uint32 [Shapes\\_Init](#) (void \*handle)  
*Initializes the Shapes API.*
- uint32 [Shapes\\_DeInit](#) (void \*pHandle)  
*Deinitializes the Shapes API.*

### 5.2.1 Detailed Description

This is an example group which groups mandatory APIs.

### 5.2.2 Function Documentation

#### 5.2.2.1 uint32 Shapes\_DeInit (void \* *pHandle*)

Deinitializes the Shapes API.

This function de-initializes various internal modules that manage the shapes and also powers off the GPU.

#### Parameters:

↔ *\*pHandle* Pointer to the handle structure that will be cleared upon successful de-initialization

Shapes\_Init must have been called prior to this function

Any Shapes APIs that were being executed at the time of deinit will have undefined behavior. So care must be taken to ensure all pending operations were completed prior to the de-initialization.

#### Returns:

Returns one of the following values. The values can be nicely presented using item tags as follows:

- SHAPES\_SUCCESS
- SHAPES\_ERROR

#### See also:

[Shapes\\_Init](#)

#### 5.2.2.2 uint32 Shapes\_Init (void \* *handle*)

Initializes the Shapes API.

This function initializes various internal modules that manage the shapes and also powers on the GPU and restores it to a known default state.

#### Parameters:

→ *\*pHandle* Pointer to the handle structure that will be updated upon successful initialization

None

None

**Returns:**

Returns one of the following values. The values can be nicely presented using item tags as follows:

- SHAPES\_SUCCESS
- SHAPES\_ERROR

**See also:**

[Shapes\\_DeInit](#)

## 5.3 Configuration APIs

- void [Shapes\\_EnableGPU](#) (bool bEnable)  
*Enable/disables the GPU.*
- bool [Shapes\\_GetGPUState](#) ()  
*Get the state of the GPU.*
- void [Shapes\\_SetDbgLvl](#) ([Shapes\\_DbgLvlType](#) eDbgLvl)  
*Sets the current debug level.*
- [Shapes\\_DbgLvlType](#) [Shapes\\_GetDbgLvl](#) ()  
*Gets the current debug level.*

### 5.3.1 Detailed Description

This example section groups all the APIs used to configure the Shapes API

### 5.3.2 Function Documentation

#### 5.3.2.1 void [Shapes\\_EnableGPU](#) (bool *bEnable*)

Enable/disables the GPU.

Enables or disables the GPU based on the value of bEnable. If bEnable is TRUE GPU is enabled, disabled otherwise.

#### Parameters:

← *bEnable* Boolean to enable the GPU

[Shapes\\_Init](#) must have been called prior to this function

None

#### Returns:

None

#### See also:

[Shapes\\_GetGPUState](#)

#### 5.3.2.2 [Shapes\\_DbgLvlType](#) [Shapes\\_GetDbgLvl](#) ()

Gets the current debug level.

Gets the debug level current configured with the Shapes API

[Shapes\\_Init](#) must have been called prior to this function

None

**Returns:**

returns the debug level current configured with the Shapes API.

**See also:**

[Shapes\\_SetDbgLvl](#)

**5.3.2.3 bool Shapes\_GetGPUState ()**

Get the state of the GPU.

Get the Graphics Processing Unit's State

Shapes\_Init must have been called prior to this function

None

**Returns:**

Returns TRUE if GPU is enabled; FALSE Otherwise.

**See also:**

[Shapes\\_EnableGPU](#)

**5.3.2.4 void Shapes\_SetDbgLvl (Shapes\_DbgLvlType *eDbgLvl*)**

Sets the current debug level.

Sets the debug level for the Shapes API

**Parameters:**

← *eDbgLvl* Debug level to be used by the Shapes API

Shapes\_Init must have been called prior to this function

None

**Returns:**

None.

**See also:**

[Shapes\\_GetDbgLvl](#)

## 5.4 Operational APIs

- void [Shapes\\_DrawPoint](#) (void \*pHandle, [Shapes\\_PtType](#) mPt)  
*Draws a point.*
- void [Shapes\\_DrawRect](#) (void \*pHandle, [Shapes\\_RectType](#) mRect)  
*Draws a rectangle.*
- void [Shapes\\_GetRectPerimeter](#) (void \*pHandle, [Shapes\\_RectType](#) mRect, uint32 \*dwPerimeter)  
*Gets the perimeter of the rectangle.*

### 5.4.1 Detailed Description

This example section groups all the APIs that manipulate and draw shapes supported by the Shapes API

### 5.4.2 Function Documentation

#### 5.4.2.1 void [Shapes\\_DrawPoint](#) (void \* *pHandle*, [Shapes\\_PtType](#) *mPt*)

Draws a point.

Draws a point using mPt for its co-ordinates.

**Parameters:**

- ← *\*pHandle* Pointer to the handle from [Shapes\\_Init](#)
- ← *mPt* Point to be drawn

[Shapes\\_Init](#) must have been called prior to this function

None

**Returns:**

None

**See also:**

[Shapes\\_DrawRect](#)

#### 5.4.2.2 void [Shapes\\_DrawRect](#) (void \* *pHandle*, [Shapes\\_RectType](#) *mRect*)

Draws a rectangle.

Draws a rectangle using mRect for its co-ordinates.

**Parameters:**

- ← *\*pHandle* Pointer to the handle from [Shapes\\_Init](#)
- ← *mRect* Rectangle to be drawn

Shapes\_Init must have been called prior to this function

None

**Returns:**

None

**See also:**

[Shapes\\_DrawPoint](#)

**5.4.2.3 void Shapes\_GetRectPerimeter (void \* *pHandle*, Shapes\_RectType *mRect*, uint32 \* *dwPerimeter*)**

Gets the perimeter of the rectangle.

Computes the perimeter of the given rectangle using the following formula: [Doxygen Allows us to use LaTeX Formula for LaTeX/PDF only outputs]  $perimeter = 2 * (length + width)$  where  $length = mRect.mTopLeft.x - mRect.mBottomRight.x$  and  $width = mRect.mTopLeft.y - mRect.mBottomRight.y$

**Parameters:**

- ← ***pHandle*** Pointer to the handle from Shapes\_Init
- ← ***mRect*** Rectangle whose perimeter to be computed
- ***dwPerimeter*** Pointer to store the perimeter value

Shapes\_Init must have been called prior to this function

None

**Returns:**

None

**See also:**

None

## 5.5 Enumerated Constants

- enum `Shapes_ShapeType` { `SHAPE_SQUARE` = 0, `SHAPE_RECT` = 1, `SHAPE_CIRCLE` = 2, `SHAPE_ELLIPSE` = 3 }
- enum `Shapes_DbgLvType` {  
`DBGLVL_ALL` = 0, `DBGLVL_INFO` = 1, `DBGLVL_WARN` = 2, `DBGLVL_ERROR` = 3,  
`DBGLVL_FATAL` = 4 }

### 5.5.1 Detailed Description

This section defines the various enumerated constants. Please note that all the enums defined here are actually typedef'ed enums and can be used without the explicit enum keyword.

### 5.5.2 Enumeration Type Documentation

#### 5.5.2.1 enum `Shapes_DbgLvType`

Debug Levels used by the Shapes API to display messages.

**Enumerator:**

- DBGLVL\_ALL*** – Everything gets printed
- DBGLVL\_INFO*** – Informative messages
- DBGLVL\_WARN*** – Warning messages
- DBGLVL\_ERROR*** – Non-critical error messages
- DBGLVL\_FATAL*** – Critical error messages.

#### 5.5.2.2 enum `Shapes_ShapeType`

Various shapes supported by the Shapes API

**Enumerator:**

- SHAPE\_SQUARE*** – quadrilateral with four right angles and unequal lengths
- SHAPE\_RECT*** – quadrilateral with four right angles and equal lengths
- SHAPE\_CIRCLE*** – Curve with equal major and minor radius
- SHAPE\_ELLIPSE*** – Curve with unequal major and minor radius

## 5.6 Data Structures

### Data Structures

- struct [Shapes\\_PtType](#)

#### 5.6.1 Detailed Description

This section defines the various data structures published in the Shapes API. Please note that all the structs defined here are actually typedef'ed struct and can be used without the explicit struct keyword.



# Chapter 6

## Data Structure Documentation

### 6.1 Shapes\_PtType Struct Reference

```
#include <Shapes.h>
```

#### Data Fields

- [uint32 x](#)
- [uint32 y](#)

#### 6.1.1 Detailed Description

A structure that represents a point

#### 6.1.2 Field Documentation

##### 6.1.2.1 `uint32 Shapes_PtType::x`

– The X coordinate

##### 6.1.2.2 `uint32 Shapes_PtType::y`

– The Y coordinate

The documentation for this struct was generated from the following file:

- [Shapes.h](#)

## 6.2 Shapes\_RectType Struct Reference

```
#include <Shapes.h>
```

### Data Fields

- [Shapes\\_PtType mTopLeft](#)
- [Shapes\\_PtType mBottomRight](#)

### 6.2.1 Detailed Description

A structure that represents a rectangle

### 6.2.2 Field Documentation

#### 6.2.2.1 Shapes\_PtType Shapes\_RectType::mBottomRight

– Point that represents the bottom right corner of the rectangle

#### 6.2.2.2 Shapes\_PtType Shapes\_RectType::mTopLeft

– Point that represents the top left corner of the rectangle

The documentation for this struct was generated from the following file:

- [Shapes.h](#)

# Chapter 7

## File Documentation

### 7.1 Shapes.h File Reference

This file contains the definitions of the constants, data structures, and interfaces that comprise the Shapes API. Only the interfaces declared shall be used by the client for accessing the Shapes API.

#### Data Structures

- struct [Shapes\\_PtType](#)
- struct [Shapes\\_RectType](#)

#### Defines

##### Return Codes

*The following constants are grouped as Return Codes using the "name" doxygen tag*

- #define [SHAPES\\_SUCCESS](#) 0x0
- #define [SHAPES\\_ERROR](#) 0x8001

#### Enumerations

- enum [Shapes\\_ShapeType](#) { [SHAPE\\_SQUARE](#) = 0, [SHAPE\\_RECT](#) = 1, [SHAPE\\_CIRCLE](#) = 2, [SHAPE\\_ELLIPSE](#) = 3 }
- enum [Shapes\\_DbgLvIType](#) { [DBGLVL\\_ALL](#) = 0, [DBGLVL\\_INFO](#) = 1, [DBGLVL\\_WARN](#) = 2, [DBGLVL\\_ERROR](#) = 3, [DBGLVL\\_FATAL](#) = 4 }

#### Functions

- uint32 [Shapes\\_Init](#) (void \*handle)  
*Initializes the Shapes API.*
- uint32 [Shapes\\_DeInit](#) (void \*pHandle)  
*Deinitializes the Shapes API.*

- void [Shapes\\_EnableGPU](#) (bool bEnable)  
*Enable/disables the GPU.*
- bool [Shapes\\_GetGPUState](#) ()  
*Get the state of the GPU.*
- void [Shapes\\_SetDbgLvl](#) ([Shapes\\_DbgLvlType](#) eDbgLvl)  
*Sets the current debug level.*
- [Shapes\\_DbgLvlType](#) [Shapes\\_GetDbgLvl](#) ()  
*Gets the current debug level.*
  
- void [Shapes\\_DrawPoint](#) (void \*pHandle, [Shapes\\_PtType](#) mPt)  
*Draws a point.*
- void [Shapes\\_DrawRect](#) (void \*pHandle, [Shapes\\_RectType](#) mRect)  
*Draws a rectangle.*
- void [Shapes\\_GetRectPerimeter](#) (void \*pHandle, [Shapes\\_RectType](#) mRect, uint32 \*dwPerimeter)  
*Gets the perimeter of the rectangle.*

### 7.1.1 Detailed Description

This file contains the definitions of the constants, data structures, and interfaces that comprise the Shapes API. Only the interfaces declared shall be used by the client for accessing the Shapes API.

Anything that starts after a blank line is a detailed description. This is also a sample header file that serves as a template as a sample documentation for doxygen. Make sure you replace the DDISHapes.h above with the name of the header file

### 7.1.2 Define Documentation

#### 7.1.2.1 #define SHAPES\_ERROR 0x8001

– The API Failed for some reason

#### 7.1.2.2 #define SHAPES\_SUCCESS 0x0

– The API Succeeded

# Index

- Configuration APIs, 12
- Data Structures, 17
- DBGLVL\_ALL
  - shapes\_enums, 16
- DBGLVL\_ERROR
  - shapes\_enums, 16
- DBGLVL\_FATAL
  - shapes\_enums, 16
- DBGLVL\_INFO
  - shapes\_enums, 16
- DBGLVL\_WARN
  - shapes\_enums, 16
- Enumerated Constants, 16
- Mandatory APIs, 10
- mBottomRight
  - Shapes\_RectType, 20
- mTopLeft
  - Shapes\_RectType, 20
- Operational APIs, 14
- SHAPE\_CIRCLE
  - shapes\_enums, 16
- SHAPE\_ELLIPSE
  - shapes\_enums, 16
- SHAPE\_RECT
  - shapes\_enums, 16
- SHAPE\_SQUARE
  - shapes\_enums, 16
- Shapes APIs, 9
- Shapes.h, 21
  - SHAPES\_ERROR, 22
  - SHAPES\_SUCCESS, 22
- shapes\_enums
  - DBGLVL\_ALL, 16
  - DBGLVL\_ERROR, 16
  - DBGLVL\_FATAL, 16
  - DBGLVL\_INFO, 16
  - DBGLVL\_WARN, 16
  - SHAPE\_CIRCLE, 16
  - SHAPE\_ELLIPSE, 16
  - SHAPE\_RECT, 16
  - SHAPE\_SQUARE, 16
- shapes\_cfg\_apis
  - Shapes\_EnableGPU, 12
  - Shapes\_GetDbgLvl, 12
  - Shapes\_GetGPUState, 13
  - Shapes\_SetDbgLvl, 13
- Shapes\_DbgLvlType
  - shapes\_enums, 16
- Shapes\_DeInit
  - shapes\_mandatory\_apis, 10
- Shapes\_DrawPoint
  - shapes\_oper\_apis, 14
- Shapes\_DrawRect
  - shapes\_oper\_apis, 14
- Shapes\_EnableGPU
  - shapes\_cfg\_apis, 12
- shapes\_enums
  - Shapes\_DbgLvlType, 16
  - Shapes\_ShapeType, 16
- SHAPES\_ERROR
  - Shapes.h, 22
- Shapes\_GetDbgLvl
  - shapes\_cfg\_apis, 12
- Shapes\_GetGPUState
  - shapes\_cfg\_apis, 13
- Shapes\_GetRectPerimeter
  - shapes\_oper\_apis, 15
- Shapes\_Init
  - shapes\_mandatory\_apis, 10
- shapes\_mandatory\_apis
  - Shapes\_DeInit, 10
  - Shapes\_Init, 10
- shapes\_oper\_apis
  - Shapes\_DrawPoint, 14
  - Shapes\_DrawRect, 14
  - Shapes\_GetRectPerimeter, 15
- Shapes\_PtType, 19
  - x, 19
  - y, 19
- Shapes\_RectType, 20
  - mBottomRight, 20
  - mTopLeft, 20
- Shapes\_SetDbgLvl
  - shapes\_cfg\_apis, 13
- Shapes\_ShapeType
  - shapes\_enums, 16

SHAPES\_SUCCESS

Shapes.h, [22](#)

x

Shapes\_PtType, [19](#)

y

Shapes\_PtType, [19](#)