VectSharp

2.0.0

Generated by Doxygen 1.8.18

Chapter 1

VectSharp: a light library for C# vector graphics

1.1 Introduction

VectSharp is a library to create vector graphics (including text) in C#, without too many dependencies.

It includes an abstract layer on top of which output layers can be written. Currently, there are four available output layers: **VectSharp.PDF** produces PDF documents, **VectSharp.Canvas** produces an `Avalonia.Controls.Canvas` object (https://avaloniaui.net/docs/controls/canvas) containing the rendered graphics objects, **VectSharp.Raster** produces raster images in PNG format, and **VectSharp.SVG** produces vector graphics in SVG format.

`VectSharp.ThreeD` adds support for 3D vector and raster graphics.

`VectSharp.Markdown` can be used to transform Markdown documents into VectSharp objects, that can then be exported e.g. as PDF or SVG files, or displayed in an Avalonia `Canvas`. **VectSharp.MarkdownCanvas** uses VectSharp.Markdown to render Markdown documents in Avalonia applications (an example of this is in the `MarkdownViewerDemo project`).

VectSharp is written using .NET Core, and is available for Mac, Windows and Linux. Since version 2.0.0, it is released under an LGPLv3 license. It includes 14 standard fonts, originally released under an ASL-2.0 license.

Since version 2.0.0, VectSharp.Raster is released under an AGPLv3 license.

VectSharp.MuPDFUtils, also released under an AGPLv3 license, contains some utility functions that use `MuPDFCore` to make it possible to include in VectSharp graphics images in various formats.

VectSharp.Fonts.Nimbus is a package released under a GPLv3 license, which contains the standard fonts that were used in VectSharp before version 2.0.0. Since these fonts are released under a GPL license, they had to be replaced when the VectSharp license changed to LGPL. See the [Font libraries](section) below for information on how to re-enable these fonts.

1.2 Installing VectSharp

To include VectSharp in your project, you will need one of the output layer NuGet packages: `VectSharp.PDF`, `VectSharp.Canvas`, `VectSharp.Raster`, or `VectSharp.SVG`. You will need `VectSharp.ThreeD` to work with 3D graphics. You may want the `VectSharp.MuPDFUtils` package if you wish to manipulate raster images, and the `VectSharp.Fonts.Nimbus` if you want to restore the GPL-licensed fonts used in previous versions of the library.

1.3 Usage

You can find the full documentation for the VectSharp library at the documentation website. A PDF reference manual is also available.

In general, working with VectSharp involves: creating a `Document`, adding `Page`s, drawing to the `Page`s' `Graphics` objects and, finally, exporting them to a PDF document, `Canvas`, PNG image or SVG document.

- Create a `Document`:
```
using VectSharp;
// ...
Document doc = new Document();
```
- Add a `Page`:
```
doc.Pages.Add(new Page(1000, 1000));
```
- Draw to the `Page`'s `Graphics` object:
```
Graphics gpr = doc.Pages.Last().Graphics;
gpr.FillRectangle(100, 100, 800, 800, Colour.FromRgb(128, 128, 128));
```
- Save as PDF document:
```
using VectSharp.PDF;
//...
doc.SaveAsPDF(@"Sample.pdf");
```
- Export the graphics to a `Canvas`:
```
using VectSharp.Canvas;
//...
Avalonia.Controls.Canvas can = doc.Pages.Last().PaintToCanvas();
```
- Export the graphics to a `Canvas`, using a multi-layer, multi-threaded, triple-buffered renderer based on SkiaSharp (which provides the best performance if you wish e.g. to place the canvas within a `ZoomBorder`):
```
using VectSharp.Canvas;
//...
// A single page
Avalonia.Controls.Canvas can = doc.Pages.Last().PaintToSKCanvas();
// The whole document - each page will correspond to a layer
Avalonia.Controls.Canvas can = doc.PaintToSKCanvas();
```
- Save as a PNG image:
```
using VectSharp.Raster;
//...
doc.Pages.Last().SaveAsPNG(@"Sample.png");
```
- Save as an SVG document:
```
using VectSharp.SVG;
//...
doc.Pages.Last().SaveAsSVG(@"Sample.svg");
```
- PDF and SVG documents support both internal and external links:
```
using VectSharp;
using VectSharp.PDF;
using VectSharp.SVG;
//...
Document document = new Document();
Page page = new Page(1000, 1000);
document.Pages.Add(page);
page.Graphics.FillRectangle(100, 100, 800, 50, Colour.FromRgb(128, 128, 128), tag: "linkToGitHub");
page.Graphics.FillRectangle(100, 300, 800, 50, Colour.FromRgb(255, 0, 0), tag: "linkToBlueRectangle");
page.Graphics.FillRectangle(100, 850, 800, 50, Colour.FromRgb(0, 0, 255), tag: "blueRectangle");
Dictionary<string, string> links = new Dictionary<string, string>() { { "linkToGitHub",
 "https://github.com/" }, { "linkToBlueRectangle", "#blueRectangle" } };
page.SaveAsSVG(@"Links.svg", linkDestinations: links);
document.SaveAsPDF(@"Links.pdf", linkDestinations: links);
```

This code produces a document with three rectangles: the grey one at the top links to the GitHub home page, while the red one in the middle is a hyperlink to the blue one at the bottom. Links in PDF documents can refer to objects that are in a different page than the one containing the link.

The public classes and methods are fully documented, and you can find a (much) more detailed code example in `MainWindow.xaml.cs`. A detailed guide about 3D graphics in VectSharp.ThreeD is available in the `VectSharp.ThreeD` folder.

1.4 Font libraries

Since version 2.0.0, font names are resolved using a "font library". This is a class that implements the `VectSharp.IFontLibrary` interface, providing methods to obtain a `FontFamily` object from a `string` or a `FontFamily.StandardFontFamilies` enumeration. The default font library included in VectSharp uses the embedded fonts (Arimo, Tinos, Cousine) as the standard font families.

In practice, assuming you want to use the default font library, you have the following options to create a `FontFamily` object:

```
using VectSharp;
// ...
FontFamily helvetica = FontFamily.ResolveFontFamily(FontFamily.StandardFontFamilies.Helvetica); // Will
       resolve to the Arimo font family.
FontFamily times = FontFamily.ResolveFontFamily("Times-Roman"); // Will resolve to the Tinos font family.
```

These replace the `FontFamily(string)` and `FontFamily(StandardFontFamilies)` constructors of previous versions of VectSharp. Overloads of this method let you specify a list of "fallback" fonts that will be used if the first font you specify is not available.

If you wish, you can replace the default font library with a different one; this will change the way font families are resolved. For example, after installing the `VectSharp.Fonts.Nimbus` NuGet package, you can do:

```
using VectSharp;
// ...
FontFamily.DefaultFontLibrary = VectSharp.Fonts.Nimbus.Library;
FontFamily helvetica = FontFamily.ResolveFontFamily(FontFamily.StandardFontFamilies.Helvetica); // Will
       resolve to the Nimbus Sans L font family.
FontFamily times = FontFamily.ResolveFontFamily("Times-Roman"); // Will resolve to the Nimbus Roman No 9 L
       font family.
```

This will let you re-enable the fonts that were used in previous versions of VectSharp.

You can also use multiple font libraries in the same project. Again, assuming you have installed the VectSharp.Fonts.Nimbus NuGet package:

```
using VectSharp;
FontFamily helvetica1 = FontFamily.ResolveFontFamily(FontFamily.StandardFontFamilies.Helvetica); // Will
       resolve to the Arimo font family.
FontFamily times1 = FontFamily.ResolveFontFamily("Times-Roman"); // Will resolve to the Tinos font family.
FontFamily helvetica2 = VectSharp.Fonts.Nimbus.ResolveFontFamily(FontFamily.StandardFontFamilies.Helvetica);
       // Will resolve to the Nimbus Sans L font family.
FontFamily times2 = VectSharp.Fonts.Nimbus.ResolveFontFamily("Times-Roman"); // Will resolve to the Nimbus
       Roman No 9 L font family.
```

Finally, you can create your own font library class (which could implement things such as dowloading fonts from Google Fonts, or finding them in the user's system font directory...) by creating a class that implements the `IFont↩Library` interface or that extends the `FontLibrary` class (in this latter case, you get a default implementation for the `ResolveFontFamily` overloads that use a list of fallback fonts).

1.5 Creating new output layers

VectSharp can be easily extended to provide additional output layers. To do so:

1. Create a new class implementing the `IGraphicsContext` interface.
2. Provide an extension method to either the `Page` or `Document` types.
3. Somewhere in the extension method, call the `CopyToIGraphicsContext` method on the `Graphics` object of the `Page`s.
4. Opportunely save or return the rendered result.

1.6 Compiling VectSharp from source

The VectSharp source code includes an example project (*VectSharp.Demo*) presenting how VectSharp can be used to produce graphics.

To be able to compile VectSharp from source, you will need to install the latest `.NET SDK` for your operating system.

You can use `Microsoft Visual Studio` to compile the program. The following instructions will cover compiling VectSharp from the command line, instead.

First of all, you will need to download the VectSharp source code: `VectSharp.tar.gz` and extract it somewhere.

1.6.1 Windows

Open a command-line window in the folder where you have extracted the source code, and type:

```
BuildDemo <Target>
```

Where `<Target>` can be one of `Win-x64`, `Linux-x64` or `Mac-x64` depending on which platform you wish to generate executables for.

In the Release folder and in the appropriate subfolder for the target platform you selected, you will find the compiled program.

1.6.2 macOS and Linux

Open a terminal in the folder where you have extracted the source code, and type:

```
./BuildDemo.sh <Target>
```

Where `<Target>` can be one of `Win-x64`, `Linux-x64` or `Mac-x64` depending on which platform you wish to generate executables for.

In the Release folder and in the appropriate subfolder for the target platform you selected, you will find the compiled program.

If you receive an error about permissions being denied, try typing `chmod +x BuildDemo.sh` first.

1.7 Note about VectSharp.MuPDFUtils and .NET Framework

If you wish to use VectSharp.MuPDFUtils in a .NET Framework project, you will need to manually copy the native MuPDFWrapper library for the platform you are using to the executable directory (this is done automatically if you target .NET core).

One way to obtain the appropriate library files is:

1. Manually download the NuGet package for `MuPFDCore` (click on the "Download package" link on the right).
2. Rename the `.nupkg` file so that it has a `.zip` extension.
3. Extract the zip file.
4. Within the extracted folder, the library files are in the `runtimes/xxx-x64/native/` folder, where `xxx` is either `linux`, `osx` or `win`, depending on the platform you are using.

Make sure you copy the appropriate file to the same folder as the executable!

Chapter 2

Namespace Index

2.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

Namespace Documentation

5.1 VectSharp Namespace Reference

Classes

- class Brush

  Represents a brush used to fill or stroke graphics elements. This could be a solid colour, or a more complicated gradient or pattern.
- struct Colour

  Represents an RGB colour.
- class Colours

  Standard colours.
- class DefaultFontLibrary

  A default font library that resolves standard families using the embedded fonts.
- class DisposableIntPtr

  An IDisposable wrapper around an IntPtr that frees the allocated memory when it is disposed.
- class Document

  Represents a collection of pages.
- class Font

  Represents a typeface with a specific size.
- class FontFamily

  Represents a typeface.
- class FontFamilyCreationException

  An exception that occurs while creating a FontFamily.
- class FontLibrary

  Abstract class with a default implementation of font family fallbacks.
- class FormattedText

  Represents a run of text that should be drawn with the same style.
- class FormattedTextExtensions

  Contains extension methods for collections of FormattedText objects.
- class GradientBrush

  Represents a brush painting with a gradient.
- struct GradientStop

  Represents a colour stop in a gradient.
- class GradientStops

  Represents a read-only list of GradientStops.

- class Graphics

  Represents an abstract drawing surface.
- class GraphicsPath

  Represents a graphics path that can be filled or stroked.
- interface IFontLibrary

  Represents a font library with methods to create FontFamily objects from a string or from FontFamily.StandardFontFamilies.
- interface IGraphicsContext

  This interface should be implemented by classes intended to provide graphics output capability to a Graphics object.
- class LinearGradientBrush

  Represents a brush painting with a linear gradient.
- struct LineDash

  Represents instructions on how to paint a dashed line.
- class Page

  Represents a Graphics object with a width and height.
- struct Point

  Represents a point relative to an origin in the top-left corner.
- class RadialGradientBrush

  Represents a brush painting with a radial gradient.
- class RasterImage

  Represents a raster image, created from raw pixel data. Consider using the derived classes included in the NuGet package "VectSharp.MuPDFUtils" if you need to load a raster image from a file or a Stream.
- class ResourceFontFamily

  Represents a FontFamily created from a resource stream.
- class Segment

  Represents a segment as part of a GraphicsPath.
- class SimpleFontLibrary

  A font library that can be used to cache and resolve font family names.
- struct Size

  Represents the size of an object.
- class SolidColourBrush

  Represents a brush painting with a single solid colour.
- class TrueTypeFile

  Represents a font file in TrueType format. Reference: http://stevehanov.ca/blog/?id=143, https://developer.apple.com/fonts/TrueType-Reference-Manual/, https://docs.↩ microsoft.com/en-us/typography/opentype/spec/
- class UnbalancedStackException

  The exception that is thrown when an unbalanced graphics state stack occurs.

Enumerations

- enum TextBaselines { TextBaselines.Top, TextBaselines.Bottom, TextBaselines.Middle, TextBaselines.Baseline }

  Represent text baselines.
- enum TextAnchors { TextAnchors.Left, TextAnchors.Center, TextAnchors.Right }

  Represents text anchors.
- enum LineCaps { LineCaps.Butt = 0, LineCaps.Round = 1, LineCaps.Square = 2 }

  Represents line caps.
- enum LineJoins { LineJoins.Bevel = 2, LineJoins.Miter = 0, LineJoins.Round = 1 }

  Represents line joining options.

- enum SegmentType { SegmentType.Move, SegmentType.Line, SegmentType.CubicBezier, SegmentType.Arc, SegmentType.Close }

  Types of Segment.
- enum UnbalancedStackActions { UnbalancedStackActions.Throw, UnbalancedStackActions.SilentlyFix, UnbalancedStackActions.Ignore }

  Represents ways to deal with unbalanced graphics state stacks.
- enum Script { Script.Normal, Script.Superscript, Script.Subscript }

  Represents the position of the text.
- enum PixelFormats { PixelFormats.RGB, PixelFormats.RGBA, PixelFormats.BGR, PixelFormats.BGRA }

  Represents the pixel format of a raster image.

5.1.1 Enumeration Type Documentation

5.1.1.1 LineCaps

`enum VectSharp.LineCaps [strong]`

Represents line caps.

Enumerator

Butt	The ends of the line are squared off at the endpoints.
Round	The ends of the lines are rounded.
Square	The ends of the lines are squared off by adding an half square box at each end.

Definition at line 70 of file Enums.cs.

5.1.1.2 LineJoins

`enum VectSharp.LineJoins [strong]`

Represents line joining options.

Enumerator

Bevel	Consecutive segments are joined by straight corners.
Miter	Consecutive segments are joined by extending their outside edges until they meet.
Round	Consecutive segments are joined by arc segments.

Definition at line 91 of file Enums.cs.

5.1.1.3 PixelFormats

enum VectSharp.PixelFormats [strong]

Represents the pixel format of a raster image.

Enumerator

RGB	RGB 24bpp format.
RGBA	RGBA 32bpp format.
BGR	BGR 24bpp format.
BGRA	BGR 32bpp format.

Definition at line 27 of file RasterImage.cs.

5.1.1.4 Script

enum VectSharp.Script [strong]

Represents the position of the text.

Enumerator

Normal	The text is normal text.
Superscript	The text is a superscript.
Subscript	The text is a subscript.

Definition at line 29 of file FormattedText.cs.

5.1.1.5 SegmentType

enum VectSharp.SegmentType [strong]

Types of Segment.

Enumerator

Move	The segment represents a move from the current point to a new point.
Line	The segment represents a straight line from the current point to a new point.
CubicBezier	The segment represents a cubic bezier curve from the current point to a new point.
Arc	The segment represents a circular arc from the current point to a new point.
Close	The segment represents the closing segment of a figure.

Definition at line 151 of file Enums.cs.

5.1.1.6 TextAnchors

enum VectSharp.TextAnchors [strong]

Represents text anchors.

Enumerator

Left	The current coordinate will determine the position of the left side of the text string.
Center	The current coordinate will determine the position of the center of the text string.
Right	The current coordinate will determine the position of the right side of the text string.

Definition at line 49 of file Enums.cs.

5.1.1.7 TextBaselines

enum VectSharp.TextBaselines [strong]

Represent text baselines.

Enumerator

Top	The current vertical coordinate determines where the top of the text string will be placed.
Bottom	The current vertical coordinate determines where the bottom of the text string will be placed.
Middle	The current vertical coordinate determines where the middle of the text string will be placed.
Baseline	The current vertical coordinate determines where the baseline of the text string will be placed.

Definition at line 23 of file Enums.cs.

5.1.1.8 UnbalancedStackActions

enum VectSharp.UnbalancedStackActions [strong]

Represents ways to deal with unbalanced graphics state stacks.

Enumerator

Throw	If the graphics state stack is unbalanced, an exception will be thrown.
SilentlyFix	The graphics state stack will be automatically balanced by adding or removing calls to Graphics.Restore as necessary.
Ignore	No attempt will be made at correcting an unbalanced graphics state stack. This may cause issues with some consumers.

Definition at line 182 of file Enums.cs.

5.2 VectSharp.Canvas Namespace Reference

Classes

- class AvaloniaContextInterpreter

  Contains methods to render a Page to an Avalonia.Controls.Canvas.
- class RenderAction

  Represents a light-weight rendering action.
- class SKMultiLayerRenderCanvas

  Represents a multi-threaded, triple-buffered canvas on which the image is drawn using SkiaSharp.
- class SKRenderAction

  Represents a light-weight rendering action.
- class SKRenderContext

  Represents a page that has been prepared for fast rendering using the SkiaSharp renderer.
- class SKRenderContextInterpreter

  Contains methods to render a Page to an Avalonia.Controls.Canvas using the SkiaSharp renderer.

5.3 VectSharp.Markdown Namespace Reference

Classes

- struct FormattedString

  Represents a string with associated formatting information.
- class HTTPUtils

  Contains utilities to resolve absolute and relative URIs.
- class Margins

  Represents the margins of a page.
- class MarkdownRenderer

  Renders Markdown documents into VectSharp graphics objects.
- class SyntaxHighlighter

  Contains methods to perform syntax highlighting.

5.4 VectSharp.MarkdownCanvas Namespace Reference

Classes

- class MarkdownCanvasControl

  A control to display a Markdown document in an Avalonia application.

5.5 VectSharp.MuPDFUtils Namespace Reference

Classes

- class ImageURIParser

  Provides a method to parse an image URI into a page.
- class RasterImageFile

  A RasterImage created from a file.
- class RasterImageStream

  A RasterImage created from a stream.

5.6 VectSharp.PDF Namespace Reference

Classes

- class PDFContextInterpreter

  Contains methods to render a Document as a PDF document.

5.7 VectSharp.Raster Namespace Reference

Classes

- class Raster

  Contains methods to render a page to a PNG image.

5.8 VectSharp.SVG Namespace Reference

Classes

- class Parser

  Contains methods to read an SVG image file.
- class SVGContextInterpreter

  Contains methods to render a Page as an SVG file.

5.9 VectSharp.ThreeD Namespace Reference

Classes

- class AmbientLightSource

  Represents a uniform ambien light source.
- class AreaLightSource

  Represents a light source emitting light from a circular area.
- class ColourMaterial

  Represents a material that always has the same colour, regardless of light.
- interface ILightSource

  Represents a light source.
- interface IMaterial

  Represents a material used to the determine the appearance of Triangle3DElement.
- interface IScene

  Represents a 3D scene.
- struct LightIntensity

  Represents the intensity of a light source at a particular point.
- class MaskedLightSource

  Represents a point light source with a stencil in front of it.
- class ObjectFactory

  A static class containing methods to create complex 3D objects.
- class ParallelLightSource

  Represents a parallel light source.
- class PhongMaterial

  Represents a material that uses a Phong reflection model to determine the colour of the material based on the light sources that hit it.
- class PointLightSource

  Represents a point light source.
- class Scene

  Represents a 3D scene.
- class SpotlightLightSource

  Represents a conic spotlight.

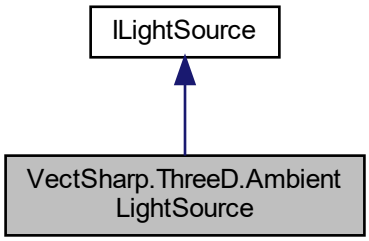
Chapter 6

Class Documentation

6.1 VectSharp.ThreeD.AmbientLightSource Class Reference

Represents a uniform ambien light source.

Inheritance diagram for VectSharp.ThreeD.AmbientLightSource:



Public Member Functions

- AmbientLightSource (double intensity)

  Creates a new AmbientLightSource instance.
- LightIntensity GetLightAt (Point3D point)

  Computes the light intensity at the specified point, without taking into account any obstructions.
- double GetObstruction (Point3D point, IEnumerable< Triangle3DElement > shadowingTriangles)

  Determines the amount of obstruction of the light that results at point due to the specified shadowingTriangles .

Public Attributes

- bool CastsShadow => false

Properties

- double Intensity [get, set]
  The intensity of the light.

6.1.1 Detailed Description

Represents a uniform ambien light source.

Definition at line 91 of file Lights.cs.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 AmbientLightSource()

```
VectSharp.ThreeD.AmbientLightSource.AmbientLightSource (
            double intensity )
```

Creates a new AmbientLightSource instance.

Parameters

intensity	The intensity of the light.

Definition at line 105 of file Lights.cs.

6.1.3 Property Documentation

6.1.3.1 Intensity

```
double VectSharp.ThreeD.AmbientLightSource.Intensity [get], [set]
```

The intensity of the light.

Definition at line 96 of file Lights.cs.

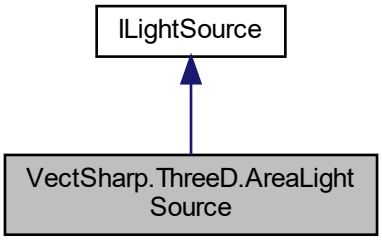
The documentation for this class was generated from the following file:

- VectSharp.ThreeD/Lights.cs

6.2 VectSharp.ThreeD.AreaLightSource Class Reference

Represents a light source emitting light from a circular area.

Inheritance diagram for VectSharp.ThreeD.AreaLightSource:



Public Member Functions

- AreaLightSource (double intensity, Point3D center, double radius, double penumbraRadius, NormalizedVector3D direction, double sourceDistance, int shadowSamplingPointCount)

  Creates a new AreaLightSource instance.
- LightIntensity GetLightAt (Point3D point)

  Computes the light intensity at the specified point, without taking into account any obstructions.
- double GetObstruction (Point3D point, IEnumerable< Triangle3DElement > shadowingTriangles)

  Determines the amount of obstruction of the light that results at point due to the specified shadowingTriangles .

Properties

- bool CastsShadow = true `[get, set]`
- Point3D Center `[get]`

  The centre of the light-emitting area.
- NormalizedVector3D Direction `[get]`

  The direction of the light's main axis, i.e. the normal to the plane containing the light-emitting area.
- double Radius `[get]`

  The radius of the light emitting area.
- double PenumbraRadius `[get]`

  The radius of the penumbra area.
- double Intensity `[get, set]`

  The base intensity of the light.
- double SourceDistance `[get]`

  The distance between the focal point of the light and the light's Center.
- double DistanceAttenuationExponent = 2 `[get, set]`

  An exponent determining how fast the light attenuates with increasing distance. Set to 0 to disable distance attenuation.
- double PenumbraAttenuationExponent = 1 `[get, set]`

  An exponent determining how fast the light attenuates between the light-emitting area radius and the penumbra radius.
- int ShadowSamplingPointCount `[get]`

  The number of points to use when determining the amount of light that is obstructed at a certain point.

6.2.1 Detailed Description

Represents a light source emitting light from a circular area.

Definition at line 579 of file Lights.cs.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 AreaLightSource()

```
VectSharp.ThreeD.AreaLightSource.AreaLightSource (
            double intensity,
            Point3D center,
            double radius,
            double penumbraRadius,
            NormalizedVector3D direction,
            double sourceDistance,
            int shadowSamplingPointCount )
```

Creates a new AreaLightSource instance.

Parameters

intensity	The base intensity of the light.
center	The centre of the light-emitting area.
radius	The radius of the light-emitting area.
penumbraRadius	The radius of the penumbra area.
direction	The direction of the light.
sourceDistance	The distance between the focal point of the light and the light's center.
shadowSamplingPointCount	The number of points to use when determining the amount of light that is obstructed at a certain point.

Definition at line 643 of file Lights.cs.

6.2.3 Property Documentation

6.2.3.1 Center

```
Point3D VectSharp.ThreeD.AreaLightSource.Center  [get]
```

The centre of the light-emitting area.

Definition at line 587 of file Lights.cs.

6.2.3.2 Direction

```
NormalizedVector3D VectSharp.ThreeD.AreaLightSource.Direction  [get]
```

The direction of the light's main axis, i.e. the normal to the plane containing the light-emitting area.

Definition at line 594 of file Lights.cs.

6.2.3.3 DistanceAttenuationExponent

```
double VectSharp.ThreeD.AreaLightSource.DistanceAttenuationExponent = 2  [get], [set]
```

An exponent determining how fast the light attenuates with increasing distance. Set to 0 to disable distance attenuation.

Definition at line 619 of file Lights.cs.

6.2.3.4 Intensity

```
double VectSharp.ThreeD.AreaLightSource.Intensity  [get], [set]
```

The base intensity of the light.

Definition at line 609 of file Lights.cs.

6.2.3.5 PenumbraAttenuationExponent

```
double VectSharp.ThreeD.AreaLightSource.PenumbraAttenuationExponent = 1  [get], [set]
```

An exponent determining how fast the light attenuates between the light-emitting area radius and the penumbra radius.

Definition at line 624 of file Lights.cs.

6.2.3.6 PenumbraRadius

```
double VectSharp.ThreeD.AreaLightSource.PenumbraRadius  [get]
```

The radius of the penumbra area.

Definition at line 604 of file Lights.cs.

6.2.3.7 Radius

```
double VectSharp.ThreeD.AreaLightSource.Radius  [get]
```

The radius of the light emitting area.

Definition at line 599 of file Lights.cs.

6.2.3.8 ShadowSamplingPointCount

```
int VectSharp.ThreeD.AreaLightSource.ShadowSamplingPointCount  [get]
```

The number of points to use when determining the amount of light that is obstructed at a certain point.

Definition at line 629 of file Lights.cs.

6.2.3.9 SourceDistance

```
double VectSharp.ThreeD.AreaLightSource.SourceDistance  [get]
```

The distance between the focal point of the light and the light's Center.

Definition at line 614 of file Lights.cs.

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

- VectSharp.ThreeD/Lights.cs

6.3 VectSharp.Canvas.AvaloniaContextInterpreter Class Reference

Contains methods to render a Page to an Avalonia.Controls.Canvas.

Public Types

- enum TextOptions { TextOptions.AlwaysConvert, TextOptions.ConvertIfNecessary, TextOptions.NeverConvert }

  Defines whether text items should be converted into paths when drawing.

Static Public Member Functions

- static Avalonia.Controls.Canvas PaintToCanvas (this Page page, TextOptions textOption=TextOptions.ConvertIfNecessary)

  Render a Page to an Avalonia.Controls.Canvas.
- static Avalonia.Controls.Canvas PaintToCanvas (this Page page, bool graphicsAsControls, TextOptions textOption=TextOptions.ConvertIfNecessary)

  Render a Page to an Avalonia.Controls.Canvas.
- static Avalonia.Controls.Canvas PaintToCanvas (this Page page, bool graphicsAsControls, Dictionary< string, Delegate > taggedActions, bool removeTaggedActionsAfterExecution=true, TextOptions textOption=TextOptions.ConvertIfNecessary)

  Render a Page to an Avalonia.Controls.Canvas.
- static Avalonia.Controls.Canvas PaintToCanvas (this Page page, Dictionary< string, Delegate > taggedActions, bool removeTaggedActionsAfterExecution=true, TextOptions textOption=TextOptions.ConvertIfNecessary)

  Render a Page to an Avalonia.Controls.Canvas.

6.3.1 Detailed Description

Contains methods to render a Page to an Avalonia.Controls.Canvas.

Definition at line 2166 of file AvaloniaContext.cs.

6.3.2 Member Enumeration Documentation

6.3.2.1 TextOptions

```
enum VectSharp.Canvas.AvaloniaContextInterpreter.TextOptions  [strong]
```

Defines whether text items should be converted into paths when drawing.

Enumerator

AlwaysConvert	Converts all text items into paths.
ConvertIfNecessary	Converts all text items into paths, with the exception of those that use a standard font.
NeverConvert	Does not convert any text items into paths.

Definition at line 2171 of file AvaloniaContext.cs.

6.3.3 Member Function Documentation

6.3.3.1 PaintToCanvas() [1/4]

```
static Avalonia.Controls.Canvas VectSharp.Canvas.AvaloniaContextInterpreter.PaintToCanvas (
            this Page page,
```

```
            bool graphicsAsControls,
            Dictionary< string, Delegate > taggedActions,
            bool removeTaggedActionsAfterExecution = true,
            TextOptions textOption = TextOptions.ConvertIfNecessary )  [static]
```

Render a Page to an Avalonia.Controls.Canvas.

Parameters

page	The Page to render.
graphicsAsControls	If this is true, each graphics object (e.g. paths, text...) is rendered as a separate Avalonia.Controls.Control. Otherwise, they are directly rendered onto the drawing context (which is faster, but does not allow interactivity).
taggedActions	A Dictionary<String, Delegate> containing the Actions that will be performed on items with the corresponding tag. If *graphicsAsControls* is true, the delegates should be voids that accept one parameter of type TextBlock or Path (depending on the tagged item), otherwise, they should accept one parameter of type RenderAction and return an IEnumerable<RenderAction> of the actions that will actually be performed.
removeTaggedActionsAfterExecution	Whether the Actions should be removed from *taggedActions* after their execution. Set to false if the same Action should be performed on multiple items with the same tag.
textOption	Defines whether text items should be converted into paths when drawing.

Returns

An Avalonia.Controls.Canvas containing the rendered graphics objects.

Definition at line 2234 of file AvaloniaContext.cs.

6.3.3.2 PaintToCanvas() [2/4]

```
static Avalonia.Controls.Canvas VectSharp.Canvas.AvaloniaContextInterpreter.PaintToCanvas (
            this Page page,
            bool graphicsAsControls,
            TextOptions textOption = TextOptions.ConvertIfNecessary )  [static]
```

Render a Page to an Avalonia.Controls.Canvas.

Parameters

page	The Page to render.
graphicsAsControls	If this is true, each graphics object (e.g. paths, text...) is rendered as a separate Avalonia.Controls.Control. Otherwise, they are directly rendered onto the drawing context (which is faster, but does not allow interactivity).
textOption	Defines whether text items should be converted into paths when drawing.

Returns

An Avalonia.Controls.Canvas containing the rendered graphics objects.

Definition at line 2210 of file AvaloniaContext.cs.

6.3.3.3 PaintToCanvas() [3/4]

```
static Avalonia.Controls.Canvas VectSharp.Canvas.AvaloniaContextInterpreter.PaintToCanvas (
            this Page page,
            Dictionary< string, Delegate > taggedActions,
            bool removeTaggedActionsAfterExecution = true,
            TextOptions textOption = TextOptions.ConvertIfNecessary )  [static]
```

Render a Page to an Avalonia.Controls.Canvas.

Parameters

page	The Page to render.
taggedActions	A Dictionary<String, Delegate> containing the Actions that will be performed on items with the corresponding tag. The delegates should accept one parameter of type TextBlock or Path (depending on the tagged item).
removeTaggedActionsAfterExecution	Whether the Actions should be removed from *taggedActions* after their execution. Set to false if the same Action should be performed on multiple items with the same tag.
textOption	Defines whether text items should be converted into paths when drawing.

Returns

An Avalonia.Controls.Canvas containing the rendered graphics objects.

Definition at line 2257 of file AvaloniaContext.cs.

6.3.3.4 PaintToCanvas() [4/4]

```
static Avalonia.Controls.Canvas VectSharp.Canvas.AvaloniaContextInterpreter.PaintToCanvas (
            this Page page,
            TextOptions textOption = TextOptions.ConvertIfNecessary )  [static]
```

Render a Page to an Avalonia.Controls.Canvas.

Parameters

page	The Page to render.
textOption	Defines whether text items should be converted into paths when drawing.

Returns

An Avalonia.Controls.Canvas containing the rendered graphics objects.

Definition at line 2195 of file AvaloniaContext.cs.

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

- VectSharp.Canvas/AvaloniaContext.cs

6.4 VectSharp.TrueTypeFile.Bearings Struct Reference

Represents the left- and right-side bearings of a glyph.

Public Attributes

- int LeftSideBearing

  The left-side bearing of the glyph.
- int RightSideBearing

  The right-side bearing of the glyph.

6.4.1 Detailed Description

Represents the left- and right-side bearings of a glyph.

Definition at line 2152 of file TrueType.cs.

6.4.2 Member Data Documentation

6.4.2.1 LeftSideBearing

```
int VectSharp.TrueTypeFile.Bearings.LeftSideBearing
```

The left-side bearing of the glyph.

Definition at line 2157 of file TrueType.cs.

6.4.2.2 RightSideBearing

```
int VectSharp.TrueTypeFile.Bearings.RightSideBearing
```

The right-side bearing of the glyph.

Definition at line 2162 of file TrueType.cs.

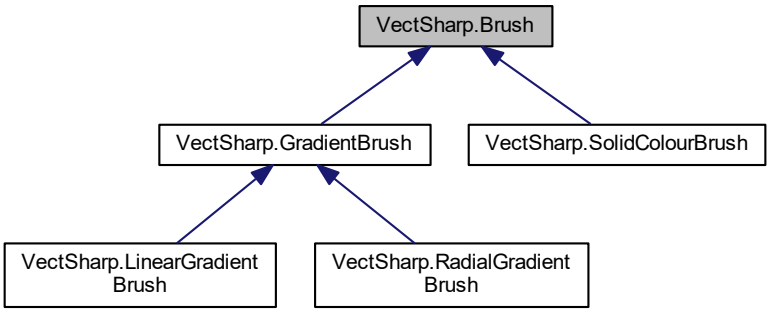
The documentation for this struct was generated from the following file:

- VectSharp/TrueType.cs

6.5 VectSharp.Brush Class Reference

Represents a brush used to fill or stroke graphics elements. This could be a solid colour, or a more complicated gradient or pattern.

Inheritance diagram for VectSharp.Brush:



Public Member Functions

- abstract Brush MultiplyOpacity (double opacity)

  Returns a brush corresponding the current instance, with the specified opacity multiplication applied.

Static Public Member Functions

- static implicit operator Brush (Colour colour)

  Implicitly converts a Colour into a SolidColourBrush.

6.5.1 Detailed Description

Represents a brush used to fill or stroke graphics elements. This could be a solid colour, or a more complicated gradient or pattern.

Definition at line 30 of file Brush.cs.

6.5.2 Member Function Documentation

6.5.2.1 MultiplyOpacity()

```
abstract Brush VectSharp.Brush.MultiplyOpacity (
            double opacity )  [pure virtual]
```

Returns a brush corresponding the current instance, with the specified *opacity* multiplication applied.

Parameters

opacity	The value that will be used to multiply the opacity of the brush.

Returns

A brush corresponding the current instance, with the specified *opacity* multiplication applied.

Implemented in VectSharp.RadialGradientBrush, VectSharp.LinearGradientBrush, and VectSharp.SolidColourBrush.

6.5.2.2 operator Brush()

```
static implicit VectSharp.Brush.operator Brush (
            Colour colour )  [static]
```

Implicitly converts a Colour into a SolidColourBrush.

Parameters

colour	The Colour to use for the brush.

Definition at line 45 of file Brush.cs.

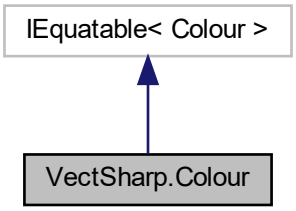
The documentation for this class was generated from the following file:

- VectSharp/Brush.cs

6.6 VectSharp.Colour Struct Reference

Represents an RGB colour.

Inheritance diagram for VectSharp.Colour:



Public Member Functions

- override bool Equals (object obj)
- bool Equals (Colour col)
- override int GetHashCode ()
- string ToCSSString (bool includeAlpha)

  Convert the Colour object into a hex string that is constituted by a "#" followed by two-digit hexadecimal representations of the red, green and blue components of the colour (in the range 0x00 - 0xFF). Optionally also includes opacity (alpha channel) data.
- Colour WithAlpha (double alpha)

  Create a new Colour with the same RGB components as the current Colour, but with the specified alpha .
- Colour WithAlpha (byte alpha)

  Create a new Colour with the same RGB components as the current Colour, but with the specified alpha .
- double double double Z **ToXYZ** ()
- double double double b **ToLab** ()
- double double double L **ToHSL** ()

Static Public Member Functions

- static Colour FromRgb (double r, double g, double b)

  Create a new colour from RGB (red, green and blue) values.
- static Colour FromRgb (byte r, byte g, byte b)

  Create a new colour from RGB (red, green and blue) values.
- static Colour FromRgb (int r, int g, int b)

  Create a new colour from RGB (red, green and blue) values.
- static Colour FromRgba (double r, double g, double b, double a)

  Create a new colour from RGBA (red, green, blue and alpha) values.
- static Colour FromRgba (byte r, byte g, byte b, byte a)

  Create a new colour from RGBA (red, green, blue and alpha) values.
- static Colour FromRgba (byte r, byte g, byte b, double a)

  Create a new colour from RGBA (red, green, blue and alpha) values.
- static Colour FromRgba (int r, int g, int b, int a)

  Create a new colour from RGBA (red, green, blue and alpha) values.
- static Colour FromRgba (int r, int g, int b, double a)

  Create a new colour from RGBA (red, green, blue and alpha) values.

- static Colour FromRgba ((int r, int g, int b, double a) colour)

  Create a new colour from RGBA (red, green, blue and alpha) values.
- static bool operator== (Colour col1, Colour col2)
- static bool operator!= (Colour col1, Colour col2)
- static ? Colour FromCSSString (string cssString)

  Convert a CSS colour string into a Colour object.
- static Colour WithAlpha (Colour original, double alpha)

  Create a new Colour with the same RGB components as the original Colour, but with the specified alpha .
- static Colour WithAlpha (Colour original, byte alpha)

  Create a new Colour with the same RGB components as the original Colour, but with the specified alpha .
- static Colour FromXYZ (double x, double y, double z)

  Creates a Colour from CIE XYZ coordinates.
- static Colour FromLab (double L, double a, double b)

  Creates a Colour from CIE Lab coordinates (under Illuminant D65).
- static Colour FromHSL (double h, double s, double l)

  Creates a Colour from HSL coordinates.

Public Attributes

- double R

  Red component of the colour. Range: [0, 1].
- double G

  Green component of the colour. Range: [0, 1].
- double B

  Blue component of the colour. Range: [0, 1].
- double A

  Alpha component of the colour. Range: [0, 1].
- double X

  Converts a Colour to the CIE XYZ colour space.
- double double **Y**
- double L

  Converts a Colour to the CIE Lab colour space (under Illuminant D65).
- double double **a**
- double H

  Converts a Colour to the HSL colour space.
- double double **S**

6.6.1 Detailed Description

Represents an RGB colour.

Definition at line 25 of file Colour.cs.

6.6.2 Member Function Documentation

6.6.2.1 FromCSSString()

```
static ?  Colour VectSharp.Colour.FromCSSString (
            string cssString )  [static]
```

Convert a CSS colour string into a Colour object.

Parameters

cssString	The CSS colour string. In addition to 148 standard colour names (case-insensitive), #RGB, #RGBA, #RRGGBB and #RRGGBBAA hex strings and rgb(r, g, b) and rgba(r, g, b, a) functional colour notations are supported.

Returns

Definition at line 225 of file Colour.cs.

6.6.2.2 FromHSL()

```
static Colour VectSharp.Colour.FromHSL (
            double h,
            double s,
            double l )  [static]
```

Creates a Colour from HSL coordinates.

Parameters

h	The H component. Should be in range [0, 1].
s	The S component. Should be in range [0, 1].
l	The L component. Should be in range [0, 1].

Returns

A Colour created from the specified components.

Definition at line 575 of file Colour.cs.

6.6.2.3 FromLab()

```
static Colour VectSharp.Colour.FromLab (
            double L,
            double a,
            double b )  [static]
```

Creates a Colour from CIE Lab coordinates (under Illuminant D65).

Parameters

L	The L∗ component.
a	The a∗ component.
b	The b∗ component.

Returns

An sRGB Colour created from the specified components.

Definition at line 497 of file Colour.cs.

6.6.2.4 FromRgb() [1/3]

```
static Colour VectSharp.Colour.FromRgb (
            byte r,
            byte g,
            byte b )  [static]
```

Create a new colour from RGB (red, green and blue) values.

Parameters

r	The red component of the colour. Range: [0, 255].
g	The green component of the colour. Range: [0, 255].
b	The blue component of the colour. Range: [0, 255].

Returns

A Colour struct with the specified components and an alpha component of 1.

Definition at line 74 of file Colour.cs.

6.6.2.5 FromRgb() [2/3]

```
static Colour VectSharp.Colour.FromRgb (
            double r,
            double g,
            double b )  [static]
```

Create a new colour from RGB (red, green and blue) values.

Parameters

r	The red component of the colour. Range: [0, 1].
g	The green component of the colour. Range: [0, 1].
b	The blue component of the colour. Range: [0, 1].

Returns

A Colour struct with the specified components and an alpha component of 1.

Definition at line 62 of file Colour.cs.

6.6.2.6 FromRgb() [3/3]

```
static Colour VectSharp.Colour.FromRgb (
            int r,
            int g,
            int b )  [static]
```

Create a new colour from RGB (red, green and blue) values.

Parameters

r	The red component of the colour. Range: [0, 255].
g	The green component of the colour. Range: [0, 255].
b	The blue component of the colour. Range: [0, 255].

Returns

A Colour struct with the specified components and an alpha component of 1.

Definition at line 86 of file Colour.cs.

6.6.2.7 FromRgba() [1/6]

```
static Colour VectSharp.Colour.FromRgba (
            (int r, int g, int b, double a) colour )  [static]
```

Create a new colour from RGBA (red, green, blue and alpha) values.

Parameters

colour	A ValueTuple<Int32, Int32, Int32, Double> containing component information for the colour. For r, g, and b, range: [0, 255]; for a, range: [0, 1].

Returns

A Colour struct with the specified components.

Definition at line 160 of file Colour.cs.

6.6.2.8 FromRgba() [2/6]

```
static Colour VectSharp.Colour.FromRgba (
            byte r,
            byte g,
            byte b,
            byte a )  [static]
```

Create a new colour from RGBA (red, green, blue and alpha) values.

Parameters

r	The red component of the colour. Range: [0, 255].
g	The green component of the colour. Range: [0, 255].
b	The blue component of the colour. Range: [0, 255].
a	The alpha component of the colour. Range: [0, 255].

Returns

A ColourColour struct with the specified components.

Definition at line 112 of file Colour.cs.

6.6.2.9 FromRgba() [3/6]

```
static Colour VectSharp.Colour.FromRgba (
            byte r,
            byte g,
            byte b,
            double a )  [static]
```

Create a new colour from RGBA (red, green, blue and alpha) values.

Parameters

r	The red component of the colour. Range: [0, 255].
g	The green component of the colour. Range: [0, 255].
b	The blue component of the colour. Range: [0, 255].
a	The alpha component of the colour. Range: [0, 1].

Returns

A Colour struct with the specified components.

Definition at line 125 of file Colour.cs.

6.6.2.10 FromRgba() [4/6]

```
static Colour VectSharp.Colour.FromRgba (
            double r,
            double g,
            double b,
            double a )  [static]
```

Create a new colour from RGBA (red, green, blue and alpha) values.

Parameters

r	The red component of the colour. Range: [0, 1].
g	The green component of the colour. Range: [0, 1].
b	The blue component of the colour. Range: [0, 1].
a	The alpha component of the colour. Range: [0, 1].

Returns

A Colour struct with the specified components.

Definition at line 99 of file Colour.cs.

6.6.2.11 FromRgba() [5/6]

```
static Colour VectSharp.Colour.FromRgba (
            int r,
            int g,
            int b,
            double a )  [static]
```

Create a new colour from RGBA (red, green, blue and alpha) values.

Parameters

r	The red component of the colour. Range: [0, 255].
g	The green component of the colour. Range: [0, 255].
b	The blue component of the colour. Range: [0, 255].
a	The alpha component of the colour. Range: [0, 1].

Returns

A Colour struct with the specified components.

Definition at line 150 of file Colour.cs.

6.6.2.12 FromRgba() [6/6]

```
static Colour VectSharp.Colour.FromRgba (
            int r,
            int g,
            int b,
            int a )  [static]
```

Create a new colour from RGBA (red, green, blue and alpha) values.

Parameters

r	The red component of the colour. Range: [0, 255].
g	The green component of the colour. Range: [0, 255].
b	The blue component of the colour. Range: [0, 255].
a	The alpha component of the colour. Range: [0, 255].

Returns

A Colour struct with the specified components.

Definition at line 137 of file Colour.cs.

6.6.2.13 FromXYZ()

```
static Colour VectSharp.Colour.FromXYZ (
            double x,
            double y,
            double z )  [static]
```

Creates a Colour from CIE XYZ coordinates.

Parameters

x	The X coordinate.
y	The Y coordinate.
z	The Z coordinate.

Returns

An sRGB Colour created from the specified components.

Definition at line 415 of file Colour.cs.

6.6.2.14 ToCSSString()

```
string VectSharp.Colour.ToCSSString (
            bool includeAlpha )
```

Convert the Colour object into a hex string that is constituted by a "#" followed by two-digit hexadecimal representations of the red, green and blue components of the colour (in the range 0x00 - 0xFF). Optionally also includes opacity (alpha channel) data.

Parameters

includeAlpha	Whether two additional hex digits representing the colour's opacity (alpha channel) should be included in the string.

Returns

A hex colour string.

Definition at line 208 of file Colour.cs.

6.6.2.15 WithAlpha() [1/4]

```
Colour VectSharp.Colour.WithAlpha (
            byte alpha )
```

Create a new Colour with the same RGB components as the current Colour, but with the specified *alpha* .

Parameters

alpha	The alpha component of the new Colour.

Returns

A Colour struct with the same RGB components as the current Colour and the specified *alpha* .

Definition at line 361 of file Colour.cs.

6.6.2.16 WithAlpha() [2/4]

```
static Colour VectSharp.Colour.WithAlpha (
            Colour original,
            byte alpha )  [static]
```

Create a new Colour with the same RGB components as the *original* Colour, but with the specified *alpha* .

Parameters

original	The original Colour from which the RGB components will be taken.
alpha	The alpha component of the new Colour.

Returns

A Colour struct with the same RGB components as the *original* Colour and the specified *alpha* .

Definition at line 341 of file Colour.cs.

6.6.2.17 WithAlpha() [3/4]

```
static Colour VectSharp.Colour.WithAlpha (
            Colour original,
            double alpha )  [static]
```

Create a new Colour with the same RGB components as the *original* Colour, but with the specified *alpha* .

Parameters

original	The original Colour from which the RGB components will be taken.
alpha	The alpha component of the new Colour.

Returns

A Colour struct with the same RGB components as the *original* Colour and the specified *alpha* .

Definition at line 330 of file Colour.cs.

6.6.2.18 WithAlpha() [4/4]

```
Colour VectSharp.Colour.WithAlpha (
            double alpha )
```

Create a new Colour with the same RGB components as the current Colour, but with the specified *alpha* .

Parameters

alpha	The alpha component of the new Colour.

Returns

A Colour struct with the same RGB components as the current Colour and the specified *alpha* .

Definition at line 351 of file Colour.cs.

6.6.3 Member Data Documentation

6.6.3.1 A

```
double VectSharp.Colour.A
```

Alpha component of the colour. Range: [0, 1].

Definition at line 45 of file Colour.cs.

6.6.3.2 B

```
double VectSharp.Colour.B
```

Blue component of the colour. Range: [0, 1].

Definition at line 40 of file Colour.cs.

6.6.3.3 G

```
double VectSharp.Colour.G
```

Green component of the colour. Range: [0, 1].

Definition at line 35 of file Colour.cs.

6.6.3.4 H

```
double VectSharp.Colour.H
```

Converts a Colour to the HSL colour space.

Returns

A ValueType containing the H, S and L components of the Colour. Each component has range [0, 1].

Definition at line 528 of file Colour.cs.

6.6.3.5 L

```
double VectSharp.Colour.L
```

Converts a Colour to the CIE Lab colour space (under Illuminant D65).

Returns

A ValueType containing the L∗, a∗ and b∗ components of the Colour.

Definition at line 459 of file Colour.cs.

6.6.3.6 R

```
double VectSharp.Colour.R
```

Red component of the colour. Range: [0, 1].

Definition at line 30 of file Colour.cs.

6.6.3.7 X

```
double VectSharp.Colour.X
```

Converts a Colour to the CIE XYZ colour space.

Returns

A ValueTuple containing the X, Y and Z components of the Colour.

Definition at line 370 of file Colour.cs.

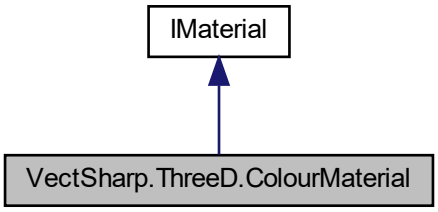
The documentation for this struct was generated from the following files:

- VectSharp/Colour.cs
- VectSharp/StandardColours.cs

6.7 VectSharp.ThreeD.ColourMaterial Class Reference

Represents a material that always has the same colour, regardless of light.

Inheritance diagram for VectSharp.ThreeD.ColourMaterial:



Public Member Functions

- ColourMaterial (Colour colour)

  Creates a new ColourMaterial instance.
- Colour GetColour (Point3D point, NormalizedVector3D surfaceNormal, Camera camera, IList< ILightSource > lights, IList< double > obstructions)

  Obtains the Colour at the specified point.

Properties

- Colour Colour [get]

  The colour of the material.

6.7.1 Detailed Description

Represents a material that always has the same colour, regardless of light.

Definition at line 48 of file Materials.cs.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 ColourMaterial()

```
VectSharp.ThreeD.ColourMaterial.ColourMaterial (
            Colour colour )
```

Creates a new ColourMaterial instance.

Parameters

colour	The colour of the material.

Definition at line 59 of file Materials.cs.

6.7.3 Property Documentation

6.7.3.1 Colour

```
Colour VectSharp.ThreeD.ColourMaterial.Colour  [get]
```

The colour of the material.

Definition at line 53 of file Materials.cs.

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

- VectSharp.ThreeD/Materials.cs

6.8 VectSharp.Colours Class Reference

Standard colours.

Static Public Attributes

- static Colour Black = Colour.FromRgb(0, 0, 0)
  Black #000000
- static Colour Navy = Colour.FromRgb(0, 0, 128)
  Navy #000080
- static Colour DarkBlue = Colour.FromRgb(0, 0, 139)
  DarkBlue #00008B
- static Colour MediumBlue = Colour.FromRgb(0, 0, 205)
  MediumBlue #0000CD
- static Colour Blue = Colour.FromRgb(0, 0, 255)
  Blue #0000FF
- static Colour DarkGreen = Colour.FromRgb(0, 100, 0)
  DarkGreen #006400
- static Colour Green = Colour.FromRgb(0, 128, 0)
  Green #008000
- static Colour Teal = Colour.FromRgb(0, 128, 128)
  Teal #008080
- static Colour DarkCyan = Colour.FromRgb(0, 139, 139)
  DarkCyan #008B8B

- static Colour DeepSkyBlue = Colour.FromRgb(0, 191, 255)

  DeepSkyBlue #00BFFF
- static Colour DarkTurquoise = Colour.FromRgb(0, 206, 209)

  DarkTurquoise #00CED1
- static Colour MediumSpringGreen = Colour.FromRgb(0, 250, 154)

  MediumSpringGreen #00FA9A
- static Colour Lime = Colour.FromRgb(0, 255, 0)

  Lime #00FF00
- static Colour SpringGreen = Colour.FromRgb(0, 255, 127)

  SpringGreen #00FF7F
- static Colour Aqua = Colour.FromRgb(0, 255, 255)

  Aqua #00FFFF
- static Colour Cyan = Colour.FromRgb(0, 255, 255)

  Cyan #00FFFF
- static Colour MidnightBlue = Colour.FromRgb(25, 25, 112)

  MidnightBlue #191970
- static Colour DodgerBlue = Colour.FromRgb(30, 144, 255)

  DodgerBlue #1E90FF
- static Colour LightSeaGreen = Colour.FromRgb(32, 178, 170)

  LightSeaGreen #20B2AA
- static Colour ForestGreen = Colour.FromRgb(34, 139, 34)

  ForestGreen #228B22
- static Colour SeaGreen = Colour.FromRgb(46, 139, 87)

  SeaGreen #2E8B57
- static Colour DarkSlateGray = Colour.FromRgb(47, 79, 79)

  DarkSlateGray #2F4F4F
- static Colour DarkSlateGrey = Colour.FromRgb(47, 79, 79)

  DarkSlateGrey #2F4F4F
- static Colour LimeGreen = Colour.FromRgb(50, 205, 50)

  LimeGreen #32CD32
- static Colour MediumSeaGreen = Colour.FromRgb(60, 179, 113)

  MediumSeaGreen #3CB371
- static Colour Turquoise = Colour.FromRgb(64, 224, 208)

  Turquoise #40E0D0
- static Colour RoyalBlue = Colour.FromRgb(65, 105, 225)

  RoyalBlue #4169E1
- static Colour SteelBlue = Colour.FromRgb(70, 130, 180)

  SteelBlue #4682B4
- static Colour DarkSlateBlue = Colour.FromRgb(72, 61, 139)

  DarkSlateBlue #483D8B
- static Colour MediumTurquoise = Colour.FromRgb(72, 209, 204)

  MediumTurquoise #48D1CC
- static Colour Indigo = Colour.FromRgb(75, 0, 130)

  Indigo #4B0082
- static Colour DarkOliveGreen = Colour.FromRgb(85, 107, 47)

  DarkOliveGreen #556B2F
- static Colour CadetBlue = Colour.FromRgb(95, 158, 160)

  CadetBlue #5F9EA0
- static Colour CornflowerBlue = Colour.FromRgb(100, 149, 237)

  CornflowerBlue #6495ED
- static Colour RebeccaPurple = Colour.FromRgb(102, 51, 153)

RebeccaPurple #663399

- static Colour MediumAquaMarine = Colour.FromRgb(102, 205, 170)

  MediumAquaMarine #66CDAA
- static Colour DimGray = Colour.FromRgb(105, 105, 105)

  DimGray #696969
- static Colour DimGrey = Colour.FromRgb(105, 105, 105)

  DimGrey #696969
- static Colour SlateBlue = Colour.FromRgb(106, 90, 205)

  SlateBlue #6A5ACD
- static Colour OliveDrab = Colour.FromRgb(107, 142, 35)

  OliveDrab #6B8E23
- static Colour SlateGray = Colour.FromRgb(112, 128, 144)

  SlateGray #708090
- static Colour SlateGrey = Colour.FromRgb(112, 128, 144)

  SlateGrey #708090
- static Colour LightSlateGray = Colour.FromRgb(119, 136, 153)

  LightSlateGray #778899
- static Colour LightSlateGrey = Colour.FromRgb(119, 136, 153)

  LightSlateGrey #778899
- static Colour MediumSlateBlue = Colour.FromRgb(123, 104, 238)

  MediumSlateBlue #7B68EE
- static Colour LawnGreen = Colour.FromRgb(124, 252, 0)

  LawnGreen #7CFC00
- static Colour Chartreuse = Colour.FromRgb(127, 255, 0)

  Chartreuse #7FFF00
- static Colour Aquamarine = Colour.FromRgb(127, 255, 212)

  Aquamarine #7FFFD4
- static Colour Maroon = Colour.FromRgb(128, 0, 0)

  Maroon #800000
- static Colour Purple = Colour.FromRgb(128, 0, 128)

  Purple #800080
- static Colour Olive = Colour.FromRgb(128, 128, 0)

  Olive #808000
- static Colour Gray = Colour.FromRgb(128, 128, 128)

  Gray #808080
- static Colour Grey = Colour.FromRgb(128, 128, 128)

  Grey #808080
- static Colour SkyBlue = Colour.FromRgb(135, 206, 235)

  SkyBlue #87CEEB
- static Colour LightSkyBlue = Colour.FromRgb(135, 206, 250)

  LightSkyBlue #87CEFA
- static Colour BlueViolet = Colour.FromRgb(138, 43, 226)

  BlueViolet #8A2BE2
- static Colour DarkRed = Colour.FromRgb(139, 0, 0)

  DarkRed #8B0000
- static Colour DarkMagenta = Colour.FromRgb(139, 0, 139)

  DarkMagenta #8B008B
- static Colour SaddleBrown = Colour.FromRgb(139, 69, 19)

  SaddleBrown #8B4513
- static Colour DarkSeaGreen = Colour.FromRgb(143, 188, 143)

  DarkSeaGreen #8FBC8F

- static Colour LightGreen = Colour.FromRgb(144, 238, 144)

  LightGreen #90EE90
- static Colour MediumPurple = Colour.FromRgb(147, 112, 219)

  MediumPurple #9370DB
- static Colour DarkViolet = Colour.FromRgb(148, 0, 211)

  DarkViolet #9400D3
- static Colour PaleGreen = Colour.FromRgb(152, 251, 152)

  PaleGreen #98FB98
- static Colour DarkOrchid = Colour.FromRgb(153, 50, 204)

  DarkOrchid #9932CC
- static Colour YellowGreen = Colour.FromRgb(154, 205, 50)

  YellowGreen #9ACD32
- static Colour Sienna = Colour.FromRgb(160, 82, 45)

  Sienna #A0522D
- static Colour Brown = Colour.FromRgb(165, 42, 42)

  Brown #A52A2A
- static Colour DarkGray = Colour.FromRgb(169, 169, 169)

  DarkGray #A9A9A9
- static Colour DarkGrey = Colour.FromRgb(169, 169, 169)

  DarkGrey #A9A9A9
- static Colour LightBlue = Colour.FromRgb(173, 216, 230)

  LightBlue #ADD8E6
- static Colour GreenYellow = Colour.FromRgb(173, 255, 47)

  GreenYellow #ADFF2F
- static Colour PaleTurquoise = Colour.FromRgb(175, 238, 238)

  PaleTurquoise #AFEEEE
- static Colour LightSteelBlue = Colour.FromRgb(176, 196, 222)

  LightSteelBlue #B0C4DE
- static Colour PowderBlue = Colour.FromRgb(176, 224, 230)

  PowderBlue #B0E0E6
- static Colour FireBrick = Colour.FromRgb(178, 34, 34)

  FireBrick #B22222
- static Colour DarkGoldenRod = Colour.FromRgb(184, 134, 11)

  DarkGoldenRod #B8860B
- static Colour MediumOrchid = Colour.FromRgb(186, 85, 211)

  MediumOrchid #BA55D3
- static Colour RosyBrown = Colour.FromRgb(188, 143, 143)

  RosyBrown #BC8F8F
- static Colour DarkKhaki = Colour.FromRgb(189, 183, 107)

  DarkKhaki #BDB76B
- static Colour Silver = Colour.FromRgb(192, 192, 192)

  Silver #C0C0C0
- static Colour MediumVioletRed = Colour.FromRgb(199, 21, 133)

  MediumVioletRed #C71585
- static Colour IndianRed = Colour.FromRgb(205, 92, 92)

  IndianRed #CD5C5C
- static Colour Peru = Colour.FromRgb(205, 133, 63)

  Peru #CD853F
- static Colour Chocolate = Colour.FromRgb(210, 105, 30)

  Chocolate #D2691E
- static Colour Tan = Colour.FromRgb(210, 180, 140)

Tan #D2B48C

- static Colour LightGray = Colour.FromRgb(211, 211, 211)

  LightGray #D3D3D3
- static Colour LightGrey = Colour.FromRgb(211, 211, 211)

  LightGrey #D3D3D3
- static Colour Thistle = Colour.FromRgb(216, 191, 216)

  Thistle #D8BFD8
- static Colour Orchid = Colour.FromRgb(218, 112, 214)

  Orchid #DA70D6
- static Colour GoldenRod = Colour.FromRgb(218, 165, 32)

  GoldenRod #DAA520
- static Colour PaleVioletRed = Colour.FromRgb(219, 112, 147)

  PaleVioletRed #DB7093
- static Colour Crimson = Colour.FromRgb(220, 20, 60)

  Crimson #DC143C
- static Colour Gainsboro = Colour.FromRgb(220, 220, 220)

  Gainsboro #DCDCDC
- static Colour Plum = Colour.FromRgb(221, 160, 221)

  Plum #DDA0DD
- static Colour BurlyWood = Colour.FromRgb(222, 184, 135)

  BurlyWood #DEB887
- static Colour LightCyan = Colour.FromRgb(224, 255, 255)

  LightCyan #E0FFFF
- static Colour Lavender = Colour.FromRgb(230, 230, 250)

  Lavender #E6E6FA
- static Colour DarkSalmon = Colour.FromRgb(233, 150, 122)

  DarkSalmon #E9967A
- static Colour Violet = Colour.FromRgb(238, 130, 238)

  Violet #EE82EE
- static Colour PaleGoldenRod = Colour.FromRgb(238, 232, 170)

  PaleGoldenRod #EEE8AA
- static Colour LightCoral = Colour.FromRgb(240, 128, 128)

  LightCoral #F08080
- static Colour Khaki = Colour.FromRgb(240, 230, 140)

  Khaki #F0E68C
- static Colour AliceBlue = Colour.FromRgb(240, 248, 255)

  AliceBlue #F0F8FF
- static Colour HoneyDew = Colour.FromRgb(240, 255, 240)

  HoneyDew #F0FFF0
- static Colour Azure = Colour.FromRgb(240, 255, 255)

  Azure #F0FFFF
- static Colour SandyBrown = Colour.FromRgb(244, 164, 96)

  SandyBrown #F4A460
- static Colour Wheat = Colour.FromRgb(245, 222, 179)

  Wheat #F5DEB3
- static Colour Beige = Colour.FromRgb(245, 245, 220)

  Beige #F5F5DC
- static Colour WhiteSmoke = Colour.FromRgb(245, 245, 245)

  WhiteSmoke #F5F5F5
- static Colour MintCream = Colour.FromRgb(245, 255, 250)

  MintCream #F5FFFA

- static Colour GhostWhite = Colour.FromRgb(248, 248, 255)

  GhostWhite #F8F8FF
- static Colour Salmon = Colour.FromRgb(250, 128, 114)

  Salmon #FA8072
- static Colour AntiqueWhite = Colour.FromRgb(250, 235, 215)

  AntiqueWhite #FAEBD7
- static Colour Linen = Colour.FromRgb(250, 240, 230)

  Linen #FAF0E6
- static Colour LightGoldenRodYellow = Colour.FromRgb(250, 250, 210)

  LightGoldenRodYellow #FAFAD2
- static Colour OldLace = Colour.FromRgb(253, 245, 230)

  OldLace #FDF5E6
- static Colour Red = Colour.FromRgb(255, 0, 0)

  Red #FF0000
- static Colour Fuchsia = Colour.FromRgb(255, 0, 255)

  Fuchsia #FF00FF
- static Colour Magenta = Colour.FromRgb(255, 0, 255)

  Magenta #FF00FF
- static Colour DeepPink = Colour.FromRgb(255, 20, 147)

  DeepPink #FF1493
- static Colour OrangeRed = Colour.FromRgb(255, 69, 0)

  OrangeRed #FF4500
- static Colour Tomato = Colour.FromRgb(255, 99, 71)

  Tomato #FF6347
- static Colour HotPink = Colour.FromRgb(255, 105, 180)

  HotPink #FF69B4
- static Colour Coral = Colour.FromRgb(255, 127, 80)

  Coral #FF7F50
- static Colour DarkOrange = Colour.FromRgb(255, 140, 0)

  DarkOrange #FF8C00
- static Colour LightSalmon = Colour.FromRgb(255, 160, 122)

  LightSalmon #FFA07A
- static Colour Orange = Colour.FromRgb(255, 165, 0)

  Orange #FFA500
- static Colour LightPink = Colour.FromRgb(255, 182, 193)

  LightPink #FFB6C1
- static Colour Pink = Colour.FromRgb(255, 192, 203)

  Pink #FFC0CB
- static Colour Gold = Colour.FromRgb(255, 215, 0)

  Gold #FFD700
- static Colour PeachPuff = Colour.FromRgb(255, 218, 185)

  PeachPuff #FFDAB9
- static Colour NavajoWhite = Colour.FromRgb(255, 222, 173)

  NavajoWhite #FFDEAD
- static Colour Moccasin = Colour.FromRgb(255, 228, 181)

  Moccasin #FFE4B5
- static Colour Bisque = Colour.FromRgb(255, 228, 196)

  Bisque #FFE4C4
- static Colour MistyRose = Colour.FromRgb(255, 228, 225)

  MistyRose #FFE4E1
- static Colour BlanchedAlmond = Colour.FromRgb(255, 235, 205)

BlanchedAlmond #FFEBCD

- static Colour PapayaWhip = Colour.FromRgb(255, 239, 213)

  PapayaWhip #FFEFD5
- static Colour LavenderBlush = Colour.FromRgb(255, 240, 245)

  LavenderBlush #FFF0F5
- static Colour SeaShell = Colour.FromRgb(255, 245, 238)

  SeaShell #FFF5EE
- static Colour Cornsilk = Colour.FromRgb(255, 248, 220)

  Cornsilk #FFF8DC
- static Colour LemonChiffon = Colour.FromRgb(255, 250, 205)

  LemonChiffon #FFFACD
- static Colour FloralWhite = Colour.FromRgb(255, 250, 240)

  FloralWhite #FFFAF0
- static Colour Snow = Colour.FromRgb(255, 250, 250)

  Snow #FFFAFA
- static Colour Yellow = Colour.FromRgb(255, 255, 0)

  Yellow #FFFF00
- static Colour LightYellow = Colour.FromRgb(255, 255, 224)

  LightYellow #FFFFE0
- static Colour Ivory = Colour.FromRgb(255, 255, 240)

  Ivory #FFFFF0
- static Colour White = Colour.FromRgb(255, 255, 255)

  White #FFFFFF

6.8.1 Detailed Description

Standard colours.

Definition at line 182 of file StandardColours.cs.

6.8.2 Member Data Documentation

6.8.2.1 AliceBlue

```
Colour VectSharp.Colours.AliceBlue = Colour.FromRgb(240, 248, 255)  [static]
```

AliceBlue #F0F8FF

Definition at line 599 of file StandardColours.cs.

6.8.2.2 AntiqueWhite

```
Colour VectSharp.Colours.AntiqueWhite = Colour.FromRgb(250, 235, 215)  [static]
```

AntiqueWhite #FAEBD7

Definition at line 639 of file StandardColours.cs.

6.8.2.3 Aqua

```
Colour VectSharp.Colours.Aqua = Colour.FromRgb(0, 255, 255)  [static]
```

Aqua #00FFFF

Definition at line 243 of file StandardColours.cs.

6.8.2.4 Aquamarine

```
Colour VectSharp.Colours.Aquamarine = Colour.FromRgb(127, 255, 212)  [static]
```

Aquamarine #7FFFD4

Definition at line 375 of file StandardColours.cs.

6.8.2.5 Azure

```
Colour VectSharp.Colours.Azure = Colour.FromRgb(240, 255, 255)  [static]
```

Azure #F0FFFF

Definition at line 607 of file StandardColours.cs.

6.8.2.6 Beige

```
Colour VectSharp.Colours.Beige = Colour.FromRgb(245, 245, 220)  [static]
```

Beige #F5F5DC

Definition at line 619 of file StandardColours.cs.

6.8.2.7 Bisque

```
Colour VectSharp.Colours.Bisque = Colour.FromRgb(255, 228, 196)  [static]
```

Bisque #FFE4C4

Definition at line 723 of file StandardColours.cs.

6.8.2.8 Black

```
Colour VectSharp.Colours.Black = Colour.FromRgb(0, 0, 0)  [static]
```

Black #000000

Definition at line 187 of file StandardColours.cs.

6.8.2.9 BlanchedAlmond

```
Colour VectSharp.Colours.BlanchedAlmond = Colour.FromRgb(255, 235, 205)  [static]
```

BlanchedAlmond #FFEBCD

Definition at line 731 of file StandardColours.cs.

6.8.2.10 Blue

```
Colour VectSharp.Colours.Blue = Colour.FromRgb(0, 0, 255)  [static]
```

Blue #0000FF

Definition at line 203 of file StandardColours.cs.

6.8.2.11 BlueViolet

```
Colour VectSharp.Colours.BlueViolet = Colour.FromRgb(138, 43, 226)  [static]
```

BlueViolet #8A2BE2

Definition at line 407 of file StandardColours.cs.

6.8.2.12 Brown

```
Colour VectSharp.Colours.Brown = Colour.FromRgb(165, 42, 42)  [static]
```

Brown #A52A2A

Definition at line 455 of file StandardColours.cs.

6.8.2.13 BurlyWood

```
Colour VectSharp.Colours.BurlyWood = Colour.FromRgb(222, 184, 135)  [static]
```

BurlyWood #DEB887

Definition at line 567 of file StandardColours.cs.

6.8.2.14 CadetBlue

```
Colour VectSharp.Colours.CadetBlue = Colour.FromRgb(95, 158, 160)  [static]
```

CadetBlue #5F9EA0

Definition at line 315 of file StandardColours.cs.

6.8.2.15 Chartreuse

```
Colour VectSharp.Colours.Chartreuse = Colour.FromRgb(127, 255, 0)  [static]
```

Chartreuse #7FFF00

Definition at line 371 of file StandardColours.cs.

6.8.2.16 Chocolate

```
Colour VectSharp.Colours.Chocolate = Colour.FromRgb(210, 105, 30)  [static]
```

Chocolate #D2691E

Definition at line 523 of file StandardColours.cs.

6.8.2.17 Coral

```
Colour VectSharp.Colours.Coral = Colour.FromRgb(255, 127, 80)  [static]
```

Coral #FF7F50

Definition at line 683 of file StandardColours.cs.

6.8.2.18 CornflowerBlue

```
Colour VectSharp.Colours.CornflowerBlue = Colour.FromRgb(100, 149, 237)  [static]
```

CornflowerBlue #6495ED

Definition at line 319 of file StandardColours.cs.

6.8.2.19 Cornsilk

```
Colour VectSharp.Colours.Cornsilk = Colour.FromRgb(255, 248, 220)  [static]
```

Cornsilk #FFF8DC

Definition at line 747 of file StandardColours.cs.

6.8.2.20 Crimson

```
Colour VectSharp.Colours.Crimson = Colour.FromRgb(220, 20, 60)  [static]
```

Crimson #DC143C

Definition at line 555 of file StandardColours.cs.

6.8.2.21 Cyan

```
Colour VectSharp.Colours.Cyan = Colour.FromRgb(0, 255, 255)  [static]
```

Cyan #00FFFF

Definition at line 247 of file StandardColours.cs.

6.8.2.22 DarkBlue

```
Colour VectSharp.Colours.DarkBlue = Colour.FromRgb(0, 0, 139)  [static]
```

DarkBlue #00008B

Definition at line 195 of file StandardColours.cs.

6.8.2.23 DarkCyan

```
Colour VectSharp.Colours.DarkCyan = Colour.FromRgb(0, 139, 139)  [static]
```

DarkCyan #008B8B

Definition at line 219 of file StandardColours.cs.

6.8.2.24 DarkGoldenRod

```
Colour VectSharp.Colours.DarkGoldenRod = Colour.FromRgb(184, 134, 11)  [static]
```

DarkGoldenRod #B8860B

Definition at line 491 of file StandardColours.cs.

6.8.2.25 DarkGray

```
Colour VectSharp.Colours.DarkGray = Colour.FromRgb(169, 169, 169)  [static]
```

DarkGray #A9A9A9

Definition at line 459 of file StandardColours.cs.

6.8.2.26 DarkGreen

```
Colour VectSharp.Colours.DarkGreen = Colour.FromRgb(0, 100, 0)  [static]
```

DarkGreen #006400

Definition at line 207 of file StandardColours.cs.

6.8.2.27 DarkGrey

```
Colour VectSharp.Colours.DarkGrey = Colour.FromRgb(169, 169, 169)  [static]
```

DarkGrey #A9A9A9

Definition at line 463 of file StandardColours.cs.

6.8.2.28 DarkKhaki

```
Colour VectSharp.Colours.DarkKhaki = Colour.FromRgb(189, 183, 107)  [static]
```

DarkKhaki #BDB76B

Definition at line 503 of file StandardColours.cs.

6.8.2.29 DarkMagenta

```
Colour VectSharp.Colours.DarkMagenta = Colour.FromRgb(139, 0, 139)  [static]
```

DarkMagenta #8B008B

Definition at line 415 of file StandardColours.cs.

6.8.2.30 DarkOliveGreen

```
Colour VectSharp.Colours.DarkOliveGreen = Colour.FromRgb(85, 107, 47)  [static]
```

DarkOliveGreen #556B2F

Definition at line 311 of file StandardColours.cs.

6.8.2.31 DarkOrange

```
Colour VectSharp.Colours.DarkOrange = Colour.FromRgb(255, 140, 0)  [static]
```

DarkOrange #FF8C00

Definition at line 687 of file StandardColours.cs.

6.8.2.32 DarkOrchid

```
Colour VectSharp.Colours.DarkOrchid = Colour.FromRgb(153, 50, 204)  [static]
```

DarkOrchid #9932CC

Definition at line 443 of file StandardColours.cs.

6.8.2.33 DarkRed

```
Colour VectSharp.Colours.DarkRed = Colour.FromRgb(139, 0, 0)  [static]
```

DarkRed #8B0000

Definition at line 411 of file StandardColours.cs.

6.8.2.34 DarkSalmon

```
Colour VectSharp.Colours.DarkSalmon = Colour.FromRgb(233, 150, 122)  [static]
```

DarkSalmon #E9967A

Definition at line 579 of file StandardColours.cs.

6.8.2.35 DarkSeaGreen

```
Colour VectSharp.Colours.DarkSeaGreen = Colour.FromRgb(143, 188, 143)  [static]
```

DarkSeaGreen #8FBC8F

Definition at line 423 of file StandardColours.cs.

6.8.2.36 DarkSlateBlue

```
Colour VectSharp.Colours.DarkSlateBlue = Colour.FromRgb(72, 61, 139)  [static]
```

DarkSlateBlue #483D8B

Definition at line 299 of file StandardColours.cs.

6.8.2.37 DarkSlateGray

```
Colour VectSharp.Colours.DarkSlateGray = Colour.FromRgb(47, 79, 79)  [static]
```

DarkSlateGray #2F4F4F

Definition at line 271 of file StandardColours.cs.

6.8.2.38 DarkSlateGrey

```
Colour VectSharp.Colours.DarkSlateGrey = Colour.FromRgb(47, 79, 79)  [static]
```

DarkSlateGrey #2F4F4F

Definition at line 275 of file StandardColours.cs.

6.8.2.39 DarkTurquoise

```
Colour VectSharp.Colours.DarkTurquoise = Colour.FromRgb(0, 206, 209)  [static]
```

DarkTurquoise #00CED1

Definition at line 227 of file StandardColours.cs.

6.8.2.40 DarkViolet

```
Colour VectSharp.Colours.DarkViolet = Colour.FromRgb(148, 0, 211)  [static]
```

DarkViolet #9400D3

Definition at line 435 of file StandardColours.cs.

6.8.2.41 DeepPink

```
Colour VectSharp.Colours.DeepPink = Colour.FromRgb(255, 20, 147)  [static]
```

DeepPink #FF1493

Definition at line 667 of file StandardColours.cs.

6.8.2.42 DeepSkyBlue

```
Colour VectSharp.Colours.DeepSkyBlue = Colour.FromRgb(0, 191, 255)  [static]
```

DeepSkyBlue #00BFFF

Definition at line 223 of file StandardColours.cs.

6.8.2.43 DimGray

```
Colour VectSharp.Colours.DimGray = Colour.FromRgb(105, 105, 105)  [static]
```

DimGray #696969

Definition at line 331 of file StandardColours.cs.

6.8.2.44 DimGrey

```
Colour VectSharp.Colours.DimGrey = Colour.FromRgb(105, 105, 105)  [static]
```

DimGrey #696969

Definition at line 335 of file StandardColours.cs.

6.8.2.45 DodgerBlue

```
Colour VectSharp.Colours.DodgerBlue = Colour.FromRgb(30, 144, 255)  [static]
```

DodgerBlue #1E90FF

Definition at line 255 of file StandardColours.cs.

6.8.2.46 FireBrick

```
Colour VectSharp.Colours.FireBrick = Colour.FromRgb(178, 34, 34)  [static]
```

FireBrick #B22222

Definition at line 487 of file StandardColours.cs.

6.8.2.47 FloralWhite

```
Colour VectSharp.Colours.FloralWhite = Colour.FromRgb(255, 250, 240)  [static]
```

FloralWhite #FFFAF0

Definition at line 755 of file StandardColours.cs.

6.8.2.48 ForestGreen

```
Colour VectSharp.Colours.ForestGreen = Colour.FromRgb(34, 139, 34)  [static]
```

ForestGreen #228B22

Definition at line 263 of file StandardColours.cs.

6.8.2.49 Fuchsia

```
Colour VectSharp.Colours.Fuchsia = Colour.FromRgb(255, 0, 255)  [static]
```

Fuchsia #FF00FF

Definition at line 659 of file StandardColours.cs.

6.8.2.50 Gainsboro

```
Colour VectSharp.Colours.Gainsboro = Colour.FromRgb(220, 220, 220)  [static]
```

Gainsboro #DCDCDC

Definition at line 559 of file StandardColours.cs.

6.8.2.51 GhostWhite

```
Colour VectSharp.Colours.GhostWhite = Colour.FromRgb(248, 248, 255)  [static]
```

GhostWhite #F8F8FF

Definition at line 631 of file StandardColours.cs.

6.8.2.52 Gold

```
Colour VectSharp.Colours.Gold = Colour.FromRgb(255, 215, 0)  [static]
```

Gold #FFD700

Definition at line 707 of file StandardColours.cs.

6.8.2.53 GoldenRod

```
Colour VectSharp.Colours.GoldenRod = Colour.FromRgb(218, 165, 32)  [static]
```

GoldenRod #DAA520

Definition at line 547 of file StandardColours.cs.

6.8.2.54 Gray

```
Colour VectSharp.Colours.Gray = Colour.FromRgb(128, 128, 128)  [static]
```

Gray #808080

Definition at line 391 of file StandardColours.cs.

6.8.2.55 Green

```
Colour VectSharp.Colours.Green = Colour.FromRgb(0, 128, 0)  [static]
```

Green #008000

Definition at line 211 of file StandardColours.cs.

6.8.2.56 GreenYellow

```
Colour VectSharp.Colours.GreenYellow = Colour.FromRgb(173, 255, 47)  [static]
```

GreenYellow #ADFF2F

Definition at line 471 of file StandardColours.cs.

6.8.2.57 Grey

```
Colour VectSharp.Colours.Grey = Colour.FromRgb(128, 128, 128)  [static]
```

Grey #808080

Definition at line 395 of file StandardColours.cs.

6.8.2.58 HoneyDew

```
Colour VectSharp.Colours.HoneyDew = Colour.FromRgb(240, 255, 240)  [static]
```

HoneyDew #F0FFF0

Definition at line 603 of file StandardColours.cs.

6.8.2.59 HotPink

```
Colour VectSharp.Colours.HotPink = Colour.FromRgb(255, 105, 180)  [static]
```

HotPink #FF69B4

Definition at line 679 of file StandardColours.cs.

6.8.2.60 IndianRed

```
Colour VectSharp.Colours.IndianRed = Colour.FromRgb(205, 92, 92)  [static]
```

IndianRed #CD5C5C

Definition at line 515 of file StandardColours.cs.

6.8.2.61 Indigo

```
Colour VectSharp.Colours.Indigo = Colour.FromRgb(75, 0, 130)  [static]
```

Indigo #4B0082

Definition at line 307 of file StandardColours.cs.

6.8.2.62 Ivory

```
Colour VectSharp.Colours.Ivory = Colour.FromRgb(255, 255, 240)  [static]
```

Ivory #FFFFF0

Definition at line 771 of file StandardColours.cs.

6.8.2.63 Khaki

```
Colour VectSharp.Colours.Khaki = Colour.FromRgb(240, 230, 140)  [static]
```

Khaki #F0E68C

Definition at line 595 of file StandardColours.cs.

6.8.2.64 Lavender

```
Colour VectSharp.Colours.Lavender = Colour.FromRgb(230, 230, 250)  [static]
```

Lavender #E6E6FA

Definition at line 575 of file StandardColours.cs.

6.8.2.65 LavenderBlush

```
Colour VectSharp.Colours.LavenderBlush = Colour.FromRgb(255, 240, 245)  [static]
```

LavenderBlush #FFF0F5

Definition at line 739 of file StandardColours.cs.

6.8.2.66 LawnGreen

```
Colour VectSharp.Colours.LawnGreen = Colour.FromRgb(124, 252, 0)  [static]
```

LawnGreen #7CFC00

Definition at line 367 of file StandardColours.cs.

6.8.2.67 LemonChiffon

```
Colour VectSharp.Colours.LemonChiffon = Colour.FromRgb(255, 250, 205)  [static]
```

LemonChiffon #FFFACD

Definition at line 751 of file StandardColours.cs.

6.8.2.68 LightBlue

```
Colour VectSharp.Colours.LightBlue = Colour.FromRgb(173, 216, 230)  [static]
```

LightBlue #ADD8E6

Definition at line 467 of file StandardColours.cs.

6.8.2.69 LightCoral

```
Colour VectSharp.Colours.LightCoral = Colour.FromRgb(240, 128, 128)  [static]
```

LightCoral #F08080

Definition at line 591 of file StandardColours.cs.

6.8.2.70 LightCyan

```
Colour VectSharp.Colours.LightCyan = Colour.FromRgb(224, 255, 255)  [static]
```

LightCyan #E0FFFF

Definition at line 571 of file StandardColours.cs.

6.8.2.71 LightGoldenRodYellow

```
Colour VectSharp.Colours.LightGoldenRodYellow = Colour.FromRgb(250, 250, 210)  [static]
```

LightGoldenRodYellow #FAFAD2

Definition at line 647 of file StandardColours.cs.

6.8.2.72 LightGray

```
Colour VectSharp.Colours.LightGray = Colour.FromRgb(211, 211, 211)  [static]
```

LightGray #D3D3D3

Definition at line 531 of file StandardColours.cs.

6.8.2.73 LightGreen

```
Colour VectSharp.Colours.LightGreen = Colour.FromRgb(144, 238, 144)  [static]
```

LightGreen #90EE90

Definition at line 427 of file StandardColours.cs.

6.8.2.74 LightGrey

```
Colour VectSharp.Colours.LightGrey = Colour.FromRgb(211, 211, 211)  [static]
```

LightGrey #D3D3D3

Definition at line 535 of file StandardColours.cs.

6.8.2.75 LightPink

```
Colour VectSharp.Colours.LightPink = Colour.FromRgb(255, 182, 193)  [static]
```

LightPink #FFB6C1

Definition at line 699 of file StandardColours.cs.

6.8.2.76 LightSalmon

```
Colour VectSharp.Colours.LightSalmon = Colour.FromRgb(255, 160, 122)  [static]
```

LightSalmon #FFA07A

Definition at line 691 of file StandardColours.cs.

6.8.2.77 LightSeaGreen

```
Colour VectSharp.Colours.LightSeaGreen = Colour.FromRgb(32, 178, 170)  [static]
```

LightSeaGreen #20B2AA

Definition at line 259 of file StandardColours.cs.

6.8.2.78 LightSkyBlue

```
Colour VectSharp.Colours.LightSkyBlue = Colour.FromRgb(135, 206, 250)  [static]
```

LightSkyBlue #87CEFA

Definition at line 403 of file StandardColours.cs.

6.8.2.79 LightSlateGray

```
Colour VectSharp.Colours.LightSlateGray = Colour.FromRgb(119, 136, 153)  [static]
```

LightSlateGray #778899

Definition at line 355 of file StandardColours.cs.

6.8.2.80 LightSlateGrey

```
Colour VectSharp.Colours.LightSlateGrey = Colour.FromRgb(119, 136, 153)  [static]
```

LightSlateGrey #778899

Definition at line 359 of file StandardColours.cs.

6.8.2.81 LightSteelBlue

```
Colour VectSharp.Colours.LightSteelBlue = Colour.FromRgb(176, 196, 222)  [static]
```

LightSteelBlue #B0C4DE

Definition at line 479 of file StandardColours.cs.

6.8.2.82 LightYellow

```
Colour VectSharp.Colours.LightYellow = Colour.FromRgb(255, 255, 224)  [static]
```

LightYellow #FFFFE0

Definition at line 767 of file StandardColours.cs.

6.8.2.83 Lime

```
Colour VectSharp.Colours.Lime = Colour.FromRgb(0, 255, 0)  [static]
```

Lime #00FF00

Definition at line 235 of file StandardColours.cs.

6.8.2.84 LimeGreen

```
Colour VectSharp.Colours.LimeGreen = Colour.FromRgb(50, 205, 50)  [static]
```

LimeGreen #32CD32

Definition at line 279 of file StandardColours.cs.

6.8.2.85 Linen

```
Colour VectSharp.Colours.Linen = Colour.FromRgb(250, 240, 230)  [static]
```

Linen #FAF0E6

Definition at line 643 of file StandardColours.cs.

6.8.2.86 Magenta

```
Colour VectSharp.Colours.Magenta = Colour.FromRgb(255, 0, 255)  [static]
```

Magenta #FF00FF

Definition at line 663 of file StandardColours.cs.

6.8.2.87 Maroon

```
Colour VectSharp.Colours.Maroon = Colour.FromRgb(128, 0, 0)  [static]
```

Maroon #800000

Definition at line 379 of file StandardColours.cs.

6.8.2.88 MediumAquaMarine

```
Colour VectSharp.Colours.MediumAquaMarine = Colour.FromRgb(102, 205, 170)  [static]
```

MediumAquaMarine #66CDAA

Definition at line 327 of file StandardColours.cs.

6.8.2.89 MediumBlue

```
Colour VectSharp.Colours.MediumBlue = Colour.FromRgb(0, 0, 205)  [static]
```

MediumBlue #0000CD

Definition at line 199 of file StandardColours.cs.

6.8.2.90 MediumOrchid

```
Colour VectSharp.Colours.MediumOrchid = Colour.FromRgb(186, 85, 211)  [static]
```

MediumOrchid #BA55D3

Definition at line 495 of file StandardColours.cs.

6.8.2.91 MediumPurple

```
Colour VectSharp.Colours.MediumPurple = Colour.FromRgb(147, 112, 219)  [static]
```

MediumPurple #9370DB

Definition at line 431 of file StandardColours.cs.

6.8.2.92 MediumSeaGreen

```
Colour VectSharp.Colours.MediumSeaGreen = Colour.FromRgb(60, 179, 113)  [static]
```

MediumSeaGreen #3CB371

Definition at line 283 of file StandardColours.cs.

6.8.2.93 MediumSlateBlue

```
Colour VectSharp.Colours.MediumSlateBlue = Colour.FromRgb(123, 104, 238)  [static]
```

MediumSlateBlue #7B68EE

Definition at line 363 of file StandardColours.cs.

6.8.2.94 MediumSpringGreen

```
Colour VectSharp.Colours.MediumSpringGreen = Colour.FromRgb(0, 250, 154)  [static]
```

MediumSpringGreen #00FA9A

Definition at line 231 of file StandardColours.cs.

6.8.2.95 MediumTurquoise

```
Colour VectSharp.Colours.MediumTurquoise = Colour.FromRgb(72, 209, 204)  [static]
```

MediumTurquoise #48D1CC

Definition at line 303 of file StandardColours.cs.

6.8.2.96 MediumVioletRed

```
Colour VectSharp.Colours.MediumVioletRed = Colour.FromRgb(199, 21, 133)  [static]
```

MediumVioletRed #C71585

Definition at line 511 of file StandardColours.cs.

6.8.2.97 MidnightBlue

```
Colour VectSharp.Colours.MidnightBlue = Colour.FromRgb(25, 25, 112)  [static]
```

MidnightBlue #191970

Definition at line 251 of file StandardColours.cs.

6.8.2.98 MintCream

```
Colour VectSharp.Colours.MintCream = Colour.FromRgb(245, 255, 250)  [static]
```

MintCream #F5FFFA

Definition at line 627 of file StandardColours.cs.

6.8.2.99 MistyRose

```
Colour VectSharp.Colours.MistyRose = Colour.FromRgb(255, 228, 225)  [static]
```

MistyRose #FFE4E1

Definition at line 727 of file StandardColours.cs.

6.8.2.100 Moccasin

```
Colour VectSharp.Colours.Moccasin = Colour.FromRgb(255, 228, 181)  [static]
```

Moccasin #FFE4B5

Definition at line 719 of file StandardColours.cs.

6.8.2.101 NavajoWhite

```
Colour VectSharp.Colours.NavajoWhite = Colour.FromRgb(255, 222, 173)  [static]
```

NavajoWhite #FFDEAD

Definition at line 715 of file StandardColours.cs.

6.8.2.102 Navy

```
Colour VectSharp.Colours.Navy = Colour.FromRgb(0, 0, 128)  [static]
```

Navy #000080

Definition at line 191 of file StandardColours.cs.

6.8.2.103 OldLace

```
Colour VectSharp.Colours.OldLace = Colour.FromRgb(253, 245, 230)  [static]
```

OldLace #FDF5E6

Definition at line 651 of file StandardColours.cs.

6.8.2.104 Olive

```
Colour VectSharp.Colours.Olive = Colour.FromRgb(128, 128, 0)  [static]
```

Olive #808000

Definition at line 387 of file StandardColours.cs.

6.8.2.105 OliveDrab

```
Colour VectSharp.Colours.OliveDrab = Colour.FromRgb(107, 142, 35)  [static]
```

OliveDrab #6B8E23

Definition at line 343 of file StandardColours.cs.

6.8.2.106 Orange

```
Colour VectSharp.Colours.Orange = Colour.FromRgb(255, 165, 0)  [static]
```

Orange #FFA500

Definition at line 695 of file StandardColours.cs.

6.8.2.107 OrangeRed

```
Colour VectSharp.Colours.OrangeRed = Colour.FromRgb(255, 69, 0)  [static]
```

OrangeRed #FF4500

Definition at line 671 of file StandardColours.cs.

6.8.2.108 Orchid

```
Colour VectSharp.Colours.Orchid = Colour.FromRgb(218, 112, 214)  [static]
```

Orchid #DA70D6

Definition at line 543 of file StandardColours.cs.

6.8.2.109 PaleGoldenRod

```
Colour VectSharp.Colours.PaleGoldenRod = Colour.FromRgb(238, 232, 170)  [static]
```

PaleGoldenRod #EEE8AA

Definition at line 587 of file StandardColours.cs.

6.8.2.110 PaleGreen

```
Colour VectSharp.Colours.PaleGreen = Colour.FromRgb(152, 251, 152)  [static]
```

PaleGreen #98FB98

Definition at line 439 of file StandardColours.cs.

6.8.2.111 PaleTurquoise

```
Colour VectSharp.Colours.PaleTurquoise = Colour.FromRgb(175, 238, 238)  [static]
```

PaleTurquoise #AFEEEE

Definition at line 475 of file StandardColours.cs.

6.8.2.112 PaleVioletRed

```
Colour VectSharp.Colours.PaleVioletRed = Colour.FromRgb(219, 112, 147)  [static]
```

PaleVioletRed #DB7093

Definition at line 551 of file StandardColours.cs.

6.8.2.113 PapayaWhip

```
Colour VectSharp.Colours.PapayaWhip = Colour.FromRgb(255, 239, 213)  [static]
```

PapayaWhip #FFEFD5

Definition at line 735 of file StandardColours.cs.

6.8.2.114 PeachPuff

```
Colour VectSharp.Colours.PeachPuff = Colour.FromRgb(255, 218, 185)  [static]
```

PeachPuff #FFDAB9

Definition at line 711 of file StandardColours.cs.

6.8.2.115 Peru

```
Colour VectSharp.Colours.Peru = Colour.FromRgb(205, 133, 63)  [static]
```

Peru #CD853F

Definition at line 519 of file StandardColours.cs.

6.8.2.116 Pink

```
Colour VectSharp.Colours.Pink = Colour.FromRgb(255, 192, 203)  [static]
```

Pink #FFC0CB

Definition at line 703 of file StandardColours.cs.

6.8.2.117 Plum

```
Colour VectSharp.Colours.Plum = Colour.FromRgb(221, 160, 221)  [static]
```

Plum #DDA0DD

Definition at line 563 of file StandardColours.cs.

6.8.2.118 PowderBlue

```
Colour VectSharp.Colours.PowderBlue = Colour.FromRgb(176, 224, 230)  [static]
```

PowderBlue #B0E0E6

Definition at line 483 of file StandardColours.cs.

6.8.2.119 Purple

```
Colour VectSharp.Colours.Purple = Colour.FromRgb(128, 0, 128)  [static]
```

Purple #800080

Definition at line 383 of file StandardColours.cs.

6.8.2.120 RebeccaPurple

```
Colour VectSharp.Colours.RebeccaPurple = Colour.FromRgb(102, 51, 153)  [static]
```

RebeccaPurple #663399

Definition at line 323 of file StandardColours.cs.

6.8.2.121 Red

```
Colour VectSharp.Colours.Red = Colour.FromRgb(255, 0, 0)  [static]
```

Red #FF0000

Definition at line 655 of file StandardColours.cs.

6.8.2.122 RosyBrown

```
Colour VectSharp.Colours.RosyBrown = Colour.FromRgb(188, 143, 143)  [static]
```

RosyBrown #BC8F8F

Definition at line 499 of file StandardColours.cs.

6.8.2.123 RoyalBlue

```
Colour VectSharp.Colours.RoyalBlue = Colour.FromRgb(65, 105, 225)  [static]
```

RoyalBlue #4169E1

Definition at line 291 of file StandardColours.cs.

6.8.2.124 SaddleBrown

```
Colour VectSharp.Colours.SaddleBrown = Colour.FromRgb(139, 69, 19)  [static]
```

SaddleBrown #8B4513

Definition at line 419 of file StandardColours.cs.

6.8.2.125 Salmon

```
Colour VectSharp.Colours.Salmon = Colour.FromRgb(250, 128, 114)  [static]
```

Salmon #FA8072

Definition at line 635 of file StandardColours.cs.

6.8.2.126 SandyBrown

```
Colour VectSharp.Colours.SandyBrown = Colour.FromRgb(244, 164, 96)  [static]
```

SandyBrown #F4A460

Definition at line 611 of file StandardColours.cs.

6.8.2.127 SeaGreen

```
Colour VectSharp.Colours.SeaGreen = Colour.FromRgb(46, 139, 87)  [static]
```

SeaGreen #2E8B57

Definition at line 267 of file StandardColours.cs.

6.8.2.128 SeaShell

```
Colour VectSharp.Colours.SeaShell = Colour.FromRgb(255, 245, 238)  [static]
```

SeaShell #FFF5EE

Definition at line 743 of file StandardColours.cs.

6.8.2.129 Sienna

```
Colour VectSharp.Colours.Sienna = Colour.FromRgb(160, 82, 45)  [static]
```

Sienna #A0522D

Definition at line 451 of file StandardColours.cs.

6.8.2.130 Silver

```
Colour VectSharp.Colours.Silver = Colour.FromRgb(192, 192, 192)  [static]
```

Silver #C0C0C0

Definition at line 507 of file StandardColours.cs.

6.8.2.131 SkyBlue

```
Colour VectSharp.Colours.SkyBlue = Colour.FromRgb(135, 206, 235)  [static]
```

SkyBlue #87CEEB

Definition at line 399 of file StandardColours.cs.

6.8.2.132 SlateBlue

```
Colour VectSharp.Colours.SlateBlue = Colour.FromRgb(106, 90, 205)  [static]
```

SlateBlue #6A5ACD

Definition at line 339 of file StandardColours.cs.

6.8.2.133 SlateGray

```
Colour VectSharp.Colours.SlateGray = Colour.FromRgb(112, 128, 144)  [static]
```

SlateGray #708090

Definition at line 347 of file StandardColours.cs.

6.8.2.134 SlateGrey

```
Colour VectSharp.Colours.SlateGrey = Colour.FromRgb(112, 128, 144)  [static]
```

SlateGrey #708090

Definition at line 351 of file StandardColours.cs.

6.8.2.135 Snow

```
Colour VectSharp.Colours.Snow = Colour.FromRgb(255, 250, 250)  [static]
```

Snow #FFFAFA

Definition at line 759 of file StandardColours.cs.

6.8.2.136 SpringGreen

```
Colour VectSharp.Colours.SpringGreen = Colour.FromRgb(0, 255, 127)  [static]
```

SpringGreen #00FF7F

Definition at line 239 of file StandardColours.cs.

6.8.2.137 SteelBlue

```
Colour VectSharp.Colours.SteelBlue = Colour.FromRgb(70, 130, 180)  [static]
```

SteelBlue #4682B4

Definition at line 295 of file StandardColours.cs.

6.8.2.138 Tan

```
Colour VectSharp.Colours.Tan = Colour.FromRgb(210, 180, 140)  [static]
```

Tan #D2B48C

Definition at line 527 of file StandardColours.cs.

6.8.2.139 Teal

```
Colour VectSharp.Colours.Teal = Colour.FromRgb(0, 128, 128)  [static]
```

Teal #008080

Definition at line 215 of file StandardColours.cs.

6.8.2.140 Thistle

```
Colour VectSharp.Colours.Thistle = Colour.FromRgb(216, 191, 216)  [static]
```

Thistle #D8BFD8

Definition at line 539 of file StandardColours.cs.

6.8.2.141 Tomato

```
Colour VectSharp.Colours.Tomato = Colour.FromRgb(255, 99, 71)  [static]
```

Tomato #FF6347

Definition at line 675 of file StandardColours.cs.

6.8.2.142 Turquoise

```
Colour VectSharp.Colours.Turquoise = Colour.FromRgb(64, 224, 208)  [static]
```

Turquoise #40E0D0

Definition at line 287 of file StandardColours.cs.

6.8.2.143 Violet

```
Colour VectSharp.Colours.Violet = Colour.FromRgb(238, 130, 238)  [static]
```

Violet #EE82EE

Definition at line 583 of file StandardColours.cs.

6.8.2.144 Wheat

```
Colour VectSharp.Colours.Wheat = Colour.FromRgb(245, 222, 179)  [static]
```

Wheat #F5DEB3

Definition at line 615 of file StandardColours.cs.

6.8.2.145 White

```
Colour VectSharp.Colours.White = Colour.FromRgb(255, 255, 255)  [static]
```

White #FFFFFF

Definition at line 775 of file StandardColours.cs.

6.8.2.146 WhiteSmoke

```
Colour VectSharp.Colours.WhiteSmoke = Colour.FromRgb(245, 245, 245)  [static]
```

WhiteSmoke #F5F5F5

Definition at line 623 of file StandardColours.cs.

6.8.2.147 Yellow

```
Colour VectSharp.Colours.Yellow = Colour.FromRgb(255, 255, 0)  [static]
```

Yellow #FFFF00

Definition at line 763 of file StandardColours.cs.

6.8.2.148 YellowGreen

```
Colour VectSharp.Colours.YellowGreen = Colour.FromRgb(154, 205, 50)  [static]
```

YellowGreen #9ACD32

Definition at line 447 of file StandardColours.cs.

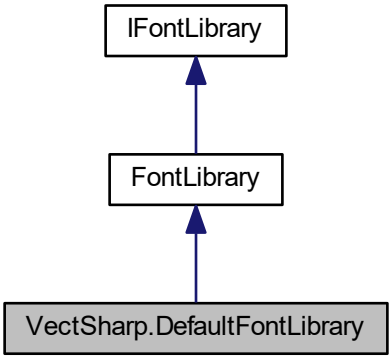
The documentation for this class was generated from the following file:

- VectSharp/StandardColours.cs

6.9 VectSharp.DefaultFontLibrary Class Reference

A default font library that resolves standard families using the embedded fonts.

Inheritance diagram for VectSharp.DefaultFontLibrary:



Public Member Functions

- override FontFamily ResolveFontFamily (string fontFamily)

  Create a new font family from the specified family name or true type file. If the family name or the true type file are not valid, an exception might be raised.
- override FontFamily ResolveFontFamily (FontFamily.StandardFontFamilies standardFontFamily)

  Create a new font family from the specified standard font family name.

6.9.1 Detailed Description

A default font library that resolves standard families using the embedded fonts.

Definition at line 461 of file FontLibrary.cs.

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

- VectSharp/FontLibrary.cs

6.10 VectSharp.Font.DetailedFontMetrics Class Reference

Represents detailed information about the metrics of a text string when drawn with a certain font.

Properties

- double Width `[get]`

  Width of the text (measured on the actual glyph outlines).
- double Height `[get]`

  Height of the text (measured on the actual glyph outlines).
- double LeftSideBearing `[get]`

  How much the leftmost glyph in the string overhangs the glyph origin on the left. Positive for glyphs that hang past the origin (e.g. italic 'f').
- double RightSideBearing `[get]`

  How much the rightmost glyph in the string overhangs the glyph end on the right. Positive for glyphs that hang past the end (e.g. italic 'f').
- double Top `[get]`

  Height of the tallest glyph in the string over the baseline. Always >= 0.
- double Bottom `[get]`

  Depth of the deepest glyph in the string below the baseline. Always <= 0.

6.10.1 Detailed Description

Represents detailed information about the metrics of a text string when drawn with a certain font.

Definition at line 33 of file Font.cs.

6.10.2 Property Documentation

6.10.2.1 Bottom

```
double VectSharp.Font.DetailedFontMetrics.Bottom [get]
```

Depth of the deepest glyph in the string below the baseline. Always <= 0.

Definition at line 63 of file Font.cs.

6.10.2.2 Height

```
double VectSharp.Font.DetailedFontMetrics.Height  [get]
```

Height of the text (measured on the actual glyph outlines).

Definition at line 43 of file Font.cs.

6.10.2.3 LeftSideBearing

```
double VectSharp.Font.DetailedFontMetrics.LeftSideBearing  [get]
```

How much the leftmost glyph in the string overhangs the glyph origin on the left. Positive for glyphs that hang past the origin (e.g. italic 'f').

Definition at line 48 of file Font.cs.

6.10.2.4 RightSideBearing

```
double VectSharp.Font.DetailedFontMetrics.RightSideBearing  [get]
```

How much the rightmost glyph in the string overhangs the glyph end on the right. Positive for glyphs that hang past the end (e.g. italic 'f').

Definition at line 53 of file Font.cs.

6.10.2.5 Top

```
double VectSharp.Font.DetailedFontMetrics.Top  [get]
```

Height of the tallest glyph in the string over the baseline. Always >= 0.

Definition at line 58 of file Font.cs.

6.10.2.6 Width

```
double VectSharp.Font.DetailedFontMetrics.Width  [get]
```

Width of the text (measured on the actual glyph outlines).

Definition at line 38 of file Font.cs.

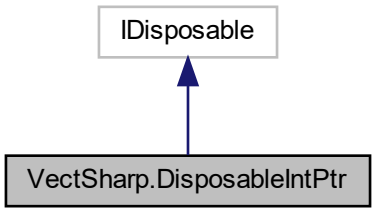
The documentation for this class was generated from the following file:

- VectSharp/Font.cs

6.11 VectSharp.DisposableIntPtr Class Reference

An IDisposable wrapper around an IntPtr that frees the allocated memory when it is disposed.

Inheritance diagram for VectSharp.DisposableIntPtr:



Public Member Functions

- DisposableIntPtr (IntPtr pointer)
  Create a new DisposableIntPtr.
- void **Dispose** ()

Public Attributes

- readonly IntPtr InternalPointer
  The pointer to the unmanaged memory.

6.11.1 Detailed Description

An IDisposable wrapper around an IntPtr that frees the allocated memory when it is disposed.

Definition at line 53 of file RasterImage.cs.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 DisposableIntPtr()

```
VectSharp.DisposableIntPtr.DisposableIntPtr (
            IntPtr pointer )
```

Create a new DisposableIntPtr.

Parameters

pointer	The pointer that should be freed upon disposing of this object.

Definition at line 64 of file RasterImage.cs.

6.11.3 Member Data Documentation

6.11.3.1 InternalPointer

```
readonly IntPtr VectSharp.DisposableIntPtr.InternalPointer
```

The pointer to the unmanaged memory.

Definition at line 58 of file RasterImage.cs.

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

- VectSharp/RasterImage.cs

6.12 VectSharp.Document Class Reference

Represents a collection of pages.

Public Member Functions

- Document ()
  Create a new document.

Public Attributes

- List< Page > Pages = new List<Page>()
  The pages in the document.

6.12.1 Detailed Description

Represents a collection of pages.

Definition at line 27 of file Document.cs.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 Document()

```
VectSharp.Document.Document ( )
```

Create a new document.

Definition at line 38 of file Document.cs.

6.12.3 Member Data Documentation

6.12.3.1 Pages

```
List<Page> VectSharp.Document.Pages = new List<Page>()
```

The pages in the document.

Definition at line 32 of file Document.cs.

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

- VectSharp/Document.cs

6.13 VectSharp.Font Class Reference

Represents a typeface with a specific size.

Classes

- class DetailedFontMetrics

  Represents detailed information about the metrics of a text string when drawn with a certain font.

Public Member Functions

- Font (FontFamily fontFamily, double fontSize)

  Create a new Font object, given the base typeface and the font size.
- Size MeasureText (string text)

  Measure the size of a text string when typeset with this font.
- DetailedFontMetrics MeasureTextAdvanced (string text)

  Measure all the metrics of a text string when typeset with this font.

Properties

- double FontSize [get]

  Font size, in graphics units.
- FontFamily FontFamily [get]

  Font typeface.
- double Ascent [get]

  Maximum height over the baseline of the usual glyphs in the font (there may be glyphs taller than this). Always >= 0.
- double WinAscent [get]

  Height above the baseline for a clipping region (Windows ascent). Always >= 0.
- double Descent [get]

  Maximum depth below the baseline of the usual glyphs in the font (there may be glyphs deeper than this). Always <= 0.
- double YMax [get]

  Absolute maximum height over the baseline of the glyphs in the font. Always >= 0.
- double YMin [get]

  Absolute maximum depth below the baseline of the glyphs in the font. Always <= 0.

6.13.1 Detailed Description

Represents a typeface with a specific size.

Definition at line 28 of file Font.cs.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 Font()

```
VectSharp.Font.Font (
            FontFamily fontFamily,
            double fontSize )
```

Create a new Font object, given the base typeface and the font size.

Parameters

fontFamily	Base typeface. See FontFamily.
fontSize	The font size, in graphics units.

Definition at line 91 of file Font.cs.

6.13.3 Member Function Documentation

6.13.3.1 MeasureText()

```
Size VectSharp.Font.MeasureText (
            string text )
```

Measure the size of a text string when typeset with this font.

Parameters

text	The string to measure.

Returns

A Size object representing the width and height of the text.

Definition at line 192 of file Font.cs.

6.13.3.2 MeasureTextAdvanced()

```
DetailedFontMetrics VectSharp.Font.MeasureTextAdvanced (
            string text )
```

Measure all the metrics of a text string when typeset with this font.

Parameters

text	The string to measure.

Returns

A DetailedFontMetrics object representing the metrics of the text.

Definition at line 225 of file Font.cs.

6.13.4 Property Documentation

6.13.4.1 Ascent

```
double VectSharp.Font.Ascent  [get]
```

Maximum height over the baseline of the usual glyphs in the font (there may be glyphs taller than this). Always >= 0.

Definition at line 100 of file Font.cs.

6.13.4.2 Descent

```
double VectSharp.Font.Descent  [get]
```

Maximum depth below the baseline of the usual glyphs in the font (there may be glyphs deeper than this). Always <= 0.

Definition at line 136 of file Font.cs.

6.13.4.3 FontFamily

```
FontFamily VectSharp.Font.FontFamily  [get]
```

Font typeface.

Definition at line 84 of file Font.cs.

6.13.4.4 FontSize

```
double VectSharp.Font.FontSize  [get]
```

Font size, in graphics units.

Definition at line 79 of file Font.cs.

6.13.4.5 WinAscent

```
double VectSharp.Font.WinAscent  [get]
```

Height above the baseline for a clipping region (Windows ascent). Always >= 0.

Definition at line 118 of file Font.cs.

6.13.4.6 YMax

```
double VectSharp.Font.YMax  [get]
```

Absolute maximum height over the baseline of the glyphs in the font. Always >= 0.

Definition at line 154 of file Font.cs.

6.13.4.7 YMin

```
double VectSharp.Font.YMin  [get]
```

Absolute maximum depth below the baseline of the glyphs in the font. Always <= 0.

Definition at line 172 of file Font.cs.

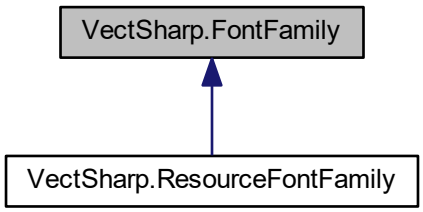
The documentation for this class was generated from the following file:

- VectSharp/Font.cs

6.14 VectSharp.FontFamily Class Reference

Represents a typeface.

Inheritance diagram for VectSharp.FontFamily:



Public Types

- enum StandardFontFamilies {
  StandardFontFamilies.TimesRoman, StandardFontFamilies.TimesBold, StandardFontFamilies.TimesItalic,
  StandardFontFamilies.TimesBoldItalic,
  StandardFontFamilies.Helvetica, StandardFontFamilies.HelveticaBold, StandardFontFamilies.HelveticaOblique,
  StandardFontFamilies.HelveticaBoldOblique,
  StandardFontFamilies.Courier, StandardFontFamilies.CourierBold, StandardFontFamilies.CourierOblique,
  StandardFontFamilies.CourierBoldOblique,
  StandardFontFamilies.Symbol, StandardFontFamilies.ZapfDingbats }

  The 14 standard font families.

Public Member Functions

- FontFamily (string fileName)

  Create a new FontFamily.
- FontFamily (Stream ttfStream)

  Create a new FontFamily.
- FontFamily (TrueTypeFile ttf)

  Create a new FontFamily.
- FontFamily (StandardFontFamilies standardFontFamily)

  Create a new standard FontFamily.

Static Public Member Functions

- static FontFamily ResolveFontFamily (string fontFamily)

  Create a new font family from the specified family name or true type file. If the family name or the true type file are not valid, an exception might be raised. Equivalent to DefaultFontLibrary.ResolveFontFamily.
- static FontFamily ResolveFontFamily (StandardFontFamilies standardFontFamily)

  Create a new font family from the specified standard font family name. Equivalent to DefaultFontLibrary.ResolveFontFamily.
- static FontFamily ResolveFontFamily (string fontFamily, params string[ ] fallback)

  Create a new font family from the specified family name or true type file. If the family name or the true type file are not valid, try to instantiate the font family using the fallback . If none of the fallback family names or true type files are valid, an exception might be raised. Equivalent to DefaultFontLibrary.ResolveFontFamily.
- static FontFamily ResolveFontFamily (string fontFamily, StandardFontFamilies finalFallback, params string[ ] fallback)

  Create a new font family from the specified family name or true type file. If the family name or the true type file are not valid, try to instantiate the font family using the fallback . If none of the fallback family names or true type files are valid, instantiate a standard font family using the finalFallback . Equivalent to DefaultFontLibrary.ResolveFontFamily.

Static Public Attributes

- static string[ ] StandardFamilies = new string[ ] { "Times-Roman", "Times-Bold", "Times-Italic", "Times-Bold↩ Italic", "Helvetica", "Helvetica-Bold", "Helvetica-Oblique", "Helvetica-BoldOblique", "Courier", "Courier-Bold", "Courier-Oblique", "Courier-BoldOblique", "Symbol", "ZapfDingbats" }

  The names of the 14 standard families that are guaranteed to be displayed correctly.
- static string[ ] StandardFontFamilyResources

  The names of the resource streams pointing to the included TrueType font files for each of the standard 14 font families.

Properties

- static IFontLibrary DefaultFontLibrary = new DefaultFontLibrary() `[get, set]`

  The default font library used to resolve font family names.
- bool IsStandardFamily `[get]`

  Whether this is one of the 14 standard font families or not.
- string FileName `[get]`

  Full path to the TrueType font file for this font family (or, if this is a standard font family, name of the font family).
- TrueTypeFile TrueTypeFile `[get]`

  Parsed TrueType font file for this font family. See also:
  See also

  VectSharp.TrueTypeFile

  .
- bool IsBold `[get]`

  Whether this font is bold or not. This is set based on the information included in the OS/2 table of the TrueType file.
- bool IsItalic `[get]`

  Whether this font is italic or oblique or not. This is set based on the information included in the OS/2 table of the TrueType file.
- bool IsOblique `[get]`

  Whether this font is oblique or not. This is set based on the information included in the OS/2 table of the TrueType file.

6.14.1 Detailed Description

Represents a typeface.

Definition at line 261 of file Font.cs.

6.14.2 Member Enumeration Documentation

6.14.2.1 StandardFontFamilies

```
enum VectSharp.FontFamily.StandardFontFamilies  [strong]
```

The 14 standard font families.

Enumerator

TimesRoman	Serif normal regular face.
TimesBold	Serif bold regular face.
TimesItalic	Serif normal italic face.
TimesBoldItalic	Serif bold italic face.
Helvetica	Sans-serif normal regular face.
HelveticaBold	Sans-serif bold regular face.
HelveticaOblique	Sans-serif normal oblique face.
HelveticaBoldOblique	Sans-serif bold oblique face.
Courier	Monospace normal regular face.
CourierBold	Monospace bold regular face.
CourierOblique	Monospace normal oblique face.
CourierBoldOblique	Monospace bold oblique face.
Symbol	Symbol font.
ZapfDingbats	Dingbat font.

Definition at line 340 of file Font.cs.

6.14.3 Constructor & Destructor Documentation

6.14.3.1 FontFamily() [1/4]

```
VectSharp.FontFamily.FontFamily (
            string fileName )
```

Create a new FontFamily.

Parameters

fileName	The full path to the TrueType font file for this font family or the name of a standard font family.

Definition at line 444 of file Font.cs.

6.14.3.2 FontFamily() [2/4]

```
VectSharp.FontFamily.FontFamily (
            Stream ttfStream )
```

Create a new FontFamily.

Parameters

ttfStream	A stream containing a file in TTF format.

Definition at line 468 of file Font.cs.

6.14.3.3 FontFamily() [3/4]

```
VectSharp.FontFamily.FontFamily (
            TrueTypeFile ttf )
```

Create a new FontFamily.

Parameters

ttf	A font file in TTF format.

Definition at line 487 of file Font.cs.

6.14.3.4 FontFamily() [4/4]

```
VectSharp.FontFamily.FontFamily (
            StandardFontFamilies standardFontFamily )
```

Create a new standard FontFamily.

Parameters

standardFontFamily	The standard font family.

Definition at line 507 of file Font.cs.

6.14.4 Member Function Documentation

6.14.4.1 ResolveFontFamily() [1/4]

```
static FontFamily VectSharp.FontFamily.ResolveFontFamily (
            StandardFontFamilies standardFontFamily )  [static]
```

Create a new font family from the specified standard font family name. Equivalent to DefaultFontLibrary.ResolveFontFamily.

Parameters

standardFontFamily	The standard name of the font family.

Returns

A FontFamily object corresponding to the specified font family.

6.14.4.2 ResolveFontFamily() [2/4]

```
static FontFamily VectSharp.FontFamily.ResolveFontFamily (
            string fontFamily )  [static]
```

Create a new font family from the specified family name or true type file. If the family name or the true type file are not valid, an exception might be raised. Equivalent to DefaultFontLibrary.ResolveFontFamily.

Parameters

fontFamily	The name of the font family to create, or the path to a TTF file.

Returns

If the font family name or the true type file is valid, a FontFamily object corresponding to the specified font family.

6.14.4.3 ResolveFontFamily() [3/4]

```
static FontFamily VectSharp.FontFamily.ResolveFontFamily (
            string fontFamily,
            params string[] fallback )  [static]
```

Create a new font family from the specified family name or true type file. If the family name or the true type file are not valid, try to instantiate the font family using the *fallback* . If none of the fallback family names or true type files are valid, an exception might be raised. Equivalent to DefaultFontLibrary.ResolveFontFamily.

Parameters

fontFamily	The name of the font family to create, or the path to a TTF file.
fallback	Names of additional font families or TTF files, which will be tried if the first *fontFamily* is not valid.

Returns

A FontFamily object corresponding to the first of the specified font families that is valid.

6.14.4.4 ResolveFontFamily() [4/4]

```
static FontFamily VectSharp.FontFamily.ResolveFontFamily (
            string fontFamily,
            StandardFontFamilies finalFallback,
            params string[] fallback )  [static]
```

Create a new font family from the specified family name or true type file. If the family name or the true type file are not valid, try to instantiate the font family using the *fallback* . If none of the fallback family names or true type files are valid, instantiate a standard font family using the *finalFallback* . Equivalent to DefaultFontLibrary.ResolveFontFamily.

Parameters

fontFamily	The name of the font family to create, or the path to a TTF file.
fallback	Names of additional font families or TTF files, which will be tried if the first *fontFamily* is not valid.
finalFallback	The standard name of the font family that will be used if none of the fallback families are valid.

Returns

A FontFamily object corresponding to the first of the specified font families that is valid.

6.14.5 Member Data Documentation

6.14.5.1 StandardFamilies

```
string [] VectSharp.FontFamily.StandardFamilies = new string[] { "Times-Roman", "Times-Bold",
"Times-Italic", "Times-BoldItalic", "Helvetica", "Helvetica-Bold", "Helvetica-Oblique", "Helvetica-Bold↩
Oblique", "Courier", "Courier-Bold", "Courier-Oblique", "Courier-BoldOblique", "Symbol", "Zapf↩
Dingbats" }  [static]
```

The names of the 14 standard families that are guaranteed to be displayed correctly.

Definition at line 319 of file Font.cs.

6.14.5.2 StandardFontFamilyResources

```
string [] VectSharp.FontFamily.StandardFontFamilyResources  [static]
```

Initial value:

```
= new string[]
        {
            "VectSharp.StandardFonts.Tinos-Regular.ttf", "VectSharp.StandardFonts.Tinos-Bold.ttf",
       "VectSharp.StandardFonts.Tinos-Italic.ttf", "VectSharp.StandardFonts.Tinos-BoldItalic.ttf",
            "VectSharp.StandardFonts.Arimo-Regular.ttf", "VectSharp.StandardFonts.Arimo-Bold.ttf",
       "VectSharp.StandardFonts.Arimo-Italic.ttf", "VectSharp.StandardFonts.Arimo-BoldItalic.ttf",
            "VectSharp.StandardFonts.Cousine-Regular.ttf", "VectSharp.StandardFonts.Cousine-Bold.ttf",
       "VectSharp.StandardFonts.Cousine-Italic.ttf", "VectSharp.StandardFonts.Cousine-BoldItalic.ttf",
            "VectSharp.StandardFonts.SymbolNeu_GB.ttf", "VectSharp.StandardFonts.Levibats-Regular_GB.ttf"
        }
```

The names of the resource streams pointing to the included TrueType font files for each of the standard 14 font families.

Definition at line 324 of file Font.cs.

6.14.6 Property Documentation

6.14.6.1 DefaultFontLibrary

```
IFontLibrary VectSharp.FontFamily.DefaultFontLibrary = new DefaultFontLibrary()  [static],
[get], [set]
```

The default font library used to resolve font family names.

Definition at line 266 of file Font.cs.

6.14.6.2 FileName

```
string VectSharp.FontFamily.FileName  [get]
```

Full path to the TrueType font file for this font family (or, if this is a standard font family, name of the font family).

Definition at line 416 of file Font.cs.

6.14.6.3 IsBold

```
bool VectSharp.FontFamily.IsBold  [get]
```

Whether this font is bold or not. This is set based on the information included in the OS/2 table of the TrueType file.

Definition at line 427 of file Font.cs.

6.14.6.4 IsItalic

```
bool VectSharp.FontFamily.IsItalic  [get]
```

Whether this font is italic or oblique or not. This is set based on the information included in the OS/2 table of the TrueType file.

Definition at line 432 of file Font.cs.

6.14.6.5 IsOblique

```
bool VectSharp.FontFamily.IsOblique  [get]
```

Whether this font is oblique or not. This is set based on the information included in the OS/2 table of the TrueType file.

Definition at line 437 of file Font.cs.

6.14.6.6 IsStandardFamily

```
bool VectSharp.FontFamily.IsStandardFamily  [get]
```

Whether this is one of the 14 standard font families or not.

Definition at line 335 of file Font.cs.

6.14.6.7 TrueTypeFile

```
TrueTypeFile VectSharp.FontFamily.TrueTypeFile  [get]
```

Parsed TrueType font file for this font family. See also:

See also

VectSharp.TrueTypeFile

.

Definition at line 422 of file Font.cs.

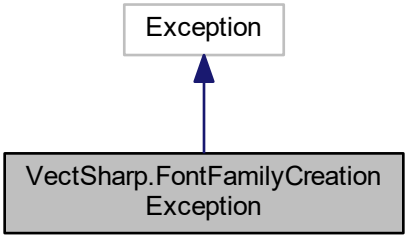
The documentation for this class was generated from the following file:

- VectSharp/Font.cs

6.15 VectSharp.FontFamilyCreationException Class Reference

An exception that occurs while creating a FontFamily.

Inheritance diagram for VectSharp.FontFamilyCreationException:



Public Member Functions

- FontFamilyCreationException (string fontFamily)

  Create a new FontFamilyCreationException instance.

Properties

- string FontFamily `[get]`

  The name of the font family that was being created.

6.15.1 Detailed Description

An exception that occurs while creating a FontFamily.

Definition at line 441 of file FontLibrary.cs.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 FontFamilyCreationException()

```
VectSharp.FontFamilyCreationException.FontFamilyCreationException (
            string fontFamily )
```

Create a new FontFamilyCreationException instance.

Parameters

fontFamily	The name of the font family that was being created.

Definition at line 452 of file FontLibrary.cs.

6.15.3 Property Documentation

6.15.3.1 FontFamily

```
string VectSharp.FontFamilyCreationException.FontFamily  [get]
```

The name of the font family that was being created.

Definition at line 446 of file FontLibrary.cs.

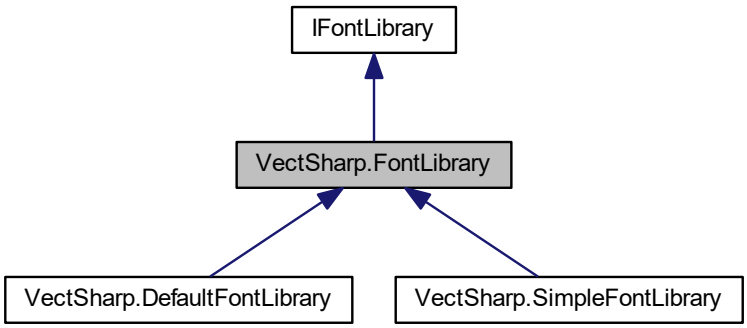
The documentation for this class was generated from the following file:

- VectSharp/FontLibrary.cs

6.16 VectSharp.FontLibrary Class Reference

Abstract class with a default implementation of font family fallbacks.

Inheritance diagram for VectSharp.FontLibrary:



Public Member Functions

- abstract FontFamily ResolveFontFamily (string fontFamily)

  Create a new font family from the specified family name or true type file. If the family name or the true type file are not valid, an exception might be raised.
- abstract FontFamily ResolveFontFamily (FontFamily.StandardFontFamilies standardFontFamily)

  Create a new font family from the specified standard font family name.
- virtual FontFamily ResolveFontFamily (string fontFamily, params string[ ] fallback)

  Create a new font family from the specified family name or true type file. If the family name or the true type file are not valid, try to instantiate the font family using the fallback . If none of the fallback family names or true type files are valid, an exception might be raised.
- virtual FontFamily ResolveFontFamily (string fontFamily, FontFamily.StandardFontFamilies finalFallback, params string[ ] fallback)

  Create a new font family from the specified family name or true type file. If the family name or the true type file are not valid, try to instantiate the font family using the fallback . If none of the fallback family names or true type files are valid, instantiate a standard font family using the finalFallback .

6.16.1 Detailed Description

Abstract class with a default implementation of font family fallbacks.

Definition at line 68 of file FontLibrary.cs.

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

- VectSharp/FontLibrary.cs

6.17 VectSharp.Markdown.FormattedString Struct Reference

Represents a string with associated formatting information.

Public Member Functions

- FormattedString (string text, Colour colour, bool isBold, bool isItalic)

  Creates a new FormattedString instance.

Properties

- string Text `[get]`

  The text represented by this object.
- Colour Colour `[get]`

  The colour of the text.
- bool IsBold `[get]`

  Whether the text should be rendered as bold or not.
- bool IsItalic `[get]`

  Whether the text should be rendered as italic or not.

6.17.1 Detailed Description

Represents a string with associated formatting information.

Definition at line 32 of file SyntaxHighlighting.cs.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 FormattedString()

```
VectSharp.Markdown.FormattedString.FormattedString (
            string text,
            Colour colour,
            bool isBold,
            bool isItalic )
```

Creates a new FormattedString instance.

Parameters

text	The text of the object.
colour	The colour of the text.
isBold	Whether the text should be rendered as bold or not.
isItalic	Whether the text should be rendered as italic or not.

Definition at line 61 of file SyntaxHighlighting.cs.

6.17.3 Property Documentation

6.17.3.1 Colour

```
Colour VectSharp.Markdown.FormattedString.Colour  [get]
```

The colour of the text.

Definition at line 42 of file SyntaxHighlighting.cs.

6.17.3.2 IsBold

```
bool VectSharp.Markdown.FormattedString.IsBold  [get]
```

Whether the text should be rendered as bold or not.

Definition at line 47 of file SyntaxHighlighting.cs.

6.17.3.3 IsItalic

```
bool VectSharp.Markdown.FormattedString.IsItalic  [get]
```

Whether the text should be rendered as italic or not.

Definition at line 52 of file SyntaxHighlighting.cs.

6.17.3.4 Text

```
string VectSharp.Markdown.FormattedString.Text  [get]
```

The text represented by this object.

Definition at line 37 of file SyntaxHighlighting.cs.

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

- VectSharp.Markdown/SyntaxHighlighting.cs

6.18 VectSharp.FormattedText Class Reference

Represents a run of text that should be drawn with the same style.

Public Member Functions

- FormattedText (string text, Font font, Script script=Script.Normal, Brush brush=null)

  Creates a new FormattedText instance with the specified text , font , script position and brush .

Static Public Member Functions

- static IEnumerable< FormattedText > Format (string text, Font normalFont, Font boldFont, Font italicFont, Font boldItalicFont, Brush defaultBrush=null)

  Parse the formatting information contained in a text string into a collection of FormattedText objects.
- static IEnumerable< FormattedText > Format (string text, FontFamily.StandardFontFamilies fontFamily, double fontSize, Brush defaultBrush=null)

  Parse the formatting information contained in a text string into a collection of FormattedText objects, using fonts from a standard font family.

Properties

- string Text [get]

  Represents the text represented by this instance.
- Font Font [get]

  Represents the font that should be used to draw the text.
- Script Script [get]

  Represents the position of the text.
- Brush Brush [get]

  Represents the brush that should be used to draw the text. If this is null, the default brush is used.

6.18.1 Detailed Description

Represents a run of text that should be drawn with the same style.

Definition at line 50 of file FormattedText.cs.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 FormattedText()

```
VectSharp.FormattedText.FormattedText (
            string text,
            Font font,
            Script script = Script.Normal,
            Brush brush = null )
```

Creates a new FormattedText instance with the specified *text* , *font* , *script* position and *brush* .

Parameters

text	The text that will be contained in the new FormattedText.
font	The font that will be used by the new FormattedText.
script	The script position of the new FormattedText.
brush	The brush that will be used by the new FormattedText.

Definition at line 79 of file FormattedText.cs.

6.18.3 Member Function Documentation

6.18.3.1 Format() [1/2]

```
static IEnumerable<FormattedText> VectSharp.FormattedText.Format (
            string text,
            Font normalFont,
            Font boldFont,
            Font italicFont,
            Font boldItalicFont,
            Brush defaultBrush = null )  [static]
```

Parse the formatting information contained in a text string into a collection of FormattedText objects.

Parameters

text	The string containing formatting information. Format information is specified using HTML-like tags: • `<b></b>` or `<strong></strong>` are used for bold text; • `<i></i>` or `<em></em>` are used for text in italics; • `<sup></sup>` and `<sub></sub>` are used, respectively, for superscript and subscript text; • `<#COLOUR></#>` is used to specify the colour of the text, where `COLOUR` is a CSS colour string (e.g. `<#red>`, `<#0080FF>`, or `<#rgba(128, 80, 52, 0.5)>`).
normalFont	The font that will be used for text that is neither bold nor italic.
boldFont	The font that will be used for text that is bold. Note that this does not necessarily have to be a bold font; this is just the font that is applied to text contained within `<b></b>` tags.
italicFont	The font that will be used for text that is in italics. Note that this does not necessarily have to be an italic font; this is just the font that is applied to text contained within `<i></i>` tags.
boldItalicFont	The font that will be used for text that is both in bold and in italics.
defaultBrush	The default Brush that will be used for text runs that do not specify a colour. If this is `null`, the default Brush will be the one specified in the painting call.

Returns

A lazy collection of FormattedText objects. Note that every enumeration of this collection causes the text to be parsed again; if you need to enumerate this collection more than once, you should probably convert it e.g. to a List<T>.

Definition at line 104 of file FormattedText.cs.

6.18.3.2 Format() [2/2]

```
static IEnumerable<FormattedText> VectSharp.FormattedText.Format (
            string text,
            FontFamily.StandardFontFamilies fontFamily,
            double fontSize,
            Brush defaultBrush = null )  [static]
```

Parse the formatting information contained in a text string into a collection of FormattedText objects, using fonts from a standard font family.

Parameters

text	The string containing formatting information. Format information is specified using HTML-like tags: • or are used for bold text; • or are used for text in italics; • and are used, respectively, for superscript and subscript text; • <#COLOUR></#> is used to specify the colour of the text, where COLOUR is a CSS colour string (e.g. <#red>, <#0080FF>, or <#rgba(128, 80, 52, 0.5)>).
fontFamily	The font family from which the fonts will be created. If this is a regular font family, the bold, italic and bold-italic versions of the font will be used for the formatted text. Otherwise, the relevant font styles will be toggled (e.g. if the supplied font family is bold, then regular text in the formatted string will be displayed as bold, while bold text in the formatted string will be displayed as regular text).
fontSize	The size of the fonts to use.
defaultBrush	The default Brush that will be used for text runs that do not specify a colour. If this is null, the default Brush will be the one specified in the painting call.

Returns

A lazy collection of FormattedText objects. Note that every enumeration of this collection causes the text to be parsed again; if you need to enumerate this collection more than once, you should probably convert it e.g. to a List<T>.

Definition at line 259 of file FormattedText.cs.

6.18.4 Property Documentation

6.18.4.1 Brush

```
Brush VectSharp.FormattedText.Brush  [get]
```

Represents the brush that should be used to draw the text. If this is null, the default brush is used.

Definition at line 70 of file FormattedText.cs.

6.18.4.2 Font

```
Font VectSharp.FormattedText.Font  [get]
```

Represents the font that should be used to draw the text.

Definition at line 60 of file FormattedText.cs.

6.18.4.3 Script

```
Script VectSharp.FormattedText.Script  [get]
```

Represents the position of the text.

Definition at line 65 of file FormattedText.cs.

6.18.4.4 Text

```
string VectSharp.FormattedText.Text  [get]
```

Represents the text represented by this instance.

Definition at line 55 of file FormattedText.cs.

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

- VectSharp/FormattedText.cs

6.19 VectSharp.FormattedTextExtensions Class Reference

Contains extension methods for collections of FormattedText objects.

Static Public Member Functions

- static Font.DetailedFontMetrics Measure (this IEnumerable< FormattedText > text)
  Measures a collection of FormattedText objects.

6.19.1 Detailed Description

Contains extension methods for collections of FormattedText objects.

Definition at line 477 of file FormattedText.cs.

6.19.2 Member Function Documentation

6.19.2.1 Measure()

```
static Font.DetailedFontMetrics VectSharp.FormattedTextExtensions.Measure (
            this IEnumerable< FormattedText > text )  [static]
```

Measures a collection of FormattedText objects.

Parameters

text	The collection of FormattedText objects to be measured.

Returns

A Font.DetailedFontMetrics containing detailed measurements for the text obtained by composing the elements in the FormattedText collection.

Definition at line 561 of file FormattedText.cs.

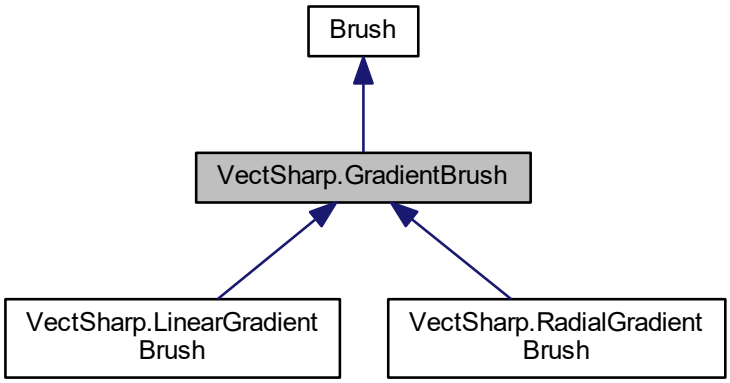
The documentation for this class was generated from the following file:

- VectSharp/FormattedText.cs

6.20 VectSharp.GradientBrush Class Reference

Represents a brush painting with a gradient.

Inheritance diagram for VectSharp.GradientBrush:



Properties

- GradientStops GradientStops `[get]`

  The colour stops in the gradient.

Additional Inherited Members

6.20.1 Detailed Description

Represents a brush painting with a gradient.

Definition at line 231 of file Brush.cs.

6.20.2 Property Documentation

6.20.2.1 GradientStops

```
GradientStops VectSharp.GradientBrush.GradientStops [get]
```

The colour stops in the gradient.

Definition at line 236 of file Brush.cs.

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

- VectSharp/Brush.cs

6.21 VectSharp.GradientStop Struct Reference

Represents a colour stop in a gradient.

Public Member Functions

- GradientStop (Colour colour, double offset)

  Creates a new GradientStop instance.
- GradientStop MultiplyOpacity (double opacity)

  Returns a GradientStop corresponding to the current instance, whose colour's opacity has been multiplied by the specified value.

Properties

- Colour Colour [get]

  The Colour at the gradient stop.
- double Offset [get]

  The offset of the gradient stop. Range: [0, 1].

6.21.1 Detailed Description

Represents a colour stop in a gradient.

Definition at line 109 of file Brush.cs.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 GradientStop()

```
VectSharp.GradientStop.GradientStop (
            Colour colour,
            double offset )
```

Creates a new GradientStop instance.

Parameters

colour	The Colour at the gradient stop.
offset	The offset of the gradient stop. Range: [0, 1].

Definition at line 126 of file Brush.cs.

6.21.3 Member Function Documentation

6.21.3.1 MultiplyOpacity()

```
GradientStop VectSharp.GradientStop.MultiplyOpacity (
            double opacity )
```

Returns a GradientStop corresponding to the current instance, whose colour's opacity has been multiplied by the specified value.

Parameters

opacity	The value that will be used to multiply the colour's opacity.

Returns

A GradientStop corresponding to the current instance, whose colour's opacity has been multiplied by the specified value.

Definition at line 137 of file Brush.cs.

6.21.4 Property Documentation

6.21.4.1 Colour

```
Colour VectSharp.GradientStop.Colour  [get]
```

The Colour at the gradient stop.

Definition at line 114 of file Brush.cs.

6.21.4.2 Offset

```
double VectSharp.GradientStop.Offset  [get]
```

The offset of the gradient stop. Range: [0, 1].

Definition at line 119 of file Brush.cs.

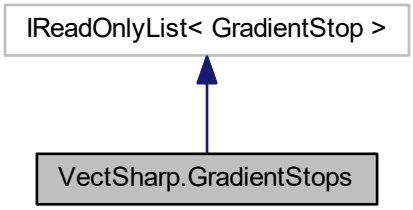
The documentation for this struct was generated from the following file:

- VectSharp/Brush.cs

6.22 VectSharp.GradientStops Class Reference

Represents a read-only list of GradientStops.

Inheritance diagram for VectSharp.GradientStops:



Public Member Functions

- IEnumerator< GradientStop > GetEnumerator ()
- GradientStops (IEnumerable< GradientStop > gradientStops)

  Creates a new GradientStops instance containing the specified gradient stops.
- GradientStops (params GradientStop[ ] gradientStops)

  Creates a new GradientStops instance containing the specified gradient stops.

Public Attributes

- GradientStop this[int index] => gradientStops[index]
- int Count => gradientStops.Count

Static Public Attributes

- static readonly double StopTolerance = 1e-7

  The minimum distance that is enforced between consecutive gradient stops.

6.22.1 Detailed Description

Represents a read-only list of GradientStops.

Definition at line 146 of file Brush.cs.

6.22.2 Constructor & Destructor Documentation

6.22.2.1 GradientStops() [1/2]

```
VectSharp.GradientStops.GradientStops (
            IEnumerable< GradientStop > gradientStops )
```

Creates a new GradientStops instance containing the specified gradient stops.

Parameters

gradientStops	The gradient stops that will be contained in the GradientStops object.

Definition at line 176 of file Brush.cs.

6.22.2.2 GradientStops() [2/2]

```
VectSharp.GradientStops.GradientStops (
            params GradientStop[] gradientStops )
```

Creates a new GradientStops instance containing the specified gradient stops.

Parameters

gradientStops	The gradient stops that will be contained in the GradientStops object.

Definition at line 222 of file Brush.cs.

6.22.3 Member Data Documentation

6.22.3.1 StopTolerance

```
readonly double VectSharp.GradientStops.StopTolerance = 1e-7  [static]
```

The minimum distance that is enforced between consecutive gradient stops.

Definition at line 151 of file Brush.cs.

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

- VectSharp/Brush.cs

6.23 VectSharp.Graphics Class Reference

Represents an abstract drawing surface.

Public Member Functions

- void FillPath (GraphicsPath path, Brush fillColour, string tag=null)

  Fill a GraphicsPath.
- void StrokePath (GraphicsPath path, Brush strokeColour, double lineWidth=1, LineCaps lineCap=LineCaps.Butt, LineJoins lineJoin=LineJoins.Miter, LineDash? lineDash=null, string tag=null)

  Stroke a GraphicsPath.
- void SetClippingPath (GraphicsPath path)

  Intersect the current clipping path with the specified GraphicsPath.
- void SetClippingPath (double leftX, double topY, double width, double height)

  Intersect the current clipping path with the specified rectangle.
- void SetClippingPath (Point topLeft, Size size)

  Intersect the current clipping path with the specified rectangle.
- void Rotate (double angle)

  Rotate the coordinate system around the origin.
- void RotateAt (double angle, Point pivot)

  Rotate the coordinate system around a pivot point.
- void Transform (double a, double b, double c, double d, double e, double f)

  Transform the coordinate system with the specified transformation matrix [[a, c, e], [b, d, f], [0, 0, 1]].
- void Translate (double x, double y)

  Translate the coordinate system origin.
- void Translate (Point delta)

  Translate the coordinate system origin.
- void Scale (double scaleX, double scaleY)

  Scale the coordinate system with respect to the origin.
- void FillRectangle (Point topLeft, Size size, Brush fillColour, string tag=null)

  Fill a rectangle.
- void FillRectangle (double leftX, double topY, double width, double height, Brush fillColour, string tag=null)

  Fill a rectangle.
- void StrokeRectangle (Point topLeft, Size size, Brush strokeColour, double lineWidth=1, LineCaps lineCap=LineCaps.Butt, LineJoins lineJoin=LineJoins.Miter, LineDash? lineDash=null, string tag=null)

  Stroke a rectangle.
- void StrokeRectangle (double leftX, double topY, double width, double height, Brush strokeColour, double lineWidth=1, LineCaps lineCap=LineCaps.Butt, LineJoins lineJoin=LineJoins.Miter, LineDash? lineDash=null, string tag=null)

  Stroke a rectangle.

- void DrawRasterImage (int sourceX, int sourceY, int sourceWidth, int sourceHeight, double destinationX, double destinationY, double destinationWidth, double destinationHeight, RasterImage image, string tag=null)

  Draw a raster image.
- void DrawRasterImage (double x, double y, RasterImage image, string tag=null)

  Draw a raster image.
- void DrawRasterImage (Point position, RasterImage image, string tag=null)

  Draw a raster image.
- void DrawRasterImage (double x, double y, double width, double height, RasterImage image, string tag=null)

  Draw a raster image.
- void DrawRasterImage (Point position, Size size, RasterImage image, string tag=null)

  Draw a raster image.
- void Save ()

  Save the current transform state (rotation, translation, scale).
- void Restore ()

  Restore the previous transform state (rotation, translation scale).
- void CopyToIGraphicsContext (IGraphicsContext destinationContext)

  Copy the current graphics to an instance of a class implementing IGraphicsContext.
- void DrawGraphics (Point origin, Graphics graphics)

  Draws a Graphics object on the current Graphics object.
- void DrawGraphics (double originX, double originY, Graphics graphics)

  Draws a Graphics object on the current Graphics object.
- Graphics Transform (Func< Point, Point > transformationFunction, double linearisationResolution)

  Creates a new Graphics object in which all the graphics actions have been transformed using an arbitrary transformation function. Raster images are replaced by grey rectangles.
- Graphics Linearise (double resolution)

  Creates a new Graphics object by linearising all of the elements of the current instance, i.e. replacing curve segments with series of line segments that approximate them. Raster images are left unchanged.
- void FillText (Point origin, string text, Font font, Brush fillColour, TextBaselines textBaseline=TextBaselines.Top, string tag=null)

  Fill a text string.
- void FillText (double originX, double originY, string text, Font font, Brush fillColour, TextBaselines textBaseline=TextBaselines.Top, string tag=null)

  Fill a text string.
- void StrokeText (Point origin, string text, Font font, Brush strokeColour, TextBaselines textBaseline=TextBaselines.Top, double lineWidth=1, LineCaps lineCap=LineCaps.Butt, LineJoins lineJoin=LineJoins.Miter, LineDash? lineDash=null, string tag=null)

  Stroke a text string.
- void StrokeText (double originX, double originY, string text, Font font, Brush strokeColour, TextBaselines textBaseline=TextBaselines.Top, double lineWidth=1, LineCaps lineCap=LineCaps.Butt, LineJoins lineJoin=LineJoins.Miter, LineDash? lineDash=null, string tag=null)

  Stroke a text string.
- void FillTextOnPath (GraphicsPath path, string text, Font font, Brush fillColour, double reference=0, TextAnchors anchor=TextAnchors.Left, TextBaselines textBaseline=TextBaselines.Top, string tag=null)

  Fill a text string along a GraphicsPath.
- void StrokeTextOnPath (GraphicsPath path, string text, Font font, Brush strokeColour, double reference=0, TextAnchors anchor=TextAnchors.Left, TextBaselines textBaseline=TextBaselines.Top, double lineWidth=1, LineCaps lineCap=LineCaps.Butt, LineJoins lineJoin=LineJoins.Miter, LineDash? lineDash=null, string tag=null)

  Stroke a text string along a GraphicsPath.
- void FillText (Point origin, IEnumerable< FormattedText > text, Brush fillColour, TextBaselines textBaseline=TextBaselines.Top, string tag=null)

  Fill a formatted text string.

- void FillText (double originX, double originY, IEnumerable< FormattedText > text, Brush fillColour, TextBaselines textBaseline=TextBaselines.Top, string tag=null)

  Fill a formatted text string.
- void StrokeText (Point origin, IEnumerable< FormattedText > text, Brush strokeColour, TextBaselines textBaseline=TextBaselines.Top, double lineWidth=1, LineCaps lineCap=LineCaps.Butt, LineJoins lineJoin=LineJoins.Miter, LineDash? lineDash=null, string tag=null)

  Stroke a formatted text string.
- void StrokeText (double originX, double originY, IEnumerable< FormattedText > text, Brush strokeColour, TextBaselines textBaseline=TextBaselines.Top, double lineWidth=1, LineCaps lineCap=LineCaps.Butt, LineJoins lineJoin=LineJoins.Miter, LineDash? lineDash=null, string tag=null)

  Stroke a formatted text string.
- Size MeasureText (string text, Font font)

  Measure a text string. See also
  See also

  Font.MeasureText(string), Font.MeasureTextAdvanced(string)

  and .
- Size MeasureText (IEnumerable< FormattedText > text)

  Measure a formatted text string. See also
  See also

  FormattedTextExtensions.Measure(IEnumerable<FormattedText>)

  .

Properties

- static UnbalancedStackActions UnbalancedStackAction = UnbalancedStackActions.Throw `[get, set]`

  Determines how an unbalanced graphics state stack (which occurs if the number of calls to Save and Restore is not equal) will be treated. The default is UnbalancedStackActions.Throw.

6.23.1 Detailed Description

Represents an abstract drawing surface.

Definition at line 253 of file Graphics.cs.

6.23.2 Member Function Documentation

6.23.2.1 CopyToIGraphicsContext()

```
void VectSharp.Graphics.CopyToIGraphicsContext (
            IGraphicsContext destinationContext )
```

Copy the current graphics to an instance of a class implementing IGraphicsContext.

Parameters

destinationContext	The IGraphicsContext on which the graphics are to be copied.

Definition at line 591 of file Graphics.cs.

6.23.2.2 DrawGraphics() [1/2]

```
void VectSharp.Graphics.DrawGraphics (
            double originX,
            double originY,
            Graphics graphics )
```

Draws a Graphics object on the current Graphics object.

Parameters

originX	The horizontal coordinate at which to place the origin of *graphics* .
originY	The vertical coordinate at which to place the origin of *graphics* .
graphics	The Graphics object to draw on the current Graphics object.

Definition at line 807 of file Graphics.cs.

6.23.2.3 DrawGraphics() [2/2]

```
void VectSharp.Graphics.DrawGraphics (
            Point origin,
            Graphics graphics )
```

Draws a Graphics object on the current Graphics object.

Parameters

origin	The point at which to place the origin of *graphics* .
graphics	The Graphics object to draw on the current Graphics object.

Definition at line 789 of file Graphics.cs.

6.23.2.4 DrawRasterImage() [1/5]

```
void VectSharp.Graphics.DrawRasterImage (
            double x,
            double y,
            double width,
            double height,
            RasterImage image,
            string tag = null )
```

Draw a raster image.

Parameters

x	The horizontal coordinate of the top-left corner of the rectangle delimiting the destination area of the image.
y	The vertical coordinate of the top-left corner of the rectangle delimiting the destination area of the image.
width	The width of the rectangle delimiting the destination area of the image.
height	The height of the rectangle delimiting the destination area of the image.
image	The image to draw.
tag	A tag to identify the drawn image.

Definition at line 496 of file Graphics.cs.

6.23.2.5 DrawRasterImage() [2/5]

```
void VectSharp.Graphics.DrawRasterImage (
            double x,
            double y,
            RasterImage image,
            string tag = null )
```

Draw a raster image.

Parameters

x	The horizontal coordinate of the top-left corner of the rectangle delimiting the destination area of the image.
y	The vertical coordinate of the top-left corner of the rectangle delimiting the destination area of the image.
image	The image to draw.
tag	A tag to identify the drawn image.

Definition at line 471 of file Graphics.cs.

6.23.2.6 DrawRasterImage() [3/5]

```
void VectSharp.Graphics.DrawRasterImage (
            int sourceX,
            int sourceY,
            int sourceWidth,
            int sourceHeight,
            double destinationX,
            double destinationY,
            double destinationWidth,
            double destinationHeight,
```

```
            RasterImage image,
            string tag = null )
```

Draw a raster image.

Parameters

sourceX	The horizontal coordinate of the top-left corner of the rectangle delimiting the source area of the image.
sourceY	The vertical coordinate of the top-left corner of the rectangle delimiting the source area of the image.
sourceWidth	The width of the rectangle delimiting the source area of the image.
sourceHeight	The height of the rectangle delimiting the source area of the image.
destinationX	The horizontal coordinate of the top-left corner of the rectangle delimiting the destination area of the image.
destinationY	The vertical coordinate of the top-left corner of the rectangle delimiting the destination area of the image.
destinationWidth	The width of the rectangle delimiting the destination area of the image.
destinationHeight	The height of the rectangle delimiting the destination area of the image.
image	The image to draw.
tag	A tag to identify the drawn image.

Definition at line 459 of file Graphics.cs.

6.23.2.7 DrawRasterImage() [4/5]

```
void VectSharp.Graphics.DrawRasterImage (
            Point position,
            RasterImage image,
            string tag = null )
```

Draw a raster image.

Parameters

position	The the top-left corner of the rectangle delimiting the destination area of the image.
image	The image to draw.
tag	A tag to identify the drawn image.

Definition at line 482 of file Graphics.cs.

6.23.2.8 DrawRasterImage() [5/5]

```
void VectSharp.Graphics.DrawRasterImage (
            Point position,
            Size size,
            RasterImage image,
            string tag = null )
```

Draw a raster image.

Parameters

position	The the top-left corner of the rectangle delimiting the destination area of the image.
size	The size of the rectangle delimiting the destination area of the image.
image	The image to draw.
tag	A tag to identify the drawn image.

Definition at line 508 of file Graphics.cs.

6.23.2.9 FillPath()

```
void VectSharp.Graphics.FillPath (
            GraphicsPath path,
            Brush fillColour,
            string tag = null )
```

Fill a GraphicsPath.

Parameters

path	The GraphicsPath to fill.
fillColour	The Brush with which to fill the GraphicsPath.
tag	A tag to identify the filled path.

Definition at line 268 of file Graphics.cs.

6.23.2.10 FillRectangle() [1/2]

```
void VectSharp.Graphics.FillRectangle (
            double leftX,
            double topY,
            double width,
            double height,
            Brush fillColour,
            string tag = null )
```

Fill a rectangle.

Parameters

leftX	The horizontal coordinate of the top-left corner of the rectangle.
topY	The vertical coordinate of the top-left corner of the rectangle.
width	The width of the rectangle.
height	The height of the rectangle.
fillColour	The colour with which to fill the rectangle.
tag	A tag to identify the filled rectangle.

Definition at line 407 of file Graphics.cs.

6.23.2.11 FillRectangle() [2/2]

```
void VectSharp.Graphics.FillRectangle (
            Point topLeft,
            Size size,
            Brush fillColour,
            string tag = null )
```

Fill a rectangle.

Parameters

topLeft	The top-left corner of the rectangle.
size	The size of the rectangle.
fillColour	The colour with which to fill the rectangle.
tag	A tag to identify the filled rectangle.

Definition at line 393 of file Graphics.cs.

6.23.2.12 FillText() [1/4]

```
void VectSharp.Graphics.FillText (
            double originX,
            double originY,
            IEnumerable< FormattedText > text,
            Brush fillColour,
            TextBaselines textBaseline = TextBaselines.Top,
            string tag = null )
```

Fill a formatted text string.

Parameters

originX	The horizontal coordinate of the text origin.
originY	The vertical coordinate of the text origin. See *textBaseline* .
text	The FormattedText to draw.
fillColour	The default Brush to use to fill the text. This can be overridden by each *text* element.
textBaseline	The text baseline (determines what *originY* represents).
tag	A tag to identify the filled text.

Definition at line 403 of file Graphics.Text.cs.

6.23.2.13 FillText() [2/4]

```
void VectSharp.Graphics.FillText (
            double originX,
            double originY,
            string text,
            Font font,
            Brush fillColour,
            TextBaselines textBaseline = TextBaselines.Top,
            string tag = null )
```

Fill a text string.

Parameters

originX	The horizontal coordinate of the text origin.
originY	The vertical coordinate of the text origin. See *textBaseline* .
text	The string to draw.
font	The font with which to draw the text.
fillColour	The Brush to use to fill the text.
textBaseline	The text baseline (determines what *originY* represents).
tag	A tag to identify the filled text.

Definition at line 52 of file Graphics.Text.cs.

6.23.2.14 FillText() [3/4]

```
void VectSharp.Graphics.FillText (
            Point origin,
            IEnumerable< FormattedText > text,
            Brush fillColour,
            TextBaselines textBaseline = TextBaselines.Top,
            string tag = null )
```

Fill a formatted text string.

Parameters

origin	The text origin. See *textBaseline* .
text	The FormattedText to draw.
fillColour	The default Brush to use to fill the text. This can be overridden by each *text* element.
textBaseline	The text baseline (determines what the vertical component of *origin* represents).
tag	A tag to identify the filled text.

Definition at line 310 of file Graphics.Text.cs.

6.23.2.15 FillText() [4/4]

```
void VectSharp.Graphics.FillText (
            Point origin,
            string text,
            Font font,
            Brush fillColour,
            TextBaselines textBaseline = TextBaselines.Top,
            string tag = null )
```

Fill a text string.

Parameters

origin	The text origin. See *textBaseline* .
text	The string to draw.
font	The font with which to draw the text.
fillColour	The Brush to use to fill the text.
textBaseline	The text baseline (determines what the vertical component of *origin* represents).
tag	A tag to identify the filled text.

Definition at line 37 of file Graphics.Text.cs.

6.23.2.16 FillTextOnPath()

```
void VectSharp.Graphics.FillTextOnPath (
            GraphicsPath path,
            string text,
            Font font,
            Brush fillColour,
            double reference = 0,
            TextAnchors anchor = TextAnchors.Left,
            TextBaselines textBaseline = TextBaselines.Top,
            string tag = null )
```

Fill a text string along a GraphicsPath.

Parameters

path	The GraphicsPath along which the text will flow.
text	The string to draw.
font	The font with which to draw the text.
fillColour	The Brush to use to fill the text.
reference	The (relative) starting point on the path starting from which the text should be drawn (0 is the start of the path, 1 is the end of the path).
anchor	The anchor in the text string that will correspond to the point specified by the *reference* .
textBaseline	The text baseline (determines which the position of the text in relation to the *path* .
tag	A tag to identify the filled text.

Definition at line 105 of file Graphics.Text.cs.

6.23.2.17 Linearise()

```
Graphics VectSharp.Graphics.Linearise (
            double resolution )
```

Creates a new Graphics object by linearising all of the elements of the current instance, i.e. replacing curve segments with series of line segments that approximate them. Raster images are left unchanged.

Parameters

resolution	The resolution that will be used to linearise curve segments.

Returns

A new Graphics object containing the linearised elements.

Definition at line 1010 of file Graphics.cs.

6.23.2.18 MeasureText() [1/2]

```
Size VectSharp.Graphics.MeasureText (
            IEnumerable< FormattedText > text )
```

Measure a formatted text string. See also

See also

FormattedTextExtensions.Measure(IEnumerable<FormattedText>)

.

Parameters

text	The collection of FormattedText objects to measure.

Returns

The size of the measured *text* .

Definition at line 540 of file Graphics.Text.cs.

6.23.2.19 MeasureText() [2/2]

```
Size VectSharp.Graphics.MeasureText (
            string text,
            Font font )
```

Measure a text string. See also

See also

Font.MeasureText(string), Font.MeasureTextAdvanced(string)

and .

Parameters

text	The string to measure.
font	The font to use to measure the string.

Returns

The size of the measured *text* .

Definition at line 529 of file Graphics.Text.cs.

6.23.2.20 Restore()

```
void VectSharp.Graphics.Restore ( )
```

Restore the previous transform state (rotation, translation scale).

Definition at line 524 of file Graphics.cs.

6.23.2.21 Rotate()

```
void VectSharp.Graphics.Rotate (
            double angle )
```

Rotate the coordinate system around the origin.

Parameters

angle	The angle (in radians) by which to rotate the coordinate system.

Definition at line 324 of file Graphics.cs.

6.23.2.22 RotateAt()

```
void VectSharp.Graphics.RotateAt (
            double angle,
            Point pivot )
```

Rotate the coordinate system around a pivot point.

Parameters

angle	The angle (in radians) by which to rotate the coordinate system.
pivot	The pivot around which the coordinate system is to be rotated.

Definition at line 334 of file Graphics.cs.

6.23.2.23 Save()

```
void VectSharp.Graphics.Save ( )
```

Save the current transform state (rotation, translation, scale).

Definition at line 516 of file Graphics.cs.

6.23.2.24 Scale()

```
void VectSharp.Graphics.Scale (
            double scaleX,
            double scaleY )
```

Scale the coordinate system with respect to the origin.

Parameters

scaleX	The horizontal scale.
scaleY	The vertical scale.

Definition at line 381 of file Graphics.cs.

6.23.2.25 SetClippingPath() [1/3]

```
void VectSharp.Graphics.SetClippingPath (
            double leftX,
```

```
            double topY,
            double width,
            double height )
```

Intersect the current clipping path with the specified rectangle.

Parameters

leftX	The horizontal coordinate of the top-left corner of the rectangle.
topY	The vertical coordinate of the top-left corner of the rectangle.
width	The width of the rectangle.
height	The height of the rectangle.

Definition at line 305 of file Graphics.cs.

6.23.2.26 SetClippingPath() [2/3]

```
void VectSharp.Graphics.SetClippingPath (
            GraphicsPath path )
```

Intersect the current clipping path with the specified GraphicsPath.

Parameters

path	The GraphicsPath to intersect with the current clipping path.

Definition at line 293 of file Graphics.cs.

6.23.2.27 SetClippingPath() [3/3]

```
void VectSharp.Graphics.SetClippingPath (
            Point topLeft,
            Size size )
```

Intersect the current clipping path with the specified rectangle.

Parameters

topLeft	The top-left corner of the rectangle.
size	The size of the rectangle.

Definition at line 315 of file Graphics.cs.

6.23.2.28 StrokePath()

```
void VectSharp.Graphics.StrokePath (
            GraphicsPath path,
            Brush strokeColour,
            double lineWidth = 1,
            LineCaps lineCap = LineCaps.Butt,
            LineJoins lineJoin = LineJoins.Miter,
            LineDash?  lineDash = null,
            string tag = null )
```

Stroke a GraphicsPath.

Parameters

path	The GraphicsPath to stroke.
strokeColour	The Brush with which to stroke the GraphicsPath.
lineWidth	The width of the line with which the path is stroked.
lineCap	The line cap to use to stroke the path.
lineJoin	The line join to use to stroke the path.
lineDash	The line dash to use to stroke the path.
tag	A tag to identify the stroked path.

Definition at line 284 of file Graphics.cs.

6.23.2.29 StrokeRectangle() [1/2]

```
void VectSharp.Graphics.StrokeRectangle (
            double leftX,
            double topY,
            double width,
            double height,
            Brush strokeColour,
            double lineWidth = 1,
            LineCaps lineCap = LineCaps.Butt,
            LineJoins lineJoin = LineJoins.Miter,
            LineDash?  lineDash = null,
            string tag = null )
```

Stroke a rectangle.

Parameters

leftX	The horizontal coordinate of the top-left corner of the rectangle.
topY	The vertical coordinate of the top-left corner of the rectangle.
width	The width of the rectangle.
height	The height of the rectangle.
strokeColour	The colour with which to stroke the rectangle.
lineWidth	The width of the line with which the rectangle is stroked.
lineCap	The line cap to use to stroke the rectangle.
lineJoin	The line join to use to stroke the rectangle.
lineDash	The line dash to use to stroke the rectangle.
tag	A tag to identify the filled rectangle.

Definition at line 441 of file Graphics.cs.

6.23.2.30 StrokeRectangle() [2/2]

```
void VectSharp.Graphics.StrokeRectangle (
            Point topLeft,
            Size size,
            Brush strokeColour,
            double lineWidth = 1,
            LineCaps lineCap = LineCaps.Butt,
            LineJoins lineJoin = LineJoins.Miter,
            LineDash?  lineDash = null,
            string tag = null )
```

Stroke a rectangle.

Parameters

topLeft	The top-left corner of the rectangle.
size	The size of the rectangle.
strokeColour	The colour with which to stroke the rectangle.
lineWidth	The width of the line with which the rectangle is stroked.
lineCap	The line cap to use to stroke the rectangle.
lineJoin	The line join to use to stroke the rectangle.
lineDash	The line dash to use to stroke the rectangle.
tag	A tag to identify the filled rectangle.

Definition at line 423 of file Graphics.cs.

6.23.2.31 StrokeText() [1/4]

```
void VectSharp.Graphics.StrokeText (
            double originX,
            double originY,
            IEnumerable< FormattedText > text,
            Brush strokeColour,
            TextBaselines textBaseline = TextBaselines.Top,
            double lineWidth = 1,
            LineCaps lineCap = LineCaps.Butt,
            LineJoins lineJoin = LineJoins.Miter,
            LineDash?  lineDash = null,
            string tag = null )
```

Stroke a formatted text string.

Parameters

originX	The horizontal coordinate of the text origin.

Parameters

originY	The vertical coordinate of the text origin. See *textBaseline* .
text	The FormattedText to draw.
strokeColour	The default Brush with which to stroke the text.
lineWidth	The width of the line with which the text is stroked.
lineCap	The line cap to use to stroke the text.
lineJoin	The line join to use to stroke the text.
lineDash	The line dash to use to stroke the text.
textBaseline	The text baseline (determines what *originY* represents).
tag	A tag to identify the stroked text.

Definition at line 517 of file Graphics.Text.cs.

6.23.2.32 StrokeText() [2/4]

```
void VectSharp.Graphics.StrokeText (
            double originX,
            double originY,
            string text,
            Font font,
            Brush strokeColour,
            TextBaselines textBaseline = TextBaselines.Top,
            double lineWidth = 1,
            LineCaps lineCap = LineCaps.Butt,
            LineJoins lineJoin = LineJoins.Miter,
            LineDash?  lineDash = null,
            string tag = null )
```

Stroke a text string.

Parameters

originX	The horizontal coordinate of the text origin.
originY	The vertical coordinate of the text origin. See *textBaseline* .
text	The string to draw.
font	The font with which to draw the text.
strokeColour	The Brush with which to stroke the text.
lineWidth	The width of the line with which the text is stroked.
lineCap	The line cap to use to stroke the text.
lineJoin	The line join to use to stroke the text.
lineDash	The line dash to use to stroke the text.
textBaseline	The text baseline (determines what *originY* represents).
tag	A tag to identify the stroked text.

Definition at line 89 of file Graphics.Text.cs.

6.23.2.33 StrokeText() [3/4]

```
void VectSharp.Graphics.StrokeText (
            Point origin,
            IEnumerable< FormattedText > text,
            Brush strokeColour,
            TextBaselines textBaseline = TextBaselines.Top,
            double lineWidth = 1,
            LineCaps lineCap = LineCaps.Butt,
            LineJoins lineJoin = LineJoins.Miter,
            LineDash?  lineDash = null,
            string tag = null )
```

Stroke a formatted text string.

Parameters

origin	The text origin. See *textBaseline* .
text	The FormattedText to draw.
strokeColour	The default Brush with which to stroke the text.
lineWidth	The width of the line with which the text is stroked.
lineCap	The line cap to use to stroke the text.
lineJoin	The line join to use to stroke the text.
lineDash	The line dash to use to stroke the text.
textBaseline	The text baseline (determines what the vertical component of *origin* represents).
tag	A tag to identify the stroked text.

Definition at line 420 of file Graphics.Text.cs.

6.23.2.34 StrokeText() [4/4]

```
void VectSharp.Graphics.StrokeText (
            Point origin,
            string text,
            Font font,
            Brush strokeColour,
            TextBaselines textBaseline = TextBaselines.Top,
            double lineWidth = 1,
            LineCaps lineCap = LineCaps.Butt,
            LineJoins lineJoin = LineJoins.Miter,
            LineDash?  lineDash = null,
            string tag = null )
```

Stroke a text string.

Parameters

origin	The text origin. See *textBaseline* .
text	The string to draw.
font	The font with which to draw the text.
strokeColour	The Brush with which to stroke the text.

Parameters

lineWidth	The width of the line with which the text is stroked.
lineCap	The line cap to use to stroke the text.
lineJoin	The line join to use to stroke the text.
lineDash	The line dash to use to stroke the text.
textBaseline	The text baseline (determines what the vertical component of *origin* represents).
tag	A tag to identify the stroked text.

Definition at line 70 of file Graphics.Text.cs.

6.23.2.35 StrokeTextOnPath()

```
void VectSharp.Graphics.StrokeTextOnPath (
              GraphicsPath path,
              string text,
              Font font,
              Brush strokeColour,
              double reference = 0,
              TextAnchors anchor = TextAnchors.Left,
              TextBaselines textBaseline = TextBaselines.Top,
              double lineWidth = 1,
              LineCaps lineCap = LineCaps.Butt,
              LineJoins lineJoin = LineJoins.Miter,
              LineDash?  lineDash = null,
              string tag = null )
```

Stroke a text string along a GraphicsPath.

Parameters

path	The GraphicsPath along which the text will flow.
text	The string to draw.
font	The font with which to draw the text.
strokeColour	The Brush with which to stroke the text.
lineWidth	The width of the line with which the text is stroked.
lineCap	The line cap to use to stroke the text.
lineJoin	The line join to use to stroke the text.
lineDash	The line dash to use to stroke the text.
reference	The (relative) starting point on the path starting from which the text should be drawn (0 is the start of the path, 1 is the end of the path).
anchor	The anchor in the text string that will correspond to the point specified by the *reference* .
textBaseline	The text baseline (determines which the position of the text in relation to the *path* .
tag	A tag to identify the stroked text.

Definition at line 211 of file Graphics.Text.cs.

6.23.2.36 Transform() [1/2]

```
void VectSharp.Graphics.Transform (
            double a,
            double b,
            double c,
            double d,
            double e,
            double f )
```

Transform the coordinate system with the specified transformation matrix [[a, c, e], [b, d, f], [0, 0, 1]].

Parameters

a	The first element of the first column.
b	The second element of the first column.
c	The first element of the second column.
d	The second element of the second column.
e	The first element of the third column.
f	The second element of the third column.

Definition at line 351 of file Graphics.cs.

6.23.2.37 Transform() [2/2]

```
Graphics VectSharp.Graphics.Transform (
            Func< Point, Point > transformationFunction,
            double linearisationResolution )
```

Creates a new Graphics object in which all the graphics actions have been transformed using an arbitrary transformation function. Raster images are replaced by grey rectangles.

Parameters

transformationFunction	An arbitrary transformation function.
linearisationResolution	The resolution that will be used to linearise curve segments.

Returns

A new Graphics object in which all graphics actions have been linearised and transformed using the *transformationFunction* .

Definition at line 885 of file Graphics.cs.

6.23.2.38 Translate() [1/2]

```
void VectSharp.Graphics.Translate (
            double x,
            double y )
```

Translate the coordinate system origin.

Parameters

x	The horizontal translation.
y	The vertical translation.

Definition at line 362 of file Graphics.cs.

6.23.2.39 Translate() [2/2]

```
void VectSharp.Graphics.Translate (
            Point delta )
```

Translate the coordinate system origin.

Parameters

delta	The new origin point.

Definition at line 371 of file Graphics.cs.

6.23.3 Property Documentation

6.23.3.1 UnbalancedStackAction

```
UnbalancedStackActions VectSharp.Graphics.UnbalancedStackAction = UnbalancedStackActions.Throw
[static], [get], [set]
```

Determines how an unbalanced graphics state stack (which occurs if the number of calls to Save and Restore is not equal) will be treated. The default is UnbalancedStackActions.Throw.

Definition at line 258 of file Graphics.cs.

The documentation for this class was generated from the following files:

- VectSharp/Graphics.cs
- VectSharp/Graphics.Text.cs

6.24 VectSharp.GraphicsPath Class Reference

Represents a graphics path that can be filled or stroked.

Public Member Functions

- GraphicsPath MoveTo (Point p)

  Move the current point without tracing a segment from the previous point.
- GraphicsPath MoveTo (double x, double y)

  Move the current point without tracing a segment from the previous point.
- GraphicsPath LineTo (Point p)

  Move the current point and trace a segment from the previous point.
- GraphicsPath LineTo (double x, double y)

  Move the current point and trace a segment from the previous point.
- GraphicsPath Arc (Point center, double radius, double startAngle, double endAngle)

  Trace an arc segment from a circle with the specified center and radius , starting at startAngle and ending at endAngle . The current point is updated to the end point of the arc.
- GraphicsPath Arc (double centerX, double centerY, double radius, double startAngle, double endAngle)

  Trace an arc segment from a circle with the specified center and radius , starting at startAngle and ending at endAngle . The current point is updated to the end point of the arc.
- GraphicsPath EllipticalArc (double radiusX, double radiusY, double axisAngle, bool largeArc, bool sweep↩ Clockwise, Point endPoint)

  Trace an arc from an ellipse with the specified radii, rotated by axisAngle with respect to the x-axis, starting at the current point and ending at the endPoint .
- GraphicsPath CubicBezierTo (Point control1, Point control2, Point endPoint)

  Trace a cubic Bezier curve from the current point to a destination point, with two control points. The current point is updated to the end point of the Bezier curve.
- GraphicsPath CubicBezierTo (double control1X, double control1Y, double control2X, double control2Y, double endPointX, double endPointY)

  Trace a cubic Bezier curve from the current point to a destination point, with two control points. The current point is updated to the end point of the Bezier curve.
- GraphicsPath Close ()

  Trace a segment from the current point to the start point of the figure and flag the figure as closed.
- GraphicsPath AddText (double originX, double originY, string text, Font font, TextBaselines text↩ Baseline=TextBaselines.Top)

  Add the contour of a text string to the current path.
- GraphicsPath AddText (Point origin, string text, Font font, TextBaselines textBaseline=TextBaselines.Top)

  Add the contour of a text string to the current path.
- GraphicsPath AddTextOnPath (GraphicsPath path, string text, Font font, double reference=0, TextAnchors anchor=TextAnchors.Left, TextBaselines textBaseline=TextBaselines.Top)

  Add the contour of a text string flowing along a GraphicsPath to the current path.
- GraphicsPath AddSmoothSpline (params Point[ ] points)

  Adds a smooth spline composed of cubic bezier segments that pass through the specified points.
- double MeasureLength ()

  Measures the length of the GraphicsPath.
- Point GetPointAtRelative (double position)

  Gets the point at the relative position specified on the GraphicsPath.
- Point GetPointAtAbsolute (double length)

  Gets the point at the absolute position specified on the GraphicsPath.
- Point GetTangentAtRelative (double position)

  Gets the tangent to the point at the relative position specified on the GraphicsPath.

- Point GetTangentAtAbsolute (double length)

  Gets the tangent to the point at the absolute position specified on the GraphicsPath.
- Point GetNormalAtAbsolute (double length)

  Gets the normal to the point at the absolute position specified on the GraphicsPath.
- Point GetNormalAtRelative (double position)

  Gets the normal to the point at the relative position specified on the GraphicsPath.
- GraphicsPath Linearise (double resolution)

  Linearises a GraphicsPath, replacing curve segments with series of line segments that approximate them.
- IEnumerable< List< Point > > GetPoints ()

  Gets a collection of the end points of all the segments in the GraphicsPath, divided by figure.
- IEnumerable< List< Point > > GetLinearisationPointsNormals (double resolution)

  Gets a collection of the tangents at the end point of the segments in which the GraphicsPath would be linearised, divided by figure.
- IEnumerable< GraphicsPath > Triangulate (double resolution, bool clockwise)

  Divides a GraphicsPath into triangles.
- GraphicsPath Transform (Func< Point, Point > transformationFunction)

  Transforms all of the Points in the GraphicsPath with an arbitrary transformation function.

Properties

- List< Segment > Segments = new List() `[get, set]`

  The segments that make up the path.

6.24.1 Detailed Description

Represents a graphics path that can be filled or stroked.

Definition at line 28 of file GraphicsPath.cs.

6.24.2 Member Function Documentation

6.24.2.1 AddSmoothSpline()

```
GraphicsPath VectSharp.GraphicsPath.AddSmoothSpline (
            params Point[] points )
```

Adds a smooth spline composed of cubic bezier segments that pass through the specified points.

Parameters

points	The points through which the spline should pass.

Returns

The GraphicsPath, to allow for chained calls.

Definition at line 479 of file GraphicsPath.cs.

6.24.2.2 AddText() [1/2]

```
GraphicsPath VectSharp.GraphicsPath.AddText (
            double originX,
            double originY,
            string text,
            Font font,
            TextBaselines textBaseline = TextBaselines.Top )
```

Add the contour of a text string to the current path.

Parameters

originX	The horizontal coordinate of the text origin.
originY	The vertical coordinate of the text origin. See *textBaseline* .
text	The string to draw.
font	The font with which to draw the text.
textBaseline	The text baseline (determines what *originY* represents).

///

Returns

The GraphicsPath, to allow for chained calls.

Definition at line 277 of file GraphicsPath.cs.

6.24.2.3 AddText() [2/2]

```
GraphicsPath VectSharp.GraphicsPath.AddText (
            Point origin,
            string text,
            Font font,
            TextBaselines textBaseline = TextBaselines.Top )
```

Add the contour of a text string to the current path.

Parameters

origin	The text origin. See *textBaseline* .
text	The string to draw.
font	The font with which to draw the text.
textBaseline	The text baseline (determines what the vertical component of *origin* represents).

Returns

The GraphicsPath, to allow for chained calls.

Definition at line 290 of file GraphicsPath.cs.

6.24.2.4 AddTextOnPath()

```
GraphicsPath VectSharp.GraphicsPath.AddTextOnPath (
            GraphicsPath path,
            string text,
            Font font,
            double reference = 0,
            TextAnchors anchor = TextAnchors.Left,
            TextBaselines textBaseline = TextBaselines.Top )
```

Add the contour of a text string flowing along a GraphicsPath to the current path.

Parameters

path	The GraphicsPath along which the text will flow.
text	The string to draw.
font	The font with which to draw the text.
reference	The (relative) starting point on the path starting from which the text should be drawn (0 is the start of the path, 1 is the end of the path).
anchor	The anchor in the text string that will correspond to the point specified by the *reference* .
textBaseline	The text baseline (determines which the position of the text in relation to the *path* .

Returns

The GraphicsPath, to allow for chained calls.

Definition at line 367 of file GraphicsPath.cs.

6.24.2.5 Arc() [1/2]

```
GraphicsPath VectSharp.GraphicsPath.Arc (
            double centerX,
            double centerY,
            double radius,
            double startAngle,
            double endAngle )
```

Trace an arc segment from a circle with the specified center and *radius* , starting at *startAngle* and ending at *endAngle* . The current point is updated to the end point of the arc.

Parameters

centerX	The horizontal coordinate of the center of the arc.
centerY	The vertical coordinate of the center of the arc.
radius	The radius of the arc.
startAngle	The start angle (in radians) of the arc.
endAngle	The end angle (in radians) of the arc.

Returns

The GraphicsPath, to allow for chained calls.

Definition at line 118 of file GraphicsPath.cs.

6.24.2.6 Arc() [2/2]

```
GraphicsPath VectSharp.GraphicsPath.Arc (
            Point center,
            double radius,
            double startAngle,
            double endAngle )
```

Trace an arc segment from a circle with the specified *center* and *radius* , starting at *startAngle* and ending at *endAngle* . The current point is updated to the end point of the arc.

Parameters

center	The center of the arc.
radius	The radius of the arc.
startAngle	The start angle (in radians) of the arc.
endAngle	The end angle (in radians) of the arc.

Returns

The GraphicsPath, to allow for chained calls.

Definition at line 98 of file GraphicsPath.cs.

6.24.2.7 Close()

```
GraphicsPath VectSharp.GraphicsPath.Close ( )
```

Trace a segment from the current point to the start point of the figure and flag the figure as closed.

Returns

The GraphicsPath, to allow for chained calls.

Definition at line 262 of file GraphicsPath.cs.

6.24.2.8 CubicBezierTo() [1/2]

```
GraphicsPath VectSharp.GraphicsPath.CubicBezierTo (
            double control1X,
            double control1Y,
            double control2X,
            double control2Y,
            double endPointX,
            double endPointY )
```

Trace a cubic Bezier curve from the current point to a destination point, with two control points. The current point is updated to the end point of the Bezier curve.

Parameters

control1X	The horizontal coordinate of the first control point.
control1Y	The vertical coordinate of the first control point.
control2X	The horizontal coordinate of the second control point.
control2Y	The vertical coordinate of the second control point.
endPointX	The horizontal coordinate of the destination point.
endPointY	The vertical coordinate of the destination point.

Returns

The GraphicsPath, to allow for chained calls.

Definition at line 252 of file GraphicsPath.cs.

6.24.2.9 CubicBezierTo() [2/2]

```
GraphicsPath VectSharp.GraphicsPath.CubicBezierTo (
            Point control1,
            Point control2,
            Point endPoint )
```

Trace a cubic Bezier curve from the current point to a destination point, with two control points. The current point is updated to the end point of the Bezier curve.

Parameters

control1	The first control point.
control2	The second control point.
endPoint	The destination point.

Returns

The GraphicsPath, to allow for chained calls.

Definition at line 231 of file GraphicsPath.cs.

6.24.2.10 EllipticalArc()

```
GraphicsPath VectSharp.GraphicsPath.EllipticalArc (
            double radiusX,
            double radiusY,
            double axisAngle,
            bool largeArc,
            bool sweepClockwise,
            Point endPoint )
```

Trace an arc from an ellipse with the specified radii, rotated by *axisAngle* with respect to the x-axis, starting at the current point and ending at the *endPoint* .

Parameters

radiusX	The horizontal radius of the ellipse.
radiusY	The vertical radius of the ellipse.
axisAngle	The angle of the horizontal axis of the ellipse with respect to the horizontal axis.
largeArc	Determines whether the large or the small arc is drawn.
sweepClockwise	Determines whether the clockwise or anticlockwise arc is drawn.
endPoint	The end point of the arc.

Returns

Definition at line 134 of file GraphicsPath.cs.

6.24.2.11 GetLinearisationPointsNormals()

```
IEnumerable<List<Point> > VectSharp.GraphicsPath.GetLinearisationPointsNormals (
            double resolution )
```

Gets a collection of the tangents at the end point of the segments in which the GraphicsPath would be linearised, divided by figure.

Parameters

resolution	The absolute length between successive samples in curve segments.

Returns

A collection of the tangents at the end point of the segments in which the GraphicsPath would be linearised, divided by figure.

Definition at line 1273 of file GraphicsPath.cs.

6.24.2.12 GetNormalAtAbsolute()

```
Point VectSharp.GraphicsPath.GetNormalAtAbsolute (
            double length )
```

Gets the normal to the point at the absolute position specified on the GraphicsPath.

Parameters

length	The distance to the point from the start of the GraphicsPath.

Returns

The normal to the point at the specified position.

Definition at line 1178 of file GraphicsPath.cs.

6.24.2.13 GetNormalAtRelative()

```
Point VectSharp.GraphicsPath.GetNormalAtRelative (
            double position )
```

Gets the normal to the point at the relative position specified on the GraphicsPath.

Parameters

position	The position on the GraphicsPath (0 is the start of the path, 1 is the end of the path).

Returns

The normal to the point at the specified position.

Definition at line 1189 of file GraphicsPath.cs.

6.24.2.14 GetPointAtAbsolute()

```
Point VectSharp.GraphicsPath.GetPointAtAbsolute (
            double length )
```

Gets the point at the absolute position specified on the GraphicsPath.

Parameters

length	The distance to the point from the start of the GraphicsPath.

Returns

The point at the specified position.

Definition at line 594 of file GraphicsPath.cs.

6.24.2.15 GetPointAtRelative()

```
Point VectSharp.GraphicsPath.GetPointAtRelative (
            double position )
```

Gets the point at the relative position specified on the GraphicsPath.

Parameters

position	The position on the GraphicsPath (0 is the start of the path, 1 is the end of the path).

Returns

The point at the specified position.

Definition at line 584 of file GraphicsPath.cs.

6.24.2.16 GetPoints()

```
IEnumerable<List<Point> > VectSharp.GraphicsPath.GetPoints ( )
```

Gets a collection of the end points of all the segments in the GraphicsPath, divided by figure.

Returns

A collection of the end points of all the segments in the GraphicsPath, divided by figure.

Definition at line 1228 of file GraphicsPath.cs.

6.24.2.17 GetTangentAtAbsolute()

```
Point VectSharp.GraphicsPath.GetTangentAtAbsolute (
            double length )
```

Gets the tangent to the point at the absolute position specified on the GraphicsPath.

Parameters

length	The distance to the point from the start of the GraphicsPath.

Returns

The tangent to the point at the specified position.

Definition at line 891 of file GraphicsPath.cs.

6.24.2.18 GetTangentAtRelative()

```
Point VectSharp.GraphicsPath.GetTangentAtRelative (
            double position )
```

Gets the tangent to the point at the relative position specified on the GraphicsPath.

Parameters

position	The position on the GraphicsPath (0 is the start of the path, 1 is the end of the path).

Returns

The tangent to the point at the specified position.

Definition at line 881 of file GraphicsPath.cs.

6.24.2.19 Linearise()

```
GraphicsPath VectSharp.GraphicsPath.Linearise (
            double resolution )
```

Linearises a GraphicsPath, replacing curve segments with series of line segments that approximate them.

Parameters

resolution	The absolute length between successive samples in curve segments.

Returns

A GraphicsPath composed only of linear segments that approximates the current GraphicsPath.

Definition at line 1200 of file GraphicsPath.cs.

6.24.2.20 LineTo() [1/2]

```
GraphicsPath VectSharp.GraphicsPath.LineTo (
            double x,
            double y )
```

Move the current point and trace a segment from the previous point.

Parameters

x	The horizontal coordinate of the new point.
y	The vertical coordinate of the new point.

Returns

The GraphicsPath, to allow for chained calls.

Definition at line 83 of file GraphicsPath.cs.

6.24.2.21 LineTo() [2/2]

```
GraphicsPath VectSharp.GraphicsPath.LineTo (
            Point p )
```

Move the current point and trace a segment from the previous point.

Parameters

p	The new point.

Returns

The GraphicsPath, to allow for chained calls.

Definition at line 64 of file GraphicsPath.cs.

6.24.2.22 MeasureLength()

```
double VectSharp.GraphicsPath.MeasureLength ( )
```

Measures the length of the GraphicsPath.

Returns

The length of the GraphicsPath

Definition at line 512 of file GraphicsPath.cs.

6.24.2.23 MoveTo() [1/2]

```
GraphicsPath VectSharp.GraphicsPath.MoveTo (
            double x,
            double y )
```

Move the current point without tracing a segment from the previous point.

Parameters

x	The horizontal coordinate of the new point.
y	The vertical coordinate of the new point.

Returns

The GraphicsPath, to allow for chained calls.

Definition at line 53 of file GraphicsPath.cs.

6.24.2.24 MoveTo() [2/2]

```
GraphicsPath VectSharp.GraphicsPath.MoveTo (
            Point p )
```

Move the current point without tracing a segment from the previous point.

Parameters

p	The new point.

Returns

The GraphicsPath, to allow for chained calls.

Definition at line 41 of file GraphicsPath.cs.

6.24.2.25 Transform()

```
GraphicsPath VectSharp.GraphicsPath.Transform (
            Func< Point, Point > transformationFunction )
```

Transforms all of the Points in the GraphicsPath with an arbitrary transformation function.

Parameters

transformationFunction	An arbitrary transformation function.

Returns

A new GraphicsPath in which all points have been replaced using the *transformationFunction* .

Definition at line 2345 of file GraphicsPath.cs.

6.24.2.26 Triangulate()

```
IEnumerable<GraphicsPath> VectSharp.GraphicsPath.Triangulate (
            double resolution,
            bool clockwise )
```

Divides a GraphicsPath into triangles.

Parameters

resolution	The resolution that will be used to linearise curve segments in the GraphicsPath.
clockwise	If this is `true`, the triangles will have their vertices in a clockwise order, otherwise they will be in anticlockwise order.

Returns

A collection of distinct GraphicsPaths, each representing one triangle.

Definition at line 1356 of file GraphicsPath.cs.

6.24.3 Property Documentation

6.24.3.1 Segments

```
List<Segment> VectSharp.GraphicsPath.Segments = new List<Segment>()  [get], [set]
```

The segments that make up the path.

Definition at line 33 of file GraphicsPath.cs.

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

- VectSharp/GraphicsPath.cs

6.25 VectSharp.Markdown.HTTPUtils Class Reference

Contains utilities to resolve absolute and relative URIs.

Static Public Attributes

- static string path

  Resolves an image Uri, by downloading the image file if necessary. It also takes care of ensuring that the file extension matches the format of the file.

Properties

- static bool LogDownloads = true `[get, set]`

  Determines whether every file that is downloaded should be logged to the standard error stream.

6.25.1 Detailed Description

Contains utilities to resolve absolute and relative URIs.

Definition at line 244 of file HtmlTag.cs.

6.25.2 Member Data Documentation

6.25.2.1 path

`string VectSharp.Markdown.HTTPUtils.path [static]`

Resolves an image Uri, by downloading the image file if necessary. It also takes care of ensuring that the file extension matches the format of the file.

Parameters

uri	The address of the image.
baseUriString	The base uri to use for relative uris.

Returns

A tuple containing the local path of the image file (either the original image, or a local copy of a remote file) and a boolean value indicating whether the image was fetched from a remote location and should be deleted after the program is done with it.

Definition at line 257 of file HtmlTag.cs.

6.25.3 Property Documentation

6.25.3.1 LogDownloads

```
bool VectSharp.Markdown.HTTPUtils.LogDownloads = true  [static], [get], [set]
```

Determines whether every file that is downloaded should be logged to the standard error stream.

Definition at line 249 of file HtmlTag.cs.

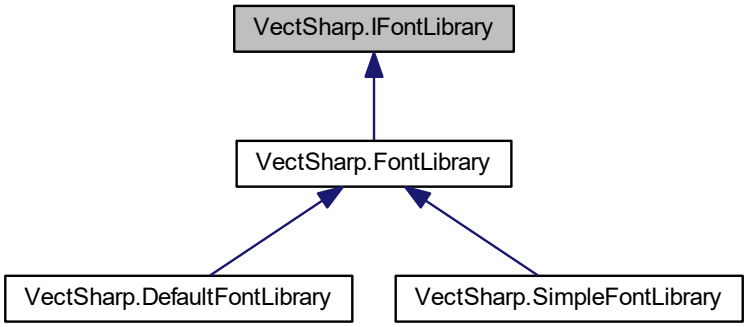
The documentation for this class was generated from the following file:

- VectSharp.Markdown/HtmlTag.cs

6.26 VectSharp.IFontLibrary Interface Reference

Represents a font library with methods to create FontFamily objects from a string or from FontFamily.StandardFontFamilies.

Inheritance diagram for VectSharp.IFontLibrary:



Public Member Functions

- FontFamily ResolveFontFamily (string fontFamily)

  Create a new font family from the specified family name or true type file. If the family name or the true type file are not valid, an exception might be raised.
- FontFamily ResolveFontFamily (string fontFamily, params string[ ] fallback)

  Create a new font family from the specified family name or true type file. If the family name or the true type file are not valid, try to instantiate the font family using the fallback . If none of the fallback family names or true type files are valid, an exception might be raised.
- FontFamily ResolveFontFamily (FontFamily.StandardFontFamilies standardFontFamily)

  Create a new font family from the specified standard font family name.
- FontFamily ResolveFontFamily (string fontFamily, FontFamily.StandardFontFamilies finalFallback, params string[ ] fallback)

  Create a new font family from the specified family name or true type file. If the family name or the true type file are not valid, try to instantiate the font family using the fallback . If none of the fallback family names or true type files are valid, instantiate a standard font family using the finalFallback .

6.26.1 Detailed Description

Represents a font library with methods to create FontFamily objects from a string or from FontFamily.StandardFontFamilies.

Definition at line 29 of file FontLibrary.cs.

6.26.2 Member Function Documentation

6.26.2.1 ResolveFontFamily() [1/4]

```
FontFamily VectSharp.IFontLibrary.ResolveFontFamily (
            FontFamily.StandardFontFamilies standardFontFamily )
```

Create a new font family from the specified standard font family name.

Parameters

standardFontFamily	The standard name of the font family.

Returns

A FontFamily object corresponding to the specified font family.

Implemented in VectSharp.DefaultFontLibrary, VectSharp.SimpleFontLibrary, and VectSharp.FontLibrary.

6.26.2.2 ResolveFontFamily() [2/4]

```
FontFamily VectSharp.IFontLibrary.ResolveFontFamily (
            string fontFamily )
```

Create a new font family from the specified family name or true type file. If the family name or the true type file are not valid, an exception might be raised.

Parameters

fontFamily	The name of the font family to create, or the path to a TTF file.

Returns

If the font family name or the true type file is valid, a FontFamily object corresponding to the specified font family.

Implemented in VectSharp.DefaultFontLibrary, VectSharp.SimpleFontLibrary, and VectSharp.FontLibrary.

6.26.2.3 ResolveFontFamily() [3/4]

```
FontFamily VectSharp.IFontLibrary.ResolveFontFamily (
            string fontFamily,
            FontFamily.StandardFontFamilies finalFallback,
            params string[] fallback )
```

Create a new font family from the specified family name or true type file. If the family name or the true type file are not valid, try to instantiate the font family using the *fallback* . If none of the fallback family names or true type files are valid, instantiate a standard font family using the *finalFallback* .

Parameters

fontFamily	The name of the font family to create, or the path to a TTF file.
fallback	Names of additional font families or TTF files, which will be tried if the first *fontFamily* is not valid.
finalFallback	The standard name of the font family that will be used if none of the fallback families are valid.

Returns

A FontFamily object corresponding to the first of the specified font families that is valid.

Implemented in VectSharp.FontLibrary.

6.26.2.4 ResolveFontFamily() [4/4]

```
FontFamily VectSharp.IFontLibrary.ResolveFontFamily (
            string fontFamily,
            params string[] fallback )
```

Create a new font family from the specified family name or true type file. If the family name or the true type file are not valid, try to instantiate the font family using the *fallback* . If none of the fallback family names or true type files are valid, an exception might be raised.

Parameters

fontFamily	The name of the font family to create, or the path to a TTF file.
fallback	Names of additional font families or TTF files, which will be tried if the first *fontFamily* is not valid.

Returns

A FontFamily object corresponding to the first of the specified font families that is valid.

Implemented in VectSharp.FontLibrary.

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

- VectSharp/FontLibrary.cs

6.27 VectSharp.IGraphicsContext Interface Reference

This interface should be implemented by classes intended to provide graphics output capability to a Graphics object.

Public Member Functions

- void Save ()

  Save the current transform state (rotation, translation, scale). This should be implemented as a LIFO stack.
- void Restore ()

  Restore the previous transform state (rotation, translation, scale). This should be implemented as a LIFO stack.
- void Translate (double x, double y)

  Translate the coordinate system origin.
- void Rotate (double angle)

  Rotate the coordinate system around the origin.
- void Scale (double scaleX, double scaleY)

  Scale the coordinate system with respect to the origin.
- void Transform (double a, double b, double c, double d, double e, double f)

  Transform the coordinate system with the specified transformation matrix [[a, c, e], [b, d, f], [0, 0, 1]].
- void FillText (string text, double x, double y)

  Fill a text string using the current Font and TextBaseline.
- void StrokeText (string text, double x, double y)

  Stroke the outline of a text string using the current Font and TextBaseline.
- void MoveTo (double x, double y)

  Change the current point without drawing a line from the previous point. If necessary, start a new figure.
- void LineTo (double x, double y)

  Draw a line from the previous point to the specified point.
- void Close ()

  Close the current figure.
- void Stroke ()

  Stroke the current path using the current StrokeStyle, LineWidth, LineCap, LineJoin and LineDash.
- void SetClippingPath ()

  Set the current clipping path as the intersection of the previous clipping path and the current path.
- void SetFillStyle ((int r, int g, int b, double a) style)

  Set the current FillStyle.
- void SetFillStyle (Brush style)

  Set the current FillStyle.
- void SetStrokeStyle ((int r, int g, int b, double a) style)

  Set the current StrokeStyle.
- void SetStrokeStyle (Brush style)

  Set the current StrokeStyle.
- void CubicBezierTo (double p1X, double p1Y, double p2X, double p2Y, double p3X, double p3Y)

  Add to the current figure a cubic Bezier from the current point to a destination point, with two control points.
- void Rectangle (double x0, double y0, double width, double height)

  Add a rectangle figure to the current path.
- void Fill ()

  Fill the current path using the current FillStyle.
- void SetLineDash (LineDash dash)

  Set the current line dash pattern.
- void DrawRasterImage (int sourceX, int sourceY, int sourceWidth, int sourceHeight, double destinationX, double destinationY, double destinationWidth, double destinationHeight, RasterImage image)

  Draw a raster image.

Properties

- double Width [get]
  Width of the graphic surface.
- double Height [get]
  Height of the graphic surface.
- Font Font [get, set]
  The current font.
- TextBaselines TextBaseline [get, set]
  The current text baseline.
- Brush FillStyle [get]
  Current brush used to fill paths.
- Brush StrokeStyle [get]
  Current brush used to stroke paths.
- double LineWidth [get, set]
  Current line width used to stroke paths.
- LineCaps LineCap [set]
  Current line cap used to stroke paths.
- LineJoins LineJoin [set]
  Current line join used to stroke paths.
- string Tag [get, set]
  The current tag. How this can be used depends on each implementation.

6.27.1 Detailed Description

This interface should be implemented by classes intended to provide graphics output capability to a Graphics object.

Definition at line 34 of file Graphics.cs.

6.27.2 Member Function Documentation

6.27.2.1 Close()

```
void VectSharp.IGraphicsContext.Close ( )
```

Close the current figure.

6.27.2.2 CubicBezierTo()

```
void VectSharp.IGraphicsContext.CubicBezierTo (
            double p1X,
            double p1Y,
            double p2X,
            double p2Y,
            double p3X,
            double p3Y )
```

Add to the current figure a cubic Bezier from the current point to a destination point, with two control points.

Parameters

p1X	The horizontal coordinate of the first control point.
p1Y	The vertical coordinate of the first control point.
p2X	The horizontal coordinate of the second control point.
p2Y	The vertical coordinate of the second control point.
p3X	The horizontal coordinate of the destination point.
p3Y	The vertical coordinate of the destination point.

6.27.2.3 DrawRasterImage()

```
void VectSharp.IGraphicsContext.DrawRasterImage (
            int sourceX,
            int sourceY,
            int sourceWidth,
            int sourceHeight,
            double destinationX,
            double destinationY,
            double destinationWidth,
            double destinationHeight,
            RasterImage image )
```

Draw a raster image.

Parameters

sourceX	The horizontal coordinate of the top-left corner of the rectangle delimiting the source area of the image.
sourceY	The vertical coordinate of the top-left corner of the rectangle delimiting the source area of the image.
sourceWidth	The width of the rectangle delimiting the source area of the image.
sourceHeight	The height of the rectangle delimiting the source area of the image.
destinationX	The horizontal coordinate of the top-left corner of the rectangle delimiting the destination area of the image.
destinationY	The vertical coordinate of the top-left corner of the rectangle delimiting the destination area of the image.
destinationWidth	The width of the rectangle delimiting the destination area of the image.
destinationHeight	The height of the rectangle delimiting the destination area of the image.
image	The image to draw.

6.27.2.4 Fill()

```
void VectSharp.IGraphicsContext.Fill ( )
```

Fill the current path using the current FillStyle.

6.27.2.5 FillText()

```
void VectSharp.IGraphicsContext.FillText (
            string text,
            double x,
            double y )
```

Fill a text string using the current Font and TextBaseline.

Parameters

text	The string to draw.
x	The horizontal coordinate of the text origin.
y	The vertical coordinate of the text origin.

6.27.2.6 LineTo()

```
void VectSharp.IGraphicsContext.LineTo (
            double x,
            double y )
```

Draw a line from the previous point to the specified point.

Parameters

x	The horizontal coordinate of the point.
y	The vertical coordinate of the point.

6.27.2.7 MoveTo()

```
void VectSharp.IGraphicsContext.MoveTo (
            double x,
            double y )
```

Change the current point without drawing a line from the previous point. If necessary, start a new figure.

Parameters

x	The horizontal coordinate of the point.
y	The vertical coordinate of the point.

6.27.2.8 Rectangle()

```
void VectSharp.IGraphicsContext.Rectangle (
            double x0,
            double y0,
            double width,
            double height )
```

Add a rectangle figure to the current path.

Parameters

x0	The horizontal coordinate of the top-left corner of the rectangle.
y0	The vertical coordinate of the top-left corner of the rectangle.
width	The width of corner of the rectangle.
height	The height of corner of the rectangle.

6.27.2.9 Restore()

```
void VectSharp.IGraphicsContext.Restore ( )
```

Restore the previous transform state (rotation, translation, scale). This should be implemented as a LIFO stack.

6.27.2.10 Rotate()

```
void VectSharp.IGraphicsContext.Rotate (
            double angle )
```

Rotate the coordinate system around the origin.

Parameters

angle	The angle (in radians) by which to rotate the coordinate system.

6.27.2.11 Save()

```
void VectSharp.IGraphicsContext.Save ( )
```

Save the current transform state (rotation, translation, scale). This should be implemented as a LIFO stack.

6.27.2.12 Scale()

```
void VectSharp.IGraphicsContext.Scale (
            double scaleX,
            double scaleY )
```

Scale the coordinate system with respect to the origin.

Parameters

scaleX	The horizontal scale.
scaleY	The vertical scale.

6.27.2.13 SetClippingPath()

```
void VectSharp.IGraphicsContext.SetClippingPath ( )
```

Set the current clipping path as the intersection of the previous clipping path and the current path.

6.27.2.14 SetFillStyle() [1/2]

```
void VectSharp.IGraphicsContext.SetFillStyle (
            (int r, int g, int b, double a) style )
```

Set the current FillStyle.

Parameters

style	A ValueTuple<Int32, Int32, Int32, Double> containing component information for the colour. For r, g, and b, range: [0, 255]; for a, range: [0, 1].

6.27.2.15 SetFillStyle() [2/2]

```
void VectSharp.IGraphicsContext.SetFillStyle (
            Brush style )
```

Set the current FillStyle.

Parameters

style	The new fill style.

6.27.2.16 SetLineDash()

```
void VectSharp.IGraphicsContext.SetLineDash (
            LineDash dash )
```

Set the current line dash pattern.

Parameters

dash	The line dash pattern.

6.27.2.17 SetStrokeStyle() [1/2]

```
void VectSharp.IGraphicsContext.SetStrokeStyle (
            (int r, int g, int b, double a) style )
```

Set the current StrokeStyle.

Parameters

style	A ValueTuple<Int32, Int32, Int32, Double> containing component information for the colour. For r, g, and b, range: [0, 255]; for a, range: [0, 1].

6.27.2.18 SetStrokeStyle() [2/2]

```
void VectSharp.IGraphicsContext.SetStrokeStyle (
            Brush style )
```

Set the current StrokeStyle.

Parameters

style	The new stroke style.

6.27.2.19 Stroke()

```
void VectSharp.IGraphicsContext.Stroke ( )
```

Stroke the current path using the current StrokeStyle, LineWidth, LineCap, LineJoin and LineDash.

6.27.2.20 StrokeText()

```
void VectSharp.IGraphicsContext.StrokeText (
            string text,
            double x,
            double y )
```

Stroke the outline of a text string using the current Font and TextBaseline.

Parameters

text	The string to draw.
x	The horizontal coordinate of the text origin.
y	The vertical coordinate of the text origin.

6.27.2.21 Transform()

```
void VectSharp.IGraphicsContext.Transform (
            double a,
            double b,
            double c,
            double d,
            double e,
            double f )
```

Transform the coordinate system with the specified transformation matrix [[a, c, e], [b, d, f], [0, 0, 1]].

Parameters

a	The first element of the first column.
b	The second element of the first column.
c	The first element of the second column.
d	The second element of the second column.
e	The first element of the third column.
f	The second element of the third column.

6.27.2.22 Translate()

```
void VectSharp.IGraphicsContext.Translate (
            double x,
            double y )
```

Translate the coordinate system origin.

Parameters

x	The horizontal translation.
y	The vertical translation.

6.27.3 Property Documentation

6.27.3.1 FillStyle

```
Brush VectSharp.IGraphicsContext.FillStyle  [get]
```

Current brush used to fill paths.

Definition at line 145 of file Graphics.cs.

6.27.3.2 Font

```
Font VectSharp.IGraphicsContext.Font  [get], [set]
```

The current font.

Definition at line 90 of file Graphics.cs.

6.27.3.3 Height

```
double VectSharp.IGraphicsContext.Height  [get]
```

Height of the graphic surface.

Definition at line 44 of file Graphics.cs.

6.27.3.4 LineCap

```
LineCaps VectSharp.IGraphicsContext.LineCap  [set]
```

Current line cap used to stroke paths.

Definition at line 209 of file Graphics.cs.

6.27.3.5 LineJoin

```
LineJoins VectSharp.IGraphicsContext.LineJoin  [set]
```

Current line join used to stroke paths.

Definition at line 214 of file Graphics.cs.

6.27.3.6 LineWidth

```
double VectSharp.IGraphicsContext.LineWidth  [get], [set]
```

Current line width used to stroke paths.

Definition at line 204 of file Graphics.cs.

6.27.3.7 StrokeStyle

```
Brush VectSharp.IGraphicsContext.StrokeStyle  [get]
```

Current brush used to stroke paths.

Definition at line 162 of file Graphics.cs.

6.27.3.8 Tag

```
string VectSharp.IGraphicsContext.Tag  [get], [set]
```

The current tag. How this can be used depends on each implementation.

Definition at line 225 of file Graphics.cs.

6.27.3.9 TextBaseline

```
TextBaselines VectSharp.IGraphicsContext.TextBaseline  [get], [set]
```

The current text baseline.

Definition at line 95 of file Graphics.cs.

6.27.3.10 Width

```
double VectSharp.IGraphicsContext.Width  [get]
```

Width of the graphic surface.

Definition at line 39 of file Graphics.cs.

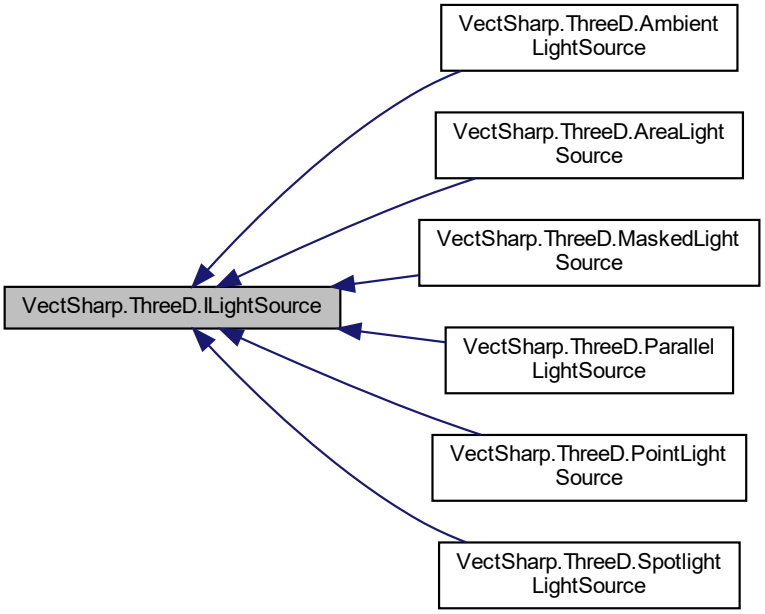
The documentation for this interface was generated from the following file:

- VectSharp/Graphics.cs

6.28 VectSharp.ThreeD.ILightSource Interface Reference

Represents a light source.

Inheritance diagram for VectSharp.ThreeD.ILightSource:



Public Member Functions

- LightIntensity GetLightAt (Point3D point)

  Computes the light intensity at the specified point, without taking into account any obstructions.
- double GetObstruction (Point3D point, IEnumerable< Triangle3DElement > shadowingTriangles)

  Determines the amount of obstruction of the light that results at point due to the specified shadowingTriangles .

Properties

- bool CastsShadow `[get]`

  Determines whether the light casts a shadow or not.

6.28.1 Detailed Description

Represents a light source.

Definition at line 65 of file Lights.cs.

6.28.2 Member Function Documentation

6.28.2.1 GetLightAt()

```
LightIntensity VectSharp.ThreeD.ILightSource.GetLightAt (
            Point3D point )
```

Computes the light intensity at the specified point, without taking into account any obstructions.

Parameters

point	The Point3DElement at which the light intensity should be computed.

Returns

Implemented in VectSharp.ThreeD.AreaLightSource, VectSharp.ThreeD.MaskedLightSource, VectSharp.ThreeD.SpotlightLightSource, VectSharp.ThreeD.PointLightSource, VectSharp.ThreeD.ParallelLightSource, and VectSharp.ThreeD.AmbientLightSource.

6.28.2.2 GetObstruction()

```
double VectSharp.ThreeD.ILightSource.GetObstruction (
            Point3D point,
            IEnumerable< Triangle3DElement > shadowingTriangles )
```

Determines the amount of obstruction of the light that results at *point* due to the specified *shadowingTriangles* .

Parameters

point	The Point3D at which the obstruction should be computed.
shadowingTriangles	A collection of Triangle3DElement casting shadows.

Returns

1 if the light is completely obstructed, 0 if the light is completely visible, a value between these if the light is partially obstructed.

Implemented in VectSharp.ThreeD.AreaLightSource, VectSharp.ThreeD.MaskedLightSource, VectSharp.ThreeD.SpotlightLightSource, VectSharp.ThreeD.PointLightSource, VectSharp.ThreeD.ParallelLightSource, and VectSharp.ThreeD.AmbientLightSource.

6.28.3 Property Documentation

6.28.3.1 CastsShadow

```
bool VectSharp.ThreeD.ILightSource.CastsShadow  [get]
```

Determines whether the light casts a shadow or not.

Definition at line 77 of file Lights.cs.

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

- VectSharp.ThreeD/Lights.cs

6.29 VectSharp.MuPDFUtils.ImageURIParser Class Reference

Provides a method to parse an image URI into a page.

Static Public Member Functions

- static Func< string, bool, Page > Parser (Func< string, bool, Page > parseSVG)

  Parses an image URI into a page. This is intended to replace the default image URI interpreter in VectSharp.SVG.Parser.ParseImageURI. To do this, use something like:

6.29.1 Detailed Description

Provides a method to parse an image URI into a page.

Definition at line 29 of file ImageURIParser.cs.

6.29.2 Member Function Documentation

6.29.2.1 Parser()

```
static Func<string, bool, Page> VectSharp.MuPDFUtils.ImageURIParser.Parser (
            Func< string, bool, Page > parseSVG )  [static]
```

Parses an image URI into a page. This is intended to replace the default image URI interpreter in VectSharp.SVG.Parser.ParseImageURI. To do this, use something like:

```
VectSharp.SVG.Parser.ParseImageURI = VectSharp.MuPDFUtils.ImageURIParser.Parser(VectShar
```

Parameters

parseSVG	A function to parse an SVG image uri into a page. You should pass VectSharp.SVG.Parser.ParseSVGURI as this argument.

Returns

A function to parse an image URI into a page.

Definition at line 37 of file ImageURIParser.cs.

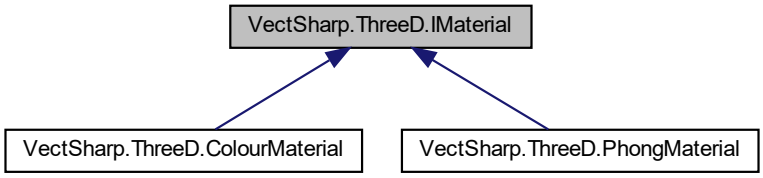
The documentation for this class was generated from the following file:

- VectSharp.MuPDFUtils/ImageURIParser.cs

6.30 VectSharp.ThreeD.IMaterial Interface Reference

Represents a material used to the determine the appearance of Triangle3DElement.

Inheritance diagram for VectSharp.ThreeD.IMaterial:



Public Member Functions

- Colour GetColour (Point3D point, NormalizedVector3D surfaceNormal, Camera camera, IList< ILightSource > lights, IList< double > obstructions)

  Obtains the Colour at the specified point.

6.30.1 Detailed Description

Represents a material used to the determine the appearance of Triangle3DElement.

Definition at line 31 of file Materials.cs.

6.30.2 Member Function Documentation

6.30.2.1 GetColour()

```
Colour VectSharp.ThreeD.IMaterial.GetColour (
            Point3D point,
            NormalizedVector3D surfaceNormal,
            Camera camera,
            IList< ILightSource > lights,
            IList< double > obstructions )
```

Obtains the Colour at the specified point.

Parameters

point	The point whose colour should be determined.
surfaceNormal	The normal to the surface at the specified *point* .
camera	The camera being used to render the scene.
lights	A list of light sources that are present in the scene.
obstructions	A list of values indicating how obstructed each light source is.

Returns

The Colour of the specified point.

Implemented in VectSharp.ThreeD.PhongMaterial, and VectSharp.ThreeD.ColourMaterial.

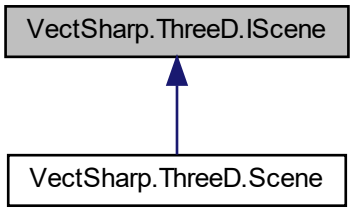
The documentation for this interface was generated from the following file:

- VectSharp.ThreeD/Materials.cs

6.31 VectSharp.ThreeD.IScene Interface Reference

Represents a 3D scene.

Inheritance diagram for VectSharp.ThreeD.IScene:



Public Member Functions

- void AddElement (Element3D element)

  Adds the specified element to the scene.
- void AddRange (IEnumerable< Element3D > elements)

  Adds the specified elements to the scene.
- void Replace (Func< Element3D, Element3D > replacementFunction)

  Replaces each element in the scene with the element returned by the replacementFunction .
- void Replace (Func< Element3D, IEnumerable< Element3D >> replacementFunction)

  Replaces each element in the scene with the element(s) returned by the replacementFunction .

Properties

- IEnumerable< Element3D > SceneElements [get]

  The Element3Ds constituting the scene.
- object SceneLock [get]

  An object used to synchronise multithreaded rendering of the same scene.

6.31.1 Detailed Description

Represents a 3D scene.

Definition at line 26 of file Scene.cs.

6.31.2 Member Function Documentation

6.31.2.1 AddElement()

```
void VectSharp.ThreeD.IScene.AddElement (
            Element3D element )
```

Adds the specified *element* to the scene.

Parameters

element	The Element3D to add.

Implemented in VectSharp.ThreeD.Scene.

6.31.2.2 AddRange()

```
void VectSharp.ThreeD.IScene.AddRange (
            IEnumerable< Element3D > elements )
```

Adds the specified *elements* to the scene.

Parameters

elements	A collection of Element3Ds to add.

Implemented in VectSharp.ThreeD.Scene.

6.31.2.3 Replace() [1/2]

```
void VectSharp.ThreeD.IScene.Replace (
            Func< Element3D, Element3D > replacementFunction )
```

Replaces each element in the scene with the element returned by the *replacementFunction* .

Parameters

replacementFunction	A function replacing each Element3D in the scene with another Element3D.

Implemented in VectSharp.ThreeD.Scene.

6.31.2.4 Replace() [2/2]

```
void VectSharp.ThreeD.IScene.Replace (
            Func< Element3D, IEnumerable< Element3D >> replacementFunction )
```

Replaces each element in the scene with the element(s) returned by the *replacementFunction* .

Parameters

replacementFunction	A function replacing each Element3D in the scene with 0 or more Element3Ds.

Implemented in VectSharp.ThreeD.Scene.

6.31.3 Property Documentation

6.31.3.1 SceneElements

```
IEnumerable<Element3D> VectSharp.ThreeD.IScene.SceneElements  [get]
```

The Element3Ds constituting the scene.

Definition at line 31 of file Scene.cs.

6.31.3.2 SceneLock

```
object VectSharp.ThreeD.IScene.SceneLock  [get]
```

An object used to synchronise multithreaded rendering of the same scene.

Definition at line 60 of file Scene.cs.

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

- VectSharp.ThreeD/Scene.cs

6.32 VectSharp.ThreeD.LightIntensity Struct Reference

Represents the intensity of a light source at a particular point.

Public Member Functions

- LightIntensity (double intensity, NormalizedVector3D direction)

  Creates a new LightIntensity.
- void Deconstruct (out double intensity, out NormalizedVector3D direction)

  Deconstructs the struct.

Public Attributes

- double Intensity

  The intensity of the light.
- NormalizedVector3D Direction

  The direction towards from which the light comes.

6.32.1 Detailed Description

Represents the intensity of a light source at a particular point.

Definition at line 27 of file Lights.cs.

6.32.2 Constructor & Destructor Documentation

6.32.2.1 LightIntensity()

```
VectSharp.ThreeD.LightIntensity.LightIntensity (
            double intensity,
            NormalizedVector3D direction )
```

Creates a new LightIntensity.

Parameters

intensity	The intensity of the light.
direction	The direction from which the light comes.

Definition at line 44 of file Lights.cs.

6.32.3 Member Function Documentation

6.32.3.1 Deconstruct()

```
void VectSharp.ThreeD.LightIntensity.Deconstruct (
            out double intensity,
            out NormalizedVector3D direction )
```

Deconstructs the struct.

Parameters

intensity	This parameter will hold the Intensity of the light.
direction	This parameter will hold the Direction of the light.

Definition at line 55 of file Lights.cs.

6.32.4 Member Data Documentation

6.32.4.1 Direction

```
NormalizedVector3D VectSharp.ThreeD.LightIntensity.Direction
```

The direction towards from which the light comes.

Definition at line 37 of file Lights.cs.

6.32.4.2 Intensity

```
double VectSharp.ThreeD.LightIntensity.Intensity
```

The intensity of the light.

Definition at line 32 of file Lights.cs.

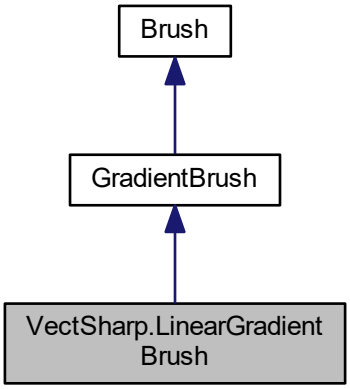
The documentation for this struct was generated from the following file:

- VectSharp.ThreeD/Lights.cs

6.33 VectSharp.LinearGradientBrush Class Reference

Represents a brush painting with a linear gradient.

Inheritance diagram for VectSharp.LinearGradientBrush:



Public Member Functions

- LinearGradientBrush (Point startPoint, Point endPoint, IEnumerable< GradientStop > gradientStops)

  Creates a new LinearGradientBrush with the specified start point, end point and gradient stops.
- LinearGradientBrush (Point startPoint, Point endPoint, params GradientStop[ ] gradientStops)

  Creates a new LinearGradientBrush with the specified start point, end point and gradient stops.
- LinearGradientBrush RelativeTo (Graphics referenceGraphics)

  Returns a LinearGradientBrush with the same gradient stops as the current instance, whose start and end point correspond to the points of the current instance in the original reference frame of the referenceGraphics . This involves computing the current transform matrix of the referenceGraphics , inverting it, and applying the inverse matrix to the StartPoint and EndPoint of the current instance.
- override Brush MultiplyOpacity (double opacity)

  Returns a brush corresponding the current instance, with the specified opacity multiplication applied.

Properties

- Point StartPoint `[get]`

  The starting point of the gradient. Note that this is relative to the current coordinate system when the gradient is used.
- Point EndPoint `[get]`

  The end point of the gradient. Note that this is relative to the current coordinate system when the gradient is used.

Additional Inherited Members

6.33.1 Detailed Description

Represents a brush painting with a linear gradient.

Definition at line 244 of file Brush.cs.

6.33.2 Constructor & Destructor Documentation

6.33.2.1 LinearGradientBrush() [1/2]

```
VectSharp.LinearGradientBrush.LinearGradientBrush (
            Point startPoint,
            Point endPoint,
            IEnumerable< GradientStop > gradientStops )
```

Creates a new LinearGradientBrush with the specified start point, end point and gradient stops.

Parameters

startPoint	The starting point of the gradient. Note that this is relative to the current coordinate system when the gradient is used.
endPoint	The ending point of the gradient. Note that this is relative to the current coordinate system when the gradient is used.
gradientStops	The colour stops in the gradient.

Definition at line 262 of file Brush.cs.

6.33.2.2 LinearGradientBrush() [2/2]

```
VectSharp.LinearGradientBrush.LinearGradientBrush (
            Point startPoint,
            Point endPoint,
            params GradientStop[] gradientStops )
```

Creates a new LinearGradientBrush with the specified start point, end point and gradient stops.

Parameters

startPoint	The starting point of the gradient. Note that this is relative to the current coordinate system when the gradient is used.
endPoint	The ending point of the gradient. Note that this is relative to the current coordinate system when the gradient is used.
gradientStops	The colour stops in the gradient.

Definition at line 276 of file Brush.cs.

6.33.3 Member Function Documentation

6.33.3.1 RelativeTo()

```
LinearGradientBrush VectSharp.LinearGradientBrush.RelativeTo (
            Graphics referenceGraphics )
```

Returns a LinearGradientBrush with the same gradient stops as the current instance, whose start and end point correspond to the points of the current instance in the original reference frame of the *referenceGraphics* . This involves computing the current transform matrix of the *referenceGraphics* , inverting it, and applying the inverse matrix to the StartPoint and EndPoint of the current instance.

Parameters

referenceGraphics	The Graphics whose original reference frame is to be used.

Returns

A LinearGradientBrush with the same gradient stops as the current instance, whose start and end point correspond to the points of the current instance in the original reference frame of the *referenceGraphics* .

Definition at line 308 of file Brush.cs.

6.33.4 Property Documentation

6.33.4.1 EndPoint

```
Point VectSharp.LinearGradientBrush.EndPoint  [get]
```

The end point of the gradient. Note that this is relative to the current coordinate system when the gradient is used.

Definition at line 254 of file Brush.cs.

6.33.4.2 StartPoint

```
Point VectSharp.LinearGradientBrush.StartPoint  [get]
```

The starting point of the gradient. Note that this is relative to the current coordinate system when the gradient is used.

Definition at line 249 of file Brush.cs.

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

- VectSharp/Brush.cs

6.34 VectSharp.LineDash Struct Reference

Represents instructions on how to paint a dashed line.

Public Member Functions

- LineDash (double unitsOn, double unitsOff, double phase)

  Define a new line dash pattern.

Public Attributes

- double UnitsOn

  Length of the "on" (painted) segment.
- double UnitsOff

  Length of the "off" (not painted) segment.
- double Phase

  Position in the dash pattern at which the line starts.

Static Public Attributes

- static LineDash SolidLine = new LineDash(0, 0, 0)

  A solid (not dashed) line

6.34.1 Detailed Description

Represents instructions on how to paint a dashed line.

Definition at line 112 of file Enums.cs.

6.34.2 Constructor & Destructor Documentation

6.34.2.1 LineDash()

```
VectSharp.LineDash.LineDash (
            double unitsOn,
            double unitsOff,
            double phase )
```

Define a new line dash pattern.

Parameters

unitsOn	The length of the "on" (painted) segment.
unitsOff	The length of the "off" (not painted) segment.
phase	The position in the dash pattern at which the line starts.

Definition at line 140 of file Enums.cs.

6.34.3 Member Data Documentation

6.34.3.1 Phase

```
double VectSharp.LineDash.Phase
```

Position in the dash pattern at which the line starts.

Definition at line 132 of file Enums.cs.

6.34.3.2 SolidLine

```
LineDash VectSharp.LineDash.SolidLine = new LineDash(0, 0, 0)  [static]
```

A solid (not dashed) line

Definition at line 117 of file Enums.cs.

6.34.3.3 UnitsOff

```
double VectSharp.LineDash.UnitsOff
```

Length of the "off" (not painted) segment.

Definition at line 127 of file Enums.cs.

6.34.3.4 UnitsOn

```
double VectSharp.LineDash.UnitsOn
```

Length of the "on" (painted) segment.

Definition at line 122 of file Enums.cs.

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

- VectSharp/Enums.cs

6.35 VectSharp.Markdown.Margins Class Reference

Represents the margins of a page.

Public Member Functions

- Margins (double left, double top, double right, double bottom)

  Creates a new Margins instance.

Properties

- double Left [get]

  The left margin.
- double Right [get]

  The right margin.
- double Top [get]

  The top margin.
- double Bottom [get]

  The bottom margin.

6.35.1 Detailed Description

Represents the margins of a page.

Definition at line 185 of file MarkdownContext.cs.

6.35.2 Constructor & Destructor Documentation

6.35.2.1 Margins()

```
VectSharp.Markdown.Margins.Margins (
            double left,
            double top,
            double right,
            double bottom )
```

Creates a new Margins instance.

Parameters

left	The left margin.
top	The top margin.
right	The right margin.
bottom	The bottom margin.

Definition at line 214 of file MarkdownContext.cs.

6.35.3 Property Documentation

6.35.3.1 Bottom

```
double VectSharp.Markdown.Margins.Bottom  [get]
```

The bottom margin.

Definition at line 205 of file MarkdownContext.cs.

6.35.3.2 Left

```
double VectSharp.Markdown.Margins.Left  [get]
```

The left margin.

Definition at line 190 of file MarkdownContext.cs.

6.35.3.3 Right

```
double VectSharp.Markdown.Margins.Right  [get]
```

The right margin.

Definition at line 195 of file MarkdownContext.cs.

6.35.3.4 Top

```
double VectSharp.Markdown.Margins.Top  [get]
```

The top margin.

Definition at line 200 of file MarkdownContext.cs.

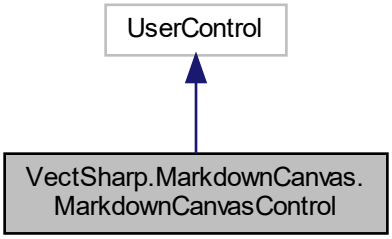
The documentation for this class was generated from the following file:

- VectSharp.Markdown/MarkdownContext.cs

6.36 VectSharp.MarkdownCanvas.MarkdownCanvasControl Class Reference

A control to display a Markdown document in an Avalonia application.

Inheritance diagram for VectSharp.MarkdownCanvas.MarkdownCanvasControl:



Public Member Functions

- MarkdownCanvasControl ()

  Initialises a new MarkdownCanvasControl.

Static Public Attributes

- static readonly StyledProperty< double > MaxRenderWidthProperty = AvaloniaProperty.Register<MarkdownCanvasControl, double>(nameof(MaxRenderWidth), double.PositiveInfinity)

  Defines the MaxRenderWidth property.
- static readonly StyledProperty< double > MinRenderWidthProperty = AvaloniaProperty.Register<MarkdownCanvasControl, double>(nameof(MinRenderWidth), 200)

  Defines the MinRenderWidth property.
- static readonly StyledProperty< double > MinVariationProperty = AvaloniaProperty.Register<MarkdownCanvasControl, double>(nameof(MinVariation), 10)

  Defines the MinVariation property.
- static readonly StyledProperty< string > DocumentSourceProperty = AvaloniaProperty.Register<MarkdownCanvasControl, string>(nameof(DocumentSource))

  Defines the DocumentSource property.
- static readonly StyledProperty< MarkdownDocument > DocumentProperty = AvaloniaProperty.↩ Register<MarkdownCanvasControl, MarkdownDocument>(nameof(Document))

  Defines the Document property.
- static readonly StyledProperty< AvaloniaContextInterpreter.TextOptions > TextConversionOptionsProperty = AvaloniaProperty.Register<MarkdownCanvasControl, AvaloniaContextInterpreter.TextOptions>(nameof(TextConversionOptior AvaloniaContextInterpreter.TextOptions.ConvertIfNecessary)

  Defines the TextConversionOption property.

Properties

- double MaxRenderWidth `[get, set]`

  The maximum width for the rendered document. This will be used even if the control's client area is larger than this (the alignment of the document within the controll will depend on the control's ContentControl.HorizontalContent↩ Alignment).
- double MinRenderWidth `[get, set]`

  The minimum width for the rendered document. If the control's client area is smaller than this, the horizontal scroll bar will be activated.
- double MinVariation `[get, set]`

  The minimum width variation that triggers a document reflow. If the control is resized, but the width changes by less than this amount, the document is not re-drawn.
- string DocumentSource `[set]`

  Sets the currently displayed document from Markdown source.
- MarkdownDocument Document `[get, set]`

  Gets or sets the currently displayed MarkdownDocument.
- AvaloniaContextInterpreter.TextOptions TextConversionOption `[get, set]`

  Gets or sets the value that determines whether text items should be converted into paths when drawing. Setting this to AvaloniaContextInterpreter.TextOptions.NeverConvert will improve performance if you are using custom fonts, but may cause unexpected results unless the font families being used are of type ResourceFontFamily.
- MarkdownRenderer Renderer `[get]`

  The MarkdownRenderer used to render the Document. You can use the properties of this object to customise the rendering. Note that setting the Avalonia.Controls.Primitives.TemplatedControl.FontSize of the MarkdownCanvasControl will propagate to the Renderer's MarkdownRenderer.BaseFontSize.

6.36.1 Detailed Description

A control to display a Markdown document in an Avalonia application.

Definition at line 35 of file MarkdownCanvas.axaml.cs.

6.36.2 Constructor & Destructor Documentation

6.36.2.1 MarkdownCanvasControl()

```
VectSharp.MarkdownCanvas.MarkdownCanvasControl.MarkdownCanvasControl ( )
```

Initialises a new MarkdownCanvasControl.

Definition at line 133 of file MarkdownCanvas.axaml.cs.

6.36.3 Member Data Documentation

6.36.3.1 DocumentProperty

```
readonly StyledProperty<MarkdownDocument> VectSharp.MarkdownCanvas.MarkdownCanvasControl.↩
DocumentProperty = AvaloniaProperty.Register<MarkdownCanvasControl, MarkdownDocument>(nameof(Document))
[static]
```

Defines the Document property.

Definition at line 95 of file MarkdownCanvas.axaml.cs.

6.36.3.2 DocumentSourceProperty

```
readonly StyledProperty<string> VectSharp.MarkdownCanvas.MarkdownCanvasControl.Document↩
SourceProperty = AvaloniaProperty.Register<MarkdownCanvasControl, string>(nameof(DocumentSource))
[static]
```

Defines the DocumentSource property.

Definition at line 82 of file MarkdownCanvas.axaml.cs.

6.36.3.3 MaxRenderWidthProperty

```
readonly StyledProperty<double> VectSharp.MarkdownCanvas.MarkdownCanvasControl.MaxRender↩
WidthProperty = AvaloniaProperty.Register<MarkdownCanvasControl, double>(nameof(MaxRenderWidth),
double.PositiveInfinity)  [static]
```

Defines the MaxRenderWidth property.

Definition at line 40 of file MarkdownCanvas.axaml.cs.

6.36.3.4 MinRenderWidthProperty

```
readonly StyledProperty<double> VectSharp.MarkdownCanvas.MarkdownCanvasControl.MinRender↩
WidthProperty = AvaloniaProperty.Register<MarkdownCanvasControl, double>(nameof(MinRenderWidth),
200)  [static]
```

Defines the MinRenderWidth property.

Definition at line 54 of file MarkdownCanvas.axaml.cs.

6.36.3.5 MinVariationProperty

```
readonly StyledProperty<double> VectSharp.MarkdownCanvas.MarkdownCanvasControl.MinVariation↩
Property = AvaloniaProperty.Register<MarkdownCanvasControl, double>(nameof(MinVariation), 10)
[static]
```

Defines the MinVariation property.

Definition at line 68 of file MarkdownCanvas.axaml.cs.

6.36.3.6 TextConversionOptionsProperty

```
readonly StyledProperty<AvaloniaContextInterpreter.TextOptions> VectSharp.MarkdownCanvas.↩
MarkdownCanvasControl.TextConversionOptionsProperty = AvaloniaProperty.Register<MarkdownCanvasControl,
AvaloniaContextInterpreter.TextOptions>(nameof(TextConversionOption), AvaloniaContextInterpreter.↩
TextOptions.ConvertIfNecessary)  [static]
```

Defines the TextConversionOption property.

Definition at line 109 of file MarkdownCanvas.axaml.cs.

6.36.4 Property Documentation

6.36.4.1 Document

```
MarkdownDocument VectSharp.MarkdownCanvas.MarkdownCanvasControl.Document  [get], [set]
```

Gets or sets the currently displayed MarkdownDocument.

Definition at line 100 of file MarkdownCanvas.axaml.cs.

6.36.4.2 DocumentSource

```
string VectSharp.MarkdownCanvas.MarkdownCanvasControl.DocumentSource  [set]
```

Sets the currently displayed document from Markdown source.

Definition at line 87 of file MarkdownCanvas.axaml.cs.

6.36.4.3 MaxRenderWidth

```
double VectSharp.MarkdownCanvas.MarkdownCanvasControl.MaxRenderWidth  [get], [set]
```

The maximum width for the rendered document. This will be used even if the control's client area is larger than this (the alignment of the document within the controll will depend on the control's ContentControl.HorizontalContent↩ Alignment).

Definition at line 45 of file MarkdownCanvas.axaml.cs.

6.36.4.4 MinRenderWidth

```
double VectSharp.MarkdownCanvas.MarkdownCanvasControl.MinRenderWidth  [get], [set]
```

The minimum width for the rendered document. If the control's client area is smaller than this, the horizontal scroll bar will be activated.

Definition at line 59 of file MarkdownCanvas.axaml.cs.

6.36.4.5 MinVariation

```
double VectSharp.MarkdownCanvas.MarkdownCanvasControl.MinVariation  [get], [set]
```

The minimum width variation that triggers a document reflow. If the control is resized, but the width changes by less than this amount, the document is not re-drawn.

Definition at line 73 of file MarkdownCanvas.axaml.cs.

6.36.4.6 Renderer

```
MarkdownRenderer VectSharp.MarkdownCanvas.MarkdownCanvasControl.Renderer  [get]
```

The MarkdownRenderer used to render the Document. You can use the properties of this object to customise the rendering. Note that setting the Avalonia.Controls.Primitives.TemplatedControl.FontSize of the MarkdownCanvasControl will propagate to the Renderer's MarkdownRenderer.BaseFontSize.

Definition at line 124 of file MarkdownCanvas.axaml.cs.

6.36.4.7 TextConversionOption

```
AvaloniaContextInterpreter.TextOptions VectSharp.MarkdownCanvas.MarkdownCanvasControl.Text↩
ConversionOption [get], [set]
```

Gets or sets the value that determines whether text items should be converted into paths when drawing. Setting this to AvaloniaContextInterpreter.TextOptions.NeverConvert will improve performance if you are using custom fonts, but may cause unexpected results unless the font families being used are of type ResourceFontFamily.

Definition at line 115 of file MarkdownCanvas.axaml.cs.

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

- VectSharp.MarkdownCanvas/MarkdownCanvas.axaml.cs

6.37 VectSharp.Markdown.MarkdownRenderer Class Reference

Renders Markdown documents into VectSharp graphics objects.

Public Types

- enum VerticalAlignment { VerticalAlignment.Top, VerticalAlignment.Middle, VerticalAlignment.Bottom }

  Defines the options for the vertical alignment of table cells.

Public Member Functions

- Page RenderSinglePage (string markdownSource, double width, out Dictionary< string, string > link↩ Destinations)

  Parses the supplied markdownSource using all the supported extensions and renders the resulting document. Page breaks are disabled, and the document is rendered as a single page with the specified width . The page will be cropped at the appropriate height to contain the entire document.
- Page RenderSinglePage (MarkdownDocument markdownDocument, double width, out Dictionary< string, string > linkDestinations)

  Renders the supplied markdownDocument . Page breaks are disabled, and the document is rendered as a single page with the specified width . The page will be cropped at the appropriate height to contain the entire document.
- Document Render (string markdownSource, out Dictionary< string, string > linkDestinations)

  Parses the supplied markdownSource using all the supported extensions and renders the resulting document. The Document produced consists of one or more pages of the size specified in the PageSize of the current instance.
- Document Render (MarkdownDocument mardownDocument, out Dictionary< string, string > link↩ Destinations)

  Renders the supplied mardownDocument . The Document produced consists of one or more pages of the size specified in the PageSize of the current instance.

Properties

- double BaseFontSize = 9.71424 `[get, set]`

  The base font size to use when rendering the document. This will be the size of regular elements, and the size of header elements will be expressed as a multiple of this.
- double[ ] HeaderFontSizeMultipliers `[get]`

  The font size for elements at each header level. The values in this array will be multiplied by the BaseFontSize.
- double[ ] HeaderLineThicknesses = new double[ ] { 1, 1, 0, 0, 0, 0 } `[get]`

  The thickness of the separator line after a header of each level. A value of 0 disables the line after headers of that level.
- double ThematicBreakThickness = 2 `[get, set]`

  The thickness of thematic break lines.
- FontFamily RegularFontFamily = FontFamily.ResolveFontFamily(FontFamily.StandardFontFamilies.Helvetica) `[get, set]`

  The font family for regular text.
- FontFamily BoldFontFamily = FontFamily.ResolveFontFamily(FontFamily.StandardFontFamilies.HelveticaBold) `[get, set]`

  The font family for bold text.
- FontFamily ItalicFontFamily = FontFamily.ResolveFontFamily(FontFamily.StandardFontFamilies.HelveticaOblique) `[get, set]`

  The font family for italic text.
- FontFamily BoldItalicFontFamily = FontFamily.ResolveFontFamily(FontFamily.StandardFontFamilies.HelveticaBoldOblique) `[get, set]`

  The font family for bold italic text.
- FontFamily CodeFont = FontFamily.ResolveFontFamily(FontFamily.StandardFontFamilies.Courier) `[get, set]`

  The font family for code elements.
- FontFamily CodeFontBold = FontFamily.ResolveFontFamily(FontFamily.StandardFontFamilies.CourierBold) `[get, set]`

  The font family for bold code elements.
- FontFamily CodeFontItalic = FontFamily.ResolveFontFamily(FontFamily.StandardFontFamilies.CourierOblique) `[get, set]`

  The font family for italic code elements.
- FontFamily CodeFontBoldItalic = FontFamily.ResolveFontFamily(FontFamily.StandardFontFamilies.CourierBoldOblique) `[get, set]`

  The font family for bold italic code elements.
- double UnderlineThickness = 0.075 `[get, set]`

  The thickness of underlines. This value will be multiplied by the font size of the element being underlined.
- double BoldUnderlineThickness = 0.15 `[get, set]`

  The thickness of underlines for bold text. This value will be multiplied by the font size of the element being underlined.
- Margins Margins = new Margins(55, 55, 55, 55) `[get, set]`

  The margins of the page.
- Margins TableCellMargins = new Margins(5, 0, 5, 0) `[get, set]`

  The margins for table cells.
- VerticalAlignment TableVAlign = VerticalAlignment.Middle `[get, set]`

  The vertical alignment of table cells.
- Size PageSize = new Size(595, 842) `[get, set]`

  The size of the page.
- double SpaceBeforeParagaph = 0 `[get, set]`

  The space before each text paragraph. This value will be multiplied by the BaseFontSize.
- double SpaceAfterParagraph = 0.75 `[get, set]`

  The space after each text paragraph. This value will be multiplied by the BaseFontSize.

- double SpaceAfterLine = 0.25 [get, set]

  The space after each line of text. This value will be multiplied by the BaseFontSize.
- double SpaceBeforeHeading = 0.25 [get, set]

  The space before each heading. This value will be multiplied by the font size of the heading.
- double SpaceAfterHeading = 0.25 [get, set]

  The space after each heading. This value will be multiplied by the font size of the heading.
- double CodeInlineMargin = 0.25 [get, set]

  The margin at the left and right of code inlines. This value will be multiplied by the current font size.
- double IndentWidth = 40 [get, set]

  The indentation width used for list items.
- double QuoteBlockIndentWidth = 30 [get, set]

  The indentation width used for block quotes.
- double QuoteBlockBarWidth = 5 [get, set]

  The thickness of the bar to the left of block quotes.
- double SubSuperscriptFontSize = 0.7 [get, set]

  The font size for subscripts and superscripts. This value will be multiplied by the current font size.
- double SuperscriptShift = 0.33 [get, set]

  The upwards shift in the baseline for superscript elements. This value will be multiplied by the current font size.
- double SubscriptShift = 0.14 [get, set]

  The downwards shift in the baseline for subscript elements. This value will be multiplied by the current font size.
- string BaseImageUri = "" [get, set]

  The base uri for resolving relative image addresses.
- Func< string, string,(string, bool)> ImageUriResolver = HTTPUtils.ResolveImageURI [get, set]

  A method used to resolve (possibly remote) image uris into local file paths. The first argument of the method should be the image uri and the second argument the base uri used to resolve relative links. The method should return a tuple containing the path of the local file and a boolean value indicating whether the file has been fetched from a remote location and should be deleted after the program has finished using it.
- Uri BaseLinkUri = new Uri("about:blank") [get, set]

  The base uri for resolving links.
- Func< string, string > LinkUriResolver = a => a [get, set]

  A method used to resolve link addresses. The argument of the method should be the absolute link, and the method should return the resolved address. This can be used to "redirect" links to a different target.
- Func< string, RasterImage > RasterImageLoader = null [get, set]

  A method used to a load raster image from a local file. The argument of the method should be the path of a local image file, and the method should return a RasterImage representing that file. For example, this can be achieved using the `RasterImageFile` class from the `VectSharp.MuPDFUtils` package. If this is `null`, only SVG images will be included in the document.
- double ImageUnitMultiplier = 0.60714 [get, set]

  The size of images (as defined in the image's width and height attributes) will be multiplied by this value to determine the actual size of the image on the page. This has no effect on images without a width or height attribute.
- double ImageMultiplier = 1 [get, set]

  The size of images will be multiplied by this value to determine the actual size of the image on the page. For images that have a width or height attribute, this will be applied in addition to the ImageUnitMultiplier. For images without width and height, only this multiplier will be applied.
- double ImageSideMargin = 10 [get, set]

  The margin on the right of left-aligned images and on the left of right-aligned images.
- double ImageMarginTolerance = 25 [get, set]

  Images will be allowed to extend into the page bottom margin area by this amount before triggering a page break. This should be smaller than the bottom margin, otherwise images risk being cut off by the page boundary.
- Func< string, string, List< List< FormattedString > > > SyntaxHighlighter = VectSharp.Markdown.SyntaxHighlighter.GetSynta [get, set]

A method used for syntax highlighting. The first argument should be the source code to highlight, while the second parameter is the name of the language to use for the highlight. The method should return a list of lists of FormattedStrings, with each list of FormattedStrings representing a line. For each code block, if the method returns `null`, no syntax highlighting is used.

- List< Action< Graphics, Colour > > Bullets `[get]`

  Bullet points used for unordered lists. Each element of this list corresponds to the bullet for each level of list indentation. If the list indentation is greater than the number of elements in this list, the bullet points will be reused cyclically. Each element of this list is a method taking two arguments: the first is the Graphics object on which the bullet point should be drawn, while the second is the colour in which it should be painted. The method should draw the bullet point centered around the origin. The size of the bullet point will be multiplied by the current font size.
- Colour ForegroundColour = Colours.Black `[get, set]`

  The foreground colour for text elements.
- Colour BackgroundColour = Colours.White `[get, set]`

  The background colour for the page.
- Colour HeaderLineColour = Colour.FromRgb(180, 180, 180) `[get, set]`

  The colour of the line below headers.
- Colour ThematicBreakLineColour = Colour.FromRgb(180, 180, 200) `[get, set]`

  The colour for thematic break lines.
- Colour LinkColour = Colour.FromRgb(25, 140, 191) `[get, set]`

  The colour for hypertext links-
- Colour CodeInlineBackgroundColour = Colour.FromRgb(240, 240, 240) `[get, set]`

  The background colour for code inlines.
- Colour CodeBlockBackgroundColour = Colour.FromRgb(240, 240, 245) `[get, set]`

  The background colour for code blocks.
- Colour QuoteBlockBarColour = Colour.FromRgb(75, 152, 220) `[get, set]`

  The colour for the bar to the left of block quotes.
- Colour QuoteBlockBackgroundColour = Colour.FromRgb(240, 240, 255) `[get, set]`

  The background colour for block quotes.
- Colour InsertedColour = Colour.FromRgb(0, 158, 115) `[get, set]`

  The colour for text that has been styled as "inserted".
- Colour MarkedColour = Colour.FromRgb(213, 94, 0) `[get, set]`

  The colour for text that has been styled as "marked".
- Colour TableHeaderRowSeparatorColour = Colours.Black `[get, set]`

  The colour for the line separating the table header row from normal rows.
- Colour TableRowSeparatorColour = Colour.FromRgb(180, 180, 180) `[get, set]`

  The colour for lines separating table rows from each other.
- double TableHeaderRowSeparatorThickness = 2 `[get, set]`

  The thickness of the line separating the table header row from normal rows.
- double TableHeaderSeparatorThickness = 1 `[get, set]`

  The thickness of lines separating table rows from each other.
- Graphics TaskListUncheckedBullet `[get, set]`

  The bullet used for unchecked task list items.
- Graphics TaskListCheckedBullet `[get, set]`

  The bullet used for checked task list items.
- bool AllowPageBreak = true `[get, set]`

  Determines whether page breaks should be treated as such in the source.

6.37.1 Detailed Description

Renders Markdown documents into VectSharp graphics objects.

Definition at line 35 of file MarkdownRenderer.cs.

6.37.2 Member Enumeration Documentation

6.37.2.1 VerticalAlignment

```
enum VectSharp.Markdown.MarkdownRenderer.VerticalAlignment  [strong]
```

Defines the options for the vertical alignment of table cells.

Enumerator

Top	Table cells will be aligned at the top of their row.
Middle	Table cells will be aligned in the middle of their row.
Bottom	Table cells will be aligned at the bottom of their row.

Definition at line 123 of file MarkdownRenderer.cs.

6.37.3 Member Function Documentation

6.37.3.1 Render() [1/2]

```
Document VectSharp.Markdown.MarkdownRenderer.Render (
            MarkdownDocument mardownDocument,
            out Dictionary< string, string > linkDestinations )
```

Renders the supplied *mardownDocument* . The Document produced consists of one or more pages of the size specified in the PageSize of the current instance.

Parameters

mardownDocument	The markdown document to render.
linkDestinations	When this method returns, this value will contain a dictionary used to associate graphic action tags to hyperlinks. This can be used to enable such links when rendering the Document to a file.

Returns

A Document containing a rendering of the supplied markdown document, consisting of one or more pages of the size specified in the PageSize of the current instance.

Definition at line 495 of file MarkdownRenderer.cs.

6.37.3.2 Render() [2/2]

```
Document VectSharp.Markdown.MarkdownRenderer.Render (
            string markdownSource,
            out Dictionary< string, string > linkDestinations )
```

Parses the supplied *markdownSource* using all the supported extensions and renders the resulting document. The Document produced consists of one or more pages of the size specified in the PageSize of the current instance.

Parameters

markdownSource	The markdown source to parse.
linkDestinations	When this method returns, this value will contain a dictionary used to associate graphic action tags to hyperlinks. This can be used to enable such links when rendering the Document to a file.

Returns

A Document containing a rendering of the supplied markdown document, consisting of one or more pages of the size specified in the PageSize of the current instance.

Definition at line 482 of file MarkdownRenderer.cs.

6.37.3.3 RenderSinglePage() [1/2]

```
Page VectSharp.Markdown.MarkdownRenderer.RenderSinglePage (
            MarkdownDocument markdownDocument,
            double width,
            out Dictionary< string, string > linkDestinations )
```

Renders the supplied *markdownDocument* . Page breaks are disabled, and the document is rendered as a single page with the specified *width* . The page will be cropped at the appropriate height to contain the entire document.

Parameters

markdownDocument	The markdown document to render.
width	The width of the page.
linkDestinations	When this method returns, this value will contain a dictionary used to associate graphic action tags to hyperlinks. This can be used to enable such links when rendering the Page to a file.

Returns

A Page containing a rendering of the supplied markdown document.

Definition at line 423 of file MarkdownRenderer.cs.

6.37.3.4 RenderSinglePage() [2/2]

```
Page VectSharp.Markdown.MarkdownRenderer.RenderSinglePage (
            string markdownSource,
            double width,
            out Dictionary< string, string > linkDestinations )
```

Parses the supplied *markdownSource* using all the supported extensions and renders the resulting document. Page breaks are disabled, and the document is rendered as a single page with the specified *width* . The page will be cropped at the appropriate height to contain the entire document.

Parameters

markdownSource	The markdown source to parse.
width	The width of the page.
linkDestinations	When this method returns, this value will contain a dictionary used to associate graphic action tags to hyperlinks. This can be used to enable such links when rendering the Page to a file.

Returns

A Page containing a rendering of the supplied markdown document.

Definition at line 409 of file MarkdownRenderer.cs.

6.37.4 Property Documentation

6.37.4.1 AllowPageBreak

```
bool VectSharp.Markdown.MarkdownRenderer.AllowPageBreak = true  [get], [set]
```

Determines whether page breaks should be treated as such in the source.

Definition at line 395 of file MarkdownRenderer.cs.

6.37.4.2 BackgroundColour

```
Colour VectSharp.Markdown.MarkdownRenderer.BackgroundColour = Colours.White  [get], [set]
```

The background colour for the page.

Definition at line 291 of file MarkdownRenderer.cs.

6.37.4.3 BaseFontSize

```
double VectSharp.Markdown.MarkdownRenderer.BaseFontSize = 9.71424  [get], [set]
```

The base font size to use when rendering the document. This will be the size of regular elements, and the size of header elements will be expressed as a multiple of this.

Definition at line 40 of file MarkdownRenderer.cs.

6.37.4.4 BaseImageUri

```
string VectSharp.Markdown.MarkdownRenderer.BaseImageUri = ""  [get], [set]
```

The base uri for resolving relative image addresses.

Definition at line 214 of file MarkdownRenderer.cs.

6.37.4.5 BaseLinkUri

```
Uri VectSharp.Markdown.MarkdownRenderer.BaseLinkUri = new Uri("about:blank")  [get], [set]
```

The base uri for resolving links.

Definition at line 224 of file MarkdownRenderer.cs.

6.37.4.6 BoldFontFamily

```
FontFamily VectSharp.Markdown.MarkdownRenderer.BoldFontFamily = FontFamily.ResolveFontFamily(FontFamily.Standa
[get], [set]
```

The font family for bold text.

Definition at line 68 of file MarkdownRenderer.cs.

6.37.4.7 BoldItalicFontFamily

```
FontFamily VectSharp.Markdown.MarkdownRenderer.BoldItalicFontFamily = FontFamily.ResolveFontFamily(FontFamily.
[get], [set]
```

The font family for bold italic text.

Definition at line 78 of file MarkdownRenderer.cs.

6.37.4.8 BoldUnderlineThickness

```
double VectSharp.Markdown.MarkdownRenderer.BoldUnderlineThickness = 0.15  [get], [set]
```

The thickness of underlines for bold text. This value will be multiplied by the font size of the element being underlined.

Definition at line 108 of file MarkdownRenderer.cs.

6.37.4.9 Bullets

```
List<Action<Graphics, Colour> > VectSharp.Markdown.MarkdownRenderer.Bullets  [get]
```

Initial value:

```
= new List<Action<Graphics, Colour»()
        {
            (graphics, colour) =>
            {
                graphics.FillPath(new GraphicsPath().Arc(-0.5, 0, 0.25, 0, 2 * Math.PI), colour);
            },
            (graphics, colour) =>
            {
                graphics.StrokePath(new GraphicsPath().Arc(-0.5, 0, 0.25, 0, 2 * Math.PI), colour, 0.1);
            },
            (graphics, colour) =>
            {
                graphics.StrokeRectangle(-0.75, -0.25, 0.5, 0.5, colour, 0.1);
            },
        }
```

Bullet points used for unordered lists. Each element of this list corresponds to the bullet for each level of list indentation. If the list indentation is greater than the number of elements in this list, the bullet points will be reused cyclically. Each element of this list is a method taking two arguments: the first is the Graphics object on which the bullet point should be drawn, while the second is the colour in which it should be painted. The method should draw the bullet point centered around the origin. The size of the bullet point will be multiplied by the current font size.

Definition at line 265 of file MarkdownRenderer.cs.

6.37.4.10 CodeBlockBackgroundColour

```
Colour VectSharp.Markdown.MarkdownRenderer.CodeBlockBackgroundColour = Colour.FromRgb(240,
240, 245)  [get], [set]
```

The background colour for code blocks.

Definition at line 316 of file MarkdownRenderer.cs.

6.37.4.11 CodeFont

```
FontFamily VectSharp.Markdown.MarkdownRenderer.CodeFont = FontFamily.ResolveFontFamily(FontFamily.StandardFont
[get], [set]
```

The font family for code elements.

Definition at line 83 of file MarkdownRenderer.cs.

6.37.4.12 CodeFontBold

```
FontFamily VectSharp.Markdown.MarkdownRenderer.CodeFontBold = FontFamily.ResolveFontFamily(FontFamily.Standard
[get], [set]
```

The font family for bold code elements.

Definition at line 88 of file MarkdownRenderer.cs.

6.37.4.13 CodeFontBoldItalic

```
FontFamily VectSharp.Markdown.MarkdownRenderer.CodeFontBoldItalic = FontFamily.ResolveFontFamily(FontFamily.St
[get], [set]
```

The font family for bold italic code elements.

Definition at line 98 of file MarkdownRenderer.cs.

6.37.4.14 CodeFontItalic

```
FontFamily VectSharp.Markdown.MarkdownRenderer.CodeFontItalic = FontFamily.ResolveFontFamily(FontFamily.Standa
[get], [set]
```

The font family for italic code elements.

Definition at line 93 of file MarkdownRenderer.cs.

6.37.4.15 CodeInlineBackgroundColour

```
Colour VectSharp.Markdown.MarkdownRenderer.CodeInlineBackgroundColour = Colour.FromRgb(240,
240, 240)  [get], [set]
```

The background colour for code inlines.

Definition at line 311 of file MarkdownRenderer.cs.

6.37.4.16 CodeInlineMargin

```
double VectSharp.Markdown.MarkdownRenderer.CodeInlineMargin = 0.25  [get], [set]
```

The margin at the left and right of code inlines. This value will be multiplied by the current font size.

Definition at line 179 of file MarkdownRenderer.cs.

6.37.4.17 ForegroundColour

```
Colour VectSharp.Markdown.MarkdownRenderer.ForegroundColour = Colours.Black  [get], [set]
```

The foreground colour for text elements.

Definition at line 286 of file MarkdownRenderer.cs.

6.37.4.18 HeaderFontSizeMultipliers

```
double [] VectSharp.Markdown.MarkdownRenderer.HeaderFontSizeMultipliers  [get]
```

Initial value:

```
= new double[]
        {
            28 / 12.0, 22 / 12.0, 16 / 12.0, 14 / 12.0, 13 / 12.0, 12 / 12.0
        }
```

The font size for elements at each header level. The values in this array will be multiplied by the BaseFontSize.

Definition at line 45 of file MarkdownRenderer.cs.

6.37.4.19 HeaderLineColour

```
Colour VectSharp.Markdown.MarkdownRenderer.HeaderLineColour = Colour.FromRgb(180, 180, 180)
[get], [set]
```

The colour of the line below headers.

Definition at line 296 of file MarkdownRenderer.cs.

6.37.4.20 HeaderLineThicknesses

```
double [] VectSharp.Markdown.MarkdownRenderer.HeaderLineThicknesses = new double[] { 1, 1, 0,
0, 0, 0 }  [get]
```

The thickness of the separator line after a header of each level. A value of 0 disables the line after headers of that level.

Definition at line 53 of file MarkdownRenderer.cs.

6.37.4.21 ImageMarginTolerance

```
double VectSharp.Markdown.MarkdownRenderer.ImageMarginTolerance = 25  [get], [set]
```

Images will be allowed to extend into the page bottom margin area by this amount before triggering a page break. This should be smaller than the bottom margin, otherwise images risk being cut off by the page boundary.

Definition at line 254 of file MarkdownRenderer.cs.

6.37.4.22 ImageMultiplier

```
double VectSharp.Markdown.MarkdownRenderer.ImageMultiplier = 1  [get], [set]
```

The size of images will be multiplied by this value to determine the actual size of the image on the page. For images that have a width or height attribute, this will be applied in addition to the ImageUnitMultiplier. For images without width and height, only this multiplier will be applied.

Definition at line 244 of file MarkdownRenderer.cs.

6.37.4.23 ImageSideMargin

```
double VectSharp.Markdown.MarkdownRenderer.ImageSideMargin = 10  [get], [set]
```

The margin on the right of left-aligned images and on the left of right-aligned images.

Definition at line 249 of file MarkdownRenderer.cs.

6.37.4.24 ImageUnitMultiplier

```
double VectSharp.Markdown.MarkdownRenderer.ImageUnitMultiplier = 0.60714  [get], [set]
```

The size of images (as defined in the image's width and height attributes) will be multiplied by this value to determine the actual size of the image on the page. This has no effect on images without a width or height attribute.

Definition at line 239 of file MarkdownRenderer.cs.

6.37.4.25 ImageUriResolver

```
Func<string, string, (string, bool)> VectSharp.Markdown.MarkdownRenderer.ImageUriResolver =
HTTPUtils.ResolveImageURI  [get], [set]
```

A method used to resolve (possibly remote) image uris into local file paths. The first argument of the method should be the image uri and the second argument the base uri used to resolve relative links. The method should return a tuple containing the path of the local file and a boolean value indicating whether the file has been fetched from a remote location and should be deleted after the program has finished using it.

Definition at line 219 of file MarkdownRenderer.cs.

6.37.4.26 IndentWidth

```
double VectSharp.Markdown.MarkdownRenderer.IndentWidth = 40  [get], [set]
```

The indentation width used for list items.

Definition at line 184 of file MarkdownRenderer.cs.

6.37.4.27 InsertedColour

```
Colour VectSharp.Markdown.MarkdownRenderer.InsertedColour = Colour.FromRgb(0, 158, 115)  [get],
[set]
```

The colour for text that has been styled as "inserted".

Definition at line 331 of file MarkdownRenderer.cs.

6.37.4.28 ItalicFontFamily

```
FontFamily VectSharp.Markdown.MarkdownRenderer.ItalicFontFamily = FontFamily.ResolveFontFamily(FontFamily.Star
[get], [set]
```

The font family for italic text.

Definition at line 73 of file MarkdownRenderer.cs.

6.37.4.29 LinkColour

```
Colour VectSharp.Markdown.MarkdownRenderer.LinkColour = Colour.FromRgb(25, 140, 191)  [get],
[set]
```

The colour for hypertext links-

Definition at line 306 of file MarkdownRenderer.cs.

6.37.4.30 LinkUriResolver

```
Func<string, string> VectSharp.Markdown.MarkdownRenderer.LinkUriResolver = a => a  [get],
[set]
```

A method used to resolve link addresses. The argument of the method should be the absolute link, and the method should return the resolved address. This can be used to "redirect" links to a different target.

Definition at line 229 of file MarkdownRenderer.cs.

6.37.4.31 Margins

```
Margins VectSharp.Markdown.MarkdownRenderer.Margins = new Margins(55, 55, 55, 55)  [get], [set]
```

The margins of the page.

Definition at line 113 of file MarkdownRenderer.cs.

6.37.4.32 MarkedColour

```
Colour VectSharp.Markdown.MarkdownRenderer.MarkedColour = Colour.FromRgb(213, 94, 0)  [get],
[set]
```

The colour for text that has been styled as "marked".

Definition at line 336 of file MarkdownRenderer.cs.

6.37.4.33 PageSize

```
Size VectSharp.Markdown.MarkdownRenderer.PageSize = new Size(595, 842)  [get], [set]
```

The size of the page.

Definition at line 149 of file MarkdownRenderer.cs.

6.37.4.34 QuoteBlockBackgroundColour

```
Colour VectSharp.Markdown.MarkdownRenderer.QuoteBlockBackgroundColour = Colour.FromRgb(240,
240, 255)  [get], [set]
```

The background colour for block quotes.

Definition at line 326 of file MarkdownRenderer.cs.

6.37.4.35 QuoteBlockBarColour

```
Colour VectSharp.Markdown.MarkdownRenderer.QuoteBlockBarColour = Colour.FromRgb(75, 152, 220)
[get], [set]
```

The colour for the bar to the left of block quotes.

Definition at line 321 of file MarkdownRenderer.cs.

6.37.4.36 QuoteBlockBarWidth

```
double VectSharp.Markdown.MarkdownRenderer.QuoteBlockBarWidth = 5  [get], [set]
```

The thickness of the bar to the left of block quotes.

Definition at line 194 of file MarkdownRenderer.cs.

6.37.4.37 QuoteBlockIndentWidth

```
double VectSharp.Markdown.MarkdownRenderer.QuoteBlockIndentWidth = 30  [get], [set]
```

The indentation width used for block quotes.

Definition at line 189 of file MarkdownRenderer.cs.

6.37.4.38 RasterImageLoader

```
Func<string, RasterImage> VectSharp.Markdown.MarkdownRenderer.RasterImageLoader = null  [get],
[set]
```

A method used to a load raster image from a local file. The argument of the method should be the path of a local image file, and the method should return a RasterImage representing that file. For example, this can be achieved using the `RasterImageFile` class from the `VectSharp.MuPDFUtils` package. If this is `null`, only SVG images will be included in the document.

Definition at line 234 of file MarkdownRenderer.cs.

6.37.4.39 RegularFontFamily

```
FontFamily VectSharp.Markdown.MarkdownRenderer.RegularFontFamily = FontFamily.ResolveFontFamily(FontFamily.Sta
[get], [set]
```

The font family for regular text.

Definition at line 63 of file MarkdownRenderer.cs.

6.37.4.40 SpaceAfterHeading

```
double VectSharp.Markdown.MarkdownRenderer.SpaceAfterHeading = 0.25  [get], [set]
```

The space after each heading. This value will be multiplied by the font size of the heading.

Definition at line 174 of file MarkdownRenderer.cs.

6.37.4.41 SpaceAfterLine

```
double VectSharp.Markdown.MarkdownRenderer.SpaceAfterLine = 0.25  [get], [set]
```

The space after each line of text. This value will be multiplied by the BaseFontSize.

Definition at line 164 of file MarkdownRenderer.cs.

6.37.4.42 SpaceAfterParagraph

```
double VectSharp.Markdown.MarkdownRenderer.SpaceAfterParagraph = 0.75  [get], [set]
```

The space after each text paragraph. This value will be multiplied by the BaseFontSize.

Definition at line 159 of file MarkdownRenderer.cs.

6.37.4.43 SpaceBeforeHeading

```
double VectSharp.Markdown.MarkdownRenderer.SpaceBeforeHeading = 0.25  [get], [set]
```

The space before each heading. This value will be multiplied by the font size of the heading.

Definition at line 169 of file MarkdownRenderer.cs.

6.37.4.44 SpaceBeforeParagaph

```
double VectSharp.Markdown.MarkdownRenderer.SpaceBeforeParagaph = 0  [get], [set]
```

The space before each text paragraph. This value will be multiplied by the BaseFontSize.

Definition at line 154 of file MarkdownRenderer.cs.

6.37.4.45 SubscriptShift

```
double VectSharp.Markdown.MarkdownRenderer.SubscriptShift = 0.14  [get], [set]
```

The downwards shift in the baseline for subscript elements. This value will be multiplied by the current font size.

Definition at line 209 of file MarkdownRenderer.cs.

6.37.4.46 SubSuperscriptFontSize

```
double VectSharp.Markdown.MarkdownRenderer.SubSuperscriptFontSize = 0.7  [get], [set]
```

The font size for subscripts and superscripts. This value will be multiplied by the current font size.

Definition at line 199 of file MarkdownRenderer.cs.

6.37.4.47 SuperscriptShift

```
double VectSharp.Markdown.MarkdownRenderer.SuperscriptShift = 0.33  [get], [set]
```

The upwards shift in the baseline for superscript elements. This value will be multiplied by the current font size.

Definition at line 204 of file MarkdownRenderer.cs.

6.37.4.48 SyntaxHighlighter

```
Func<string, string, List<List<FormattedString> > > VectSharp.Markdown.MarkdownRenderer.↩
SyntaxHighlighter = VectSharp.Markdown.SyntaxHighlighter.GetSyntaxHighlightedLines  [get],
[set]
```

A method used for syntax highlighting. The first argument should be the source code to highlight, while the second parameter is the name of the language to use for the highlight. The method should return a list of lists of FormattedStrings, with each list of FormattedStrings representing a line. For each code block, if the method returns `null`, no syntax highlighting is used.

Definition at line 259 of file MarkdownRenderer.cs.

6.37.4.49 TableCellMargins

```
Margins VectSharp.Markdown.MarkdownRenderer.TableCellMargins = new Margins(5, 0, 5, 0)  [get],
[set]
```

The margins for table cells.

Definition at line 118 of file MarkdownRenderer.cs.

6.37.4.50 TableHeaderRowSeparatorColour

```
Colour VectSharp.Markdown.MarkdownRenderer.TableHeaderRowSeparatorColour = Colours.Black  [get], [set]
```

The colour for the line separating the table header row from normal rows.

Definition at line 341 of file MarkdownRenderer.cs.

6.37.4.51 TableHeaderRowSeparatorThickness

```
double VectSharp.Markdown.MarkdownRenderer.TableHeaderRowSeparatorThickness = 2  [get], [set]
```

The thickness of the line separating the table header row from normal rows.

Definition at line 351 of file MarkdownRenderer.cs.

6.37.4.52 TableHeaderSeparatorThickness

```
double VectSharp.Markdown.MarkdownRenderer.TableHeaderSeparatorThickness = 1  [get], [set]
```

The thickness of lines separating table rows from each other.

Definition at line 356 of file MarkdownRenderer.cs.

6.37.4.53 TableRowSeparatorColour

```
Colour VectSharp.Markdown.MarkdownRenderer.TableRowSeparatorColour = Colour.FromRgb(180, 180, 180)  [get], [set]
```

The colour for lines separating table rows from each other.

Definition at line 346 of file MarkdownRenderer.cs.

6.37.4.54 TableVAlign

```
VerticalAlignment VectSharp.Markdown.MarkdownRenderer.TableVAlign = VerticalAlignment.Middle  [get], [set]
```

The vertical alignment of table cells.

Definition at line 144 of file MarkdownRenderer.cs.

6.37.4.55 TaskListCheckedBullet

```
Graphics VectSharp.Markdown.MarkdownRenderer.TaskListCheckedBullet  [get], [set]
```

Initial value:

```
= new Func<Graphics>(() =>
        {
            Graphics tbr = new Graphics();
            GraphicsPath checkboxPath = new GraphicsPath().MoveTo(-0.7, -0.4).LineTo(-0.3, -0.4).Arc(-0.3,
       -0.2, 0.2, 3 * Math.PI / 2, 2 * Math.PI).LineTo(-0.1, 0.2).Arc(-0.3, 0.2, 0.2, 0, Math.PI /
       2).LineTo(-0.7, 0.4).Arc(-0.7, 0.2, 0.2, Math.PI / 2, Math.PI).LineTo(-0.9, -0.2).Arc(-0.7, -0.2,
       0.2, Math.PI, 3 * Math.PI / 2).Close();
            tbr.FillPath(checkboxPath, Colour.FromRgb(240, 246, 249));
            tbr.StrokePath(checkboxPath, Colour.FromRgb(0, 114, 178), 0.075);
            GraphicsPath tickpath = new GraphicsPath().MoveTo(-0.75, -0.1).LineTo(-0.5, 0.15).LineTo(-0.1,
       -0.4);
            tbr.StrokePath(new GraphicsPath().MoveTo(-0.5, 0.15).LineTo(-0.1, -0.4), Colour.FromRgb(240,
       246, 249), 0.3, LineCaps.Round);
            tbr.StrokePath(tickpath, Colour.FromRgb(0, 158, 115), 0.2, LineCaps.Round);
            return tbr;
        })()
```

The bullet used for checked task list items.

Definition at line 376 of file MarkdownRenderer.cs.

6.37.4.56 TaskListUncheckedBullet

```
Graphics VectSharp.Markdown.MarkdownRenderer.TaskListUncheckedBullet  [get], [set]
```

Initial value:

```
= new Func<Graphics>(() =>
        {
            Graphics tbr = new Graphics();
            GraphicsPath checkboxPath = new GraphicsPath().MoveTo(-0.7, -0.4).LineTo(-0.3, -0.4).Arc(-0.3,
       -0.2, 0.2, 3 * Math.PI / 2, 2 * Math.PI).LineTo(-0.1, 0.2).Arc(-0.3, 0.2, 0.2, 0, Math.PI /
       2).LineTo(-0.7, 0.4).Arc(-0.7, 0.2, 0.2, Math.PI / 2, Math.PI).LineTo(-0.9, -0.2).Arc(-0.7, -0.2,
       0.2, Math.PI, 3 * Math.PI / 2).Close();
            tbr.FillPath(checkboxPath, Colour.FromRgb(240, 246, 249));
            tbr.StrokePath(checkboxPath, Colour.FromRgb(0, 114, 178), 0.075);
            return tbr;
        })()
```

The bullet used for unchecked task list items.

Definition at line 361 of file MarkdownRenderer.cs.

6.37.4.57 ThematicBreakLineColour

```
Colour VectSharp.Markdown.MarkdownRenderer.ThematicBreakLineColour = Colour.FromRgb(180, 180,
200)  [get], [set]
```

The colour for thematic break lines.

Definition at line 301 of file MarkdownRenderer.cs.

6.37.4.58 ThematicBreakThickness

```
double VectSharp.Markdown.MarkdownRenderer.ThematicBreakThickness = 2  [get], [set]
```

The thickness of thematic break lines.

Definition at line 58 of file MarkdownRenderer.cs.

6.37.4.59 UnderlineThickness

```
double VectSharp.Markdown.MarkdownRenderer.UnderlineThickness = 0.075  [get], [set]
```

The thickness of underlines. This value will be multiplied by the font size of the element being underlined.

Definition at line 103 of file MarkdownRenderer.cs.

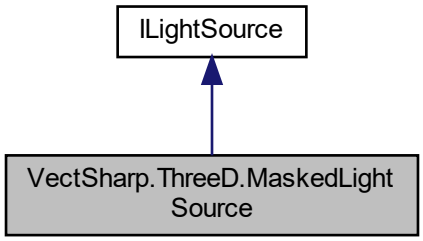
The documentation for this class was generated from the following file:

- VectSharp.Markdown/MarkdownRenderer.cs

6.38 VectSharp.ThreeD.MaskedLightSource Class Reference

Represents a point light source with a stencil in front of it.

Inheritance diagram for VectSharp.ThreeD.MaskedLightSource:



Public Member Functions

- MaskedLightSource (double intensity, Point3D position, NormalizedVector3D direction, double distance, GraphicsPath mask, double maskOrientation, double triangulationResolution)

  Creates a new MaskedLightSource by triangulating the specified GraphicsPath.
- MaskedLightSource (double intensity, Point3D position, NormalizedVector3D direction, double distance, IEnumerable< GraphicsPath > triangulatedMask, double maskOrientation)

  Creates a new MaskedLightSource using the specified triangulatedMask .
- LightIntensity GetLightAt (Point3D point)

  Computes the light intensity at the specified point, without taking into account any obstructions.
- double GetObstruction (Point3D point, IEnumerable< Triangle3DElement > shadowingTriangles)

  Determines the amount of obstruction of the light that results at point due to the specified shadowingTriangles .

Properties

- bool CastsShadow = true [get, set]
- Point3D Position [get]

  The position of the light source.
- Point3D Origin [get]

  The projection of the Position on the mask plane along the light's Direction.
- NormalizedVector3D Direction [get]

  The direction of the light.
- double Distance [get]

  The distance between the light source and the mask plane.
- double Intensity [get, set]

  The base intensity of the light.
- double DistanceAttenuationExponent = 2 [get, set]

  An exponent determining how fast the light attenuates with increasing distance. Set to 0 to disable distance attenuation.
- double AngleAttenuationExponent = 1 [get, set]

  An exponent determining how fast the light attenuates away from the light's axis. Set to 0 to disable angular attenuation.

6.38.1 Detailed Description

Represents a point light source with a stencil in front of it.

Definition at line 385 of file Lights.cs.

6.38.2 Constructor & Destructor Documentation

6.38.2.1 MaskedLightSource() [1/2]

```
VectSharp.ThreeD.MaskedLightSource.MaskedLightSource (
            double intensity,
            Point3D position,
            NormalizedVector3D direction,
            double distance,
            GraphicsPath mask,
            double maskOrientation,
            double triangulationResolution )
```

Creates a new MaskedLightSource by triangulating the specified GraphicsPath.

Parameters

intensity	The base intensity of the light.
position	The position of the light source.
direction	The direction of the light.
distance	The distance between the light source and the mask plane.
mask	A GraphicsPath representing the transparent part of the mask.
maskOrientation	An angle in radians determining the orientation of the 2D mask in the mask plane.
triangulationResolution	The resolution to use to triangulate the *mask* .

Definition at line 437 of file Lights.cs.

6.38.2.2 MaskedLightSource() [2/2]

```
VectSharp.ThreeD.MaskedLightSource.MaskedLightSource (
            double intensity,
            Point3D position,
            NormalizedVector3D direction,
            double distance,
            IEnumerable< GraphicsPath > triangulatedMask,
            double maskOrientation )
```

Creates a new MaskedLightSource using the specified *triangulatedMask* .

Parameters

intensity	The base intensity of the light.
position	The position of the light source.
direction	The direction of the light.
distance	The distance between the light source and the mask plane.
triangulatedMask	A collection of GraphicsPaths representing the transparent part of the mask. Each GraphicsPath should represent a single triangle.
maskOrientation	An angle in radians determining the orientation of the 2D mask in the mask plane.

Definition at line 451 of file Lights.cs.

6.38.3 Property Documentation

6.38.3.1 AngleAttenuationExponent

```
double VectSharp.ThreeD.MaskedLightSource.AngleAttenuationExponent = 1  [get], [set]
```

An exponent determining how fast the light attenuates away from the light's axis. Set to 0 to disable angular attenuation.

Definition at line 425 of file Lights.cs.

6.38.3.2 Direction

```
NormalizedVector3D VectSharp.ThreeD.MaskedLightSource.Direction  [get]
```

The direction of the light.

Definition at line 403 of file Lights.cs.

6.38.3.3 Distance

```
double VectSharp.ThreeD.MaskedLightSource.Distance  [get]
```

The distance between the light source and the mask plane.

Definition at line 408 of file Lights.cs.

6.38.3.4 DistanceAttenuationExponent

```
double VectSharp.ThreeD.MaskedLightSource.DistanceAttenuationExponent = 2  [get], [set]
```

An exponent determining how fast the light attenuates with increasing distance. Set to 0 to disable distance attenuation.

Definition at line 420 of file Lights.cs.

6.38.3.5 Intensity

```
double VectSharp.ThreeD.MaskedLightSource.Intensity  [get], [set]
```

The base intensity of the light.

Definition at line 415 of file Lights.cs.

6.38.3.6 Origin

```
Point3D VectSharp.ThreeD.MaskedLightSource.Origin  [get]
```

The projection of the Position on the mask plane along the light's Direction.

Definition at line 398 of file Lights.cs.

6.38.3.7 Position

```
Point3D VectSharp.ThreeD.MaskedLightSource.Position  [get]
```

The position of the light source.

Definition at line 393 of file Lights.cs.

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

- VectSharp.ThreeD/Lights.cs

6.39 VectSharp.ThreeD.ObjectFactory Class Reference

A static class containing methods to create complex 3D objects.

Static Public Member Functions

- static List< Element3D > CreateCube (Point3D center, double size, IEnumerable< IMaterial > fill, string tag=null, int zIndex=0)

  Creates a cube.
- static List< Element3D > CreateCuboid (Point3D center, double sizeX, double sizeY, double sizeZ, IEnumerable< IMaterial > fill, string tag=null, int zIndex=0)

  Creates a cuboid.
- static List< Element3D > CreateRectangle (Point3D point1, Point3D point2, Point3D point3, Point3D point4, IEnumerable< IMaterial > fill, string tag=null, int zIndex=0)

  Creates a quadrilater. All the vertices need not be coplanar.
- static List< Element3D > CreateRectangle (Point3D point1, Point3D point2, Point3D point3, Point3D point4, NormalizedVector3D point1Normal, NormalizedVector3D point2Normal, NormalizedVector3D point3Normal, NormalizedVector3D point4Normal, IEnumerable< IMaterial > fill, string tag=null, int zIndex=0)

  Creates a quadrilater, specifying the vertex normals at the four vertices. All the vertices need not be coplanar.
- static List< Element3D > CreateSphere (Point3D center, double radius, int steps, IEnumerable< IMaterial > fill, string tag=null, int zIndex=0)

  Creates a sphere.
- static List< Element3D > CreateTetrahedron (Point3D center, double radius, IEnumerable< IMaterial > fill, string tag=null, int zIndex=0)

  Creates a tetrahedron inscribed in a sphere.
- static List< Element3D > CreatePolygon (GraphicsPath polygon2D, double triangulationResolution, Point3D origin, NormalizedVector3D xAxis, NormalizedVector3D yAxis, bool reverseTriangles, IEnumerable< IMaterial > fill, string tag=null, int zIndex=0)

  Creates a flat polygon.
- static List< Element3D > CreatePrism (GraphicsPath polygonBase2D, double triangulationResolution, Point3D bottomOrigin, Point3D topOrigin, NormalizedVector3D baseXAxis, NormalizedVector3D baseYAxis, IEnumerable< IMaterial > fill, string tag=null, int zIndex=0)

  Creates a prism with the specified base.
- static List< Element3D > CreateWireframe (IEnumerable< Element3D > object3D, Colour colour, double thickness=1, LineCaps lineCap=LineCaps.Butt, LineDash? lineDash=null, string tag=null, int zIndex=0)

  Creates a wireframe from a collection of Element3Ds.
- static List< Element3D > CreatePoints (IEnumerable< Element3D > object3D, Colour colour, double diameter=1, string tag=null, int zIndex=0)

  Obtains a list of Point3DElement corresponding to the vertices of a list of Element3Ds.

6.39.1 Detailed Description

A static class containing methods to create complex 3D objects.

Definition at line 28 of file ObjectFactory.cs.

6.39.2 Member Function Documentation

6.39.2.1 CreateCube()

```
static List<Element3D> VectSharp.ThreeD.ObjectFactory.CreateCube (
            Point3D center,
            double size,
            IEnumerable< IMaterial > fill,
            string tag = null,
            int zIndex = 0 )  [static]
```

Creates a cube.

Parameters

center	The centre of the cube.
size	The length of each side of the cube.
fill	A collection of materials that will be applied to the Triangle3DElements returned by this method.
tag	A tag that will be applied to the Triangle3DElements returned by this method.
zIndex	A z-index that will be applied to the Triangle3DElements returned by this method.

Returns

A list of Triangle3DElements that constitute the cube.

Definition at line 39 of file ObjectFactory.cs.

6.39.2.2 CreateCuboid()

```
static List<Element3D> VectSharp.ThreeD.ObjectFactory.CreateCuboid (
            Point3D center,
            double sizeX,
            double sizeY,
            double sizeZ,
            IEnumerable< IMaterial > fill,
            string tag = null,
            int zIndex = 0 )  [static]
```

Creates a cuboid.

Parameters

center	The centre of the cube.
sizeX	The length of the sides of the cube parallel to the x axis.
sizeY	The length of the sides of the cube parallel to the y axis.
sizeZ	The length of the sides of the cube parallel to the z axis.
fill	A collection of materials that will be applied to the Triangle3DElements returned by this method.
tag	A tag that will be applied to the Triangle3DElements returned by this method.
zIndex	A z-index that will be applied to the Triangle3DElements returned by this method.

Returns

A list of Triangle3DElements that constitute the cuboid.

Definition at line 55 of file ObjectFactory.cs.

6.39.2.3 CreatePoints()

```
static List<Element3D> VectSharp.ThreeD.ObjectFactory.CreatePoints (
            IEnumerable< Element3D > object3D,
            Colour colour,
            double diameter = 1,
            string tag = null,
            int zIndex = 0 )  [static]
```

Obtains a list of Point3DElement corresponding to the vertices of a list of Element3Ds.

Parameters

object3D	The collection of Element3Ds. Point3DElements are ignored.
colour	The colour of the Point3DElements returned by this method.
diameter	The diameter of the Point3DElements returned by this method.
tag	A tag that will be applied to the Point3DElements returned by this method.
zIndex	A z-index that will be applied to the Point3DElements returned by this method.

Returns

A list of Point3DElements corresponding to the vertices of the Element3Ds.

Definition at line 412 of file ObjectFactory.cs.

6.39.2.4 CreatePolygon()

```
static List<Element3D> VectSharp.ThreeD.ObjectFactory.CreatePolygon (
            GraphicsPath polygon2D,
            double triangulationResolution,
            Point3D origin,
            NormalizedVector3D xAxis,
            NormalizedVector3D yAxis,
            bool reverseTriangles,
            IEnumerable< IMaterial > fill,
            string tag = null,
            int zIndex = 0 )  [static]
```

Creates a flat polygon.

Parameters

polygon2D	A 2D GraphicsPath representing the polygon.
triangulationResolution	The resolution that will be used to linearise curve segments in the GraphicsPath.
origin	A Point3D that will correspond to the origin of the 2D reference system.
xAxis	A NormalizedVector3D that will correspond to the x axis of the 2D reference system. This will be orthonormalised to the *yAxis* .
yAxis	A NormalizedVector3D that will correspond to the y axis of the 2D reference system.
reverseTriangles	Indicates whether the order of the points (and thus the normals) of all the triangles returned by this method should be reversed.
fill	A collection of materials that will be applied to the Triangle3DElements returned by this method.
tag	A tag that will be applied to the Triangle3DElements returned by this method.
zIndex	A z-index that will be applied to the Triangle3DElements returned by this method.

Returns

A list of Triangle3DElements that constitute the polygon.

Definition at line 273 of file ObjectFactory.cs.

6.39.2.5 CreatePrism()

```
static List<Element3D> VectSharp.ThreeD.ObjectFactory.CreatePrism (
            GraphicsPath polygonBase2D,
            double triangulationResolution,
            Point3D bottomOrigin,
            Point3D topOrigin,
            NormalizedVector3D baseXAxis,
            NormalizedVector3D baseYAxis,
            IEnumerable< IMaterial > fill,
            string tag = null,
            int zIndex = 0 )  [static]
```

Creates a prism with the specified base.

Parameters

polygonBase2D	A 2D GraphicsPath representing the base of the prism.
triangulationResolution	The resolution that will be used to linearise curve segments in the GraphicsPath.
bottomOrigin	A Point3D that will correspond to the origin of the 2D reference system of the bottom base.
topOrigin	A Point3D that will correspond to the origin of the 2D reference system of the top base.
baseXAxis	A NormalizedVector3D that will correspond to the x axis of the 2D reference system of the bases. This will be orthonormalised to the *baseYAxis* .
baseYAxis	A NormalizedVector3D that will correspond to the y axis of the 2D reference system of the bases.
fill	A collection of materials that will be applied to the Triangle3DElements returned by this method.
tag	A tag that will be applied to the Triangle3DElements returned by this method.
zIndex	A z-index that will be applied to the Triangle3DElements returned by this method.

Returns

A list of Triangle3DElements that constitute the prism.

Definition at line 314 of file ObjectFactory.cs.

6.39.2.6 CreateRectangle() [1/2]

```
static List<Element3D> VectSharp.ThreeD.ObjectFactory.CreateRectangle (
            Point3D point1,
            Point3D point2,
            Point3D point3,
            Point3D point4,
            IEnumerable< IMaterial > fill,
            string tag = null,
            int zIndex = 0 )  [static]
```

Creates a quadrilater. All the vertices need not be coplanar.

Parameters

point1	The first vertex of the quadrilater.
point2	The second vertex of the quadrilater.
point3	The third vertex of the quadrilater.
point4	The fourth vertex of the quadrilater.
fill	A collection of materials that will be applied to the Triangle3DElements returned by this method.
tag	A tag that will be applied to the Triangle3DElements returned by this method.
zIndex	A z-index that will be applied to the Triangle3DElements returned by this method.

Returns

A list containing two Triangle3DElements representing the quadrilater.

Definition at line 93 of file ObjectFactory.cs.

6.39.2.7 CreateRectangle() [2/2]

```
static List<Element3D> VectSharp.ThreeD.ObjectFactory.CreateRectangle (
            Point3D point1,
            Point3D point2,
            Point3D point3,
            Point3D point4,
            NormalizedVector3D point1Normal,
            NormalizedVector3D point2Normal,
            NormalizedVector3D point3Normal,
            NormalizedVector3D point4Normal,
            IEnumerable< IMaterial > fill,
            string tag = null,
            int zIndex = 0 )  [static]
```

Creates a quadrilater, specifying the vertex normals at the four vertices. All the vertices need not be coplanar.

Parameters

point1	The first vertex of the quadrilater.
point2	The second vertex of the quadrilater.
point3	The third vertex of the quadrilater.
point4	The fourth vertex of the quadrilater.
point1Normal	The vertex normal at the first vertex of the quadrilater.
point2Normal	The vertex normal at the second vertex of the quadrilater.
point3Normal	The vertex normal at the third vertex of the quadrilater.
point4Normal	The vertex normal at the fourth vertex of the quadrilater.
fill	A collection of materials that will be applied to the Triangle3DElements returned by this method.
tag	A tag that will be applied to the Triangle3DElements returned by this method.
zIndex	A z-index that will be applied to the Triangle3DElements returned by this method.

Returns

A list containing two Triangle3DElements representing the quadrilater.

Definition at line 123 of file ObjectFactory.cs.

6.39.2.8 CreateSphere()

```
static List<Element3D> VectSharp.ThreeD.ObjectFactory.CreateSphere (
            Point3D center,
            double radius,
            int steps,
            IEnumerable< IMaterial > fill,
            string tag = null,
            int zIndex = 0 )  [static]
```

Creates a sphere.

Parameters

center	The centre of the sphere.
radius	The radius of the sphere.
steps	The number of meridians and parallels to use when generating the sphere.
fill	A collection of materials that will be applied to the Triangle3DElements returned by this method.
tag	A tag that will be applied to the Triangle3DElements returned by this method.
zIndex	A z-index that will be applied to the Triangle3DElements returned by this method.

Returns

A list of Triangle3DElements that constitute the sphere.

Definition at line 148 of file ObjectFactory.cs.

6.39.2.9 CreateTetrahedron()

```
static List<Element3D> VectSharp.ThreeD.ObjectFactory.CreateTetrahedron (
            Point3D center,
            double radius,
            IEnumerable< IMaterial > fill,
            string tag = null,
            int zIndex = 0 )  [static]
```

Creates a tetrahedron inscribed in a sphere.

Parameters

center	The centre of the tetrahedron.
radius	The radius of the sphere in which the tetrahedron is inscribed.
fill	A collection of materials that will be applied to the Triangle3DElements returned by this method.
tag	A tag that will be applied to the Triangle3DElements returned by this method.
zIndex	A z-index that will be applied to the Triangle3DElements returned by this method.

Returns

A list of Triangle3DElements that constitute the sphere.

Definition at line 238 of file ObjectFactory.cs.

6.39.2.10 CreateWireframe()

```
static List<Element3D> VectSharp.ThreeD.ObjectFactory.CreateWireframe (
            IEnumerable< Element3D > object3D,
            Colour colour,
            double thickness = 1,
            LineCaps lineCap = LineCaps.Butt,
            LineDash?  lineDash = null,
            string tag = null,
            int zIndex = 0 )  [static]
```

Creates a wireframe from a collection of Element3Ds.

Parameters

object3D	The collection of Element3Ds. Line3DElements and Point3DElements are ignored.
colour	The colour of the Line3DElements returned by this method.
thickness	The thickness of the Line3DElements returned by this method.
lineCap	The line cap of the Line3DElements returned by this method.
lineDash	The line dash of the Line3DElements returned by this method.
tag	A tag that will be applied to the Line3DElements returned by this method.
zIndex	A z-index that will be applied to the Line3DElements returned by this method.

Returns

A list of Line3DElements that constitute the wireframe.

Definition at line 370 of file ObjectFactory.cs.

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

- VectSharp.ThreeD/ObjectFactory.cs

6.40 VectSharp.Page Class Reference

Represents a Graphics object with a width and height.

Public Member Functions

- Page (double width, double height)

  Create a new page.
- void Crop (Point topLeft, Size size)

  Translate and resize the Page so that it displays the rectangle defined by topLeft and size .

Properties

- double Width `[get, set]`

  Width of the page.
- double Height `[get, set]`

  Height of the page.
- Graphics Graphics `[get, set]`

  Graphics surface of the page.
- Colour Background = Colour.FromRgba(255, 255, 255, 0) `[get, set]`

  Background colour of the page.

6.40.1 Detailed Description

Represents a Graphics object with a width and height.

Definition at line 47 of file Document.cs.

6.40.2 Constructor & Destructor Documentation

6.40.2.1 Page()

```
VectSharp.Page.Page (
            double width,
            double height )
```

Create a new page.

Parameters

width	The width of the page.
height	The height of the page.

Definition at line 74 of file Document.cs.

6.40.3 Member Function Documentation

6.40.3.1 Crop()

```
void VectSharp.Page.Crop (
            Point topLeft,
            Size size )
```

Translate and resize the Page so that it displays the rectangle defined by *topLeft* and *size* .

Parameters

topLeft	The top left corner of the area to include in the page.
size	The size of the area to include in the page.

Definition at line 88 of file Document.cs.

6.40.4 Property Documentation

6.40.4.1 Background

```
Colour VectSharp.Page.Background = Colour.FromRgba(255, 255, 255, 0)  [get], [set]
```

Background colour of the page.

Definition at line 67 of file Document.cs.

6.40.4.2 Graphics

```
Graphics VectSharp.Page.Graphics  [get], [set]
```

Graphics surface of the page.

Definition at line 62 of file Document.cs.

6.40.4.3 Height

```
double VectSharp.Page.Height  [get], [set]
```

Height of the page.

Definition at line 57 of file Document.cs.

6.40.4.4 Width

```
double VectSharp.Page.Width  [get], [set]
```

Width of the page.

Definition at line 52 of file Document.cs.

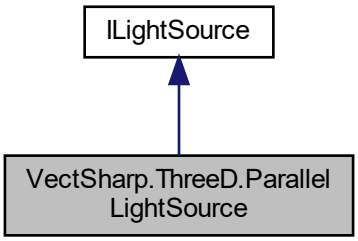
The documentation for this class was generated from the following file:

- VectSharp/Document.cs

6.41 VectSharp.ThreeD.ParallelLightSource Class Reference

Represents a parallel light source.

Inheritance diagram for VectSharp.ThreeD.ParallelLightSource:



Public Member Functions

- ParallelLightSource (double intensity, NormalizedVector3D direction)

  Creates a new ParallelLightSource instance.
- LightIntensity GetLightAt (Point3D point)

  Computes the light intensity at the specified point, without taking into account any obstructions.
- double GetObstruction (Point3D point, IEnumerable< Triangle3DElement > shadowingTriangles)

  Determines the amount of obstruction of the light that results at point due to the specified shadowingTriangles .

Properties

- double Intensity [get, set]

  The intensity of the light.
- NormalizedVector3D Direction [get]

  The direction along which the light travels.
- NormalizedVector3D ReverseDirection [get]

  The reverse of Direction.
- bool CastsShadow = true [get, set]

6.41.1 Detailed Description

Represents a parallel light source.

Definition at line 126 of file Lights.cs.

6.41.2 Constructor & Destructor Documentation

6.41.2.1 ParallelLightSource()

```
VectSharp.ThreeD.ParallelLightSource.ParallelLightSource (
            double intensity,
            NormalizedVector3D direction )
```

Creates a new ParallelLightSource instance.

Parameters

intensity	The intensity of the light.
direction	The direction along which the light travels.

Definition at line 151 of file Lights.cs.

6.41.3 Property Documentation

6.41.3.1 Direction

```
NormalizedVector3D VectSharp.ThreeD.ParallelLightSource.Direction [get]
```

The direction along which the light travels.

Definition at line 136 of file Lights.cs.

6.41.3.2 Intensity

```
double VectSharp.ThreeD.ParallelLightSource.Intensity  [get], [set]
```

The intensity of the light.

Definition at line 131 of file Lights.cs.

6.41.3.3 ReverseDirection

```
NormalizedVector3D VectSharp.ThreeD.ParallelLightSource.ReverseDirection  [get]
```

The reverse of Direction.

Definition at line 141 of file Lights.cs.

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

- VectSharp.ThreeD/Lights.cs

6.42 VectSharp.SVG.Parser Class Reference

Contains methods to read an SVG image file.

Static Public Member Functions

- static Page ParseSVGURI (string uri, bool ignored=false)

  Parses an SVG image URI.
- static Page FromString (string svgSource)

  Parses SVG source into a Page containing the image represented by the code.
- static Page FromFile (string fileName)

  Parses an SVG image file into a Page containing the image.
- static Page FromStream (Stream svgSourceStream)

  Parses an stream containing SVG source code into a Page containing the image represented by the code.

Static Public Attributes

- static Func< string, bool, Page > ParseImageURI

  A function that takes as input an image URI and a boolean value indicating whether the image should be interpolated, and returns a Page object containing the image. By default, this is equal to ParseSVGURI, i.e. it is only able to parse SVG images. If you wish to enable the parsing of other formats, you should install the "VectSharp.MuPDFUtils" NuGet package and enable the parser in your program by doing something like:

6.42.1 Detailed Description

Contains methods to read an SVG image file.

Definition at line 32 of file SVGParser.cs.

6.42.2 Member Function Documentation

6.42.2.1 FromFile()

```
static Page VectSharp.SVG.Parser.FromFile (
            string fileName )  [static]
```

Parses an SVG image file into a Page containing the image.

Parameters

fileName	The path to the SVG image file.

Returns

A Page containing the image represented by the file.

Definition at line 154 of file SVGParser.cs.

6.42.2.2 FromStream()

```
static Page VectSharp.SVG.Parser.FromStream (
            Stream svgSourceStream )  [static]
```

Parses an stream containing SVG source code into a Page containing the image represented by the code.

Parameters

svgSourceStream	The stream containing SVG source code.

Returns

A Page containing the image represented by the *svgSourceStream* .

Definition at line 164 of file SVGParser.cs.

6.42.2.3 FromString()

```
static Page VectSharp.SVG.Parser.FromString (
            string svgSource )  [static]
```

Parses SVG source into a Page containing the image represented by the code.

Parameters

svgSource	The SVG source code.

Returns

A Page containing the image represented by the *svgSource* .

Definition at line 102 of file SVGParser.cs.

6.42.2.4 ParseSVGURI()

```
static Page VectSharp.SVG.Parser.ParseSVGURI (
            string uri,
            bool ignored = false )  [static]
```

Parses an SVG image URI.

Parameters

uri	The image URI to parse.
ignored	This value is ignored and is only needed for compatibility.

Returns

A Page containing the parsed SVG image, or null.

Definition at line 53 of file SVGParser.cs.

6.42.3 Member Data Documentation

6.42.3.1 ParseImageURI

```
Func<string, bool, Page> VectSharp.SVG.Parser.ParseImageURI  [static]
```

A function that takes as input an image URI and a boolean value indicating whether the image should be interpolated, and returns a Page object containing the image. By default, this is equal to ParseSVGURI, i.e. it is only able

to parse SVG images. If you wish to enable the parsing of other formats, you should install the "VectSharp.MuP↩
DFUtils" NuGet package and enable the parser in your program by doing something like:

```
VectSharp.SVG.Parser.ParseImageURI = VectSharp.MuPDFUtils.ImageURIParser.Parser(VectShar
```

Definition at line 45 of file SVGParser.cs.

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

- VectSharp.SVG/SVGParser.cs

6.43 VectSharp.PDF.PDFContextInterpreter Class Reference

Contains methods to render a Document as a PDF document.

Public Types

- enum TextOptions { TextOptions.SubsetFonts, TextOptions.ConvertIntoPaths }

  Defines whether the used fonts should be included in the file.

Static Public Member Functions

- static void SaveAsPDF (this Document document, string fileName, TextOptions textOption=TextOptions.SubsetFonts, bool compressStreams=true, Dictionary< string, string > linkDestinations=null)

  Save the document to a PDF file.
- static void SaveAsPDF (this Document document, Stream stream, TextOptions textOption=TextOptions.SubsetFonts, bool compressStreams=true, Dictionary< string, string > linkDestinations=null)

  Save the document to a PDF stream.

6.43.1 Detailed Description

Contains methods to render a Document as a PDF document.

Definition at line 585 of file PDFContext.cs.

6.43.2 Member Enumeration Documentation

6.43.2.1 TextOptions

```
enum VectSharp.PDF.PDFContextInterpreter.TextOptions  [strong]
```

Defines whether the used fonts should be included in the file.

Enumerator

SubsetFonts	Embeds subsetted font files containing only the glyphs for the characters that have been used.
ConvertIntoPaths	Does not embed any font file and converts all text items into paths.

Definition at line 795 of file PDFContext.cs.

6.43.3 Member Function Documentation

6.43.3.1 SaveAsPDF() [1/2]

```
static void VectSharp.PDF.PDFContextInterpreter.SaveAsPDF (
            this Document document,
            Stream stream,
            TextOptions textOption = TextOptions.SubsetFonts,
            bool compressStreams = true,
            Dictionary< string, string > linkDestinations = null )  [static]
```

Save the document to a PDF stream.

Parameters

document	The Document to save.
stream	The stream to which the PDF data will be written.
textOption	Defines whether the used fonts should be included in the file.
compressStreams	Indicates whether the streams in the PDF file should be compressed.
linkDestinations	A dictionary associating element tags to link targets. If this is provided, objects that have been drawn with a tag contained in the dictionary will become hyperlink to the destination specified in the dictionary. If the destination starts with a hash (#), it is interpreted as the tag of another object in the current document; otherwise, it is interpreted as an external URI.

Definition at line 818 of file PDFContext.cs.

6.43.3.2 SaveAsPDF() [2/2]

```
static void VectSharp.PDF.PDFContextInterpreter.SaveAsPDF (
            this Document document,
            string fileName,
            TextOptions textOption = TextOptions.SubsetFonts,
            bool compressStreams = true,
            Dictionary< string, string > linkDestinations = null )  [static]
```

Save the document to a PDF file.

Parameters

document	The Document to save.
fileName	The full path to the file to save. If it exists, it will be overwritten.
textOption	Defines whether the used fonts should be included in the file.
compressStreams	Indicates whether the streams in the PDF file should be compressed.
linkDestinations	A dictionary associating element tags to link targets. If this is provided, objects that have been drawn with a tag contained in the dictionary will become hyperlink to the destination specified in the dictionary. If the destination starts with a hash (#), it is interpreted as the tag of another object in the current document; otherwise, it is interpreted as an external URI.

Definition at line 784 of file PDFContext.cs.

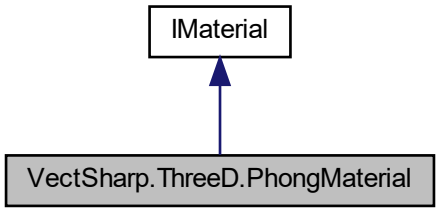
The documentation for this class was generated from the following file:

- VectSharp.PDF/PDFContext.cs

6.44 VectSharp.ThreeD.PhongMaterial Class Reference

Represents a material that uses a Phong reflection model to determine the colour of the material based on the light sources that hit it.

Inheritance diagram for VectSharp.ThreeD.PhongMaterial:



Public Member Functions

- PhongMaterial (Colour colour)

  Creates a new PhongMaterial instance.
- Colour GetColour (Point3D point, NormalizedVector3D surfaceNormal, Camera camera, IList< ILightSource > lights, IList< double > obstructions)

  Obtains the Colour at the specified point.

Properties

- Colour Colour [get]

  The base colour of the material.
- double AmbientReflectionCoefficient = 1 [get, set]

  A coefficient determining how much ambient light is reflected by the material.
- double DiffuseReflectionCoefficient = 1 [get, set]

  A coefficient determining how much directional light is reflected by the material.
- double SpecularReflectionCoefficient = 1 [get, set]

  A coefficient determining the intensity of specular highlights.
- double SpecularShininess = 1 [get, set]

  A coefficient determining the extent of specular highlights.

6.44.1 Detailed Description

Represents a material that uses a Phong reflection model to determine the colour of the material based on the light sources that hit it.

Definition at line 74 of file Materials.cs.

6.44.2 Constructor & Destructor Documentation

6.44.2.1 PhongMaterial()

```
VectSharp.ThreeD.PhongMaterial.PhongMaterial (
            Colour colour )
```

Creates a new PhongMaterial instance.

Parameters

colour	The base colour of the material.

Definition at line 111 of file Materials.cs.

6.44.3 Property Documentation

6.44.3.1 AmbientReflectionCoefficient

```
double VectSharp.ThreeD.PhongMaterial.AmbientReflectionCoefficient = 1 [get], [set]
```

A coefficient determining how much ambient light is reflected by the material.

Definition at line 90 of file Materials.cs.

6.44.3.2 Colour

```
Colour VectSharp.ThreeD.PhongMaterial.Colour  [get]
```

The base colour of the material.

Definition at line 79 of file Materials.cs.

6.44.3.3 DiffuseReflectionCoefficient

```
double VectSharp.ThreeD.PhongMaterial.DiffuseReflectionCoefficient = 1  [get], [set]
```

A coefficient determining how much directional light is reflected by the material.

Definition at line 95 of file Materials.cs.

6.44.3.4 SpecularReflectionCoefficient

```
double VectSharp.ThreeD.PhongMaterial.SpecularReflectionCoefficient = 1  [get], [set]
```

A coefficient determining the intensity of specular highlights.

Definition at line 100 of file Materials.cs.

6.44.3.5 SpecularShininess

```
double VectSharp.ThreeD.PhongMaterial.SpecularShininess = 1  [get], [set]
```

A coefficient determining the extent of specular highlights.

Definition at line 105 of file Materials.cs.

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

- VectSharp.ThreeD/Materials.cs

6.45 VectSharp.Point Struct Reference

Represents a point relative to an origin in the top-left corner.

Public Member Functions

- Point (double x, double y)

  Create a new Point.
- double Modulus ()

  Computes the modulus of the vector represented by the Point.
- Point Normalize ()

  Normalises a Point.
- bool IsEqual (Point p2, double tolerance)

  Checks whether this Point is equal to another Point, up to a specified tolerance.

Public Attributes

- double X

  Horizontal (x) coordinate, measured to the right of the origin.
- double Y

  Vertical (y) coordinate, measured to the bottom of the origin.

6.45.1 Detailed Description

Represents a point relative to an origin in the top-left corner.

Definition at line 25 of file Point.cs.

6.45.2 Constructor & Destructor Documentation

6.45.2.1 Point()

```
VectSharp.Point.Point (
            double x,
            double y )
```

Create a new Point.

Parameters

x	The horizontal (x) coordinate.
y	The vertical (y) coordinate.

Definition at line 42 of file Point.cs.

6.45.3 Member Function Documentation

6.45.3.1 IsEqual()

```
bool VectSharp.Point.IsEqual (
            Point p2,
            double tolerance )
```

Checks whether this Point is equal to another Point, up to a specified tolerance.

Parameters

p2	The Point to compare.
tolerance	The tolerance threshold.

Returns

`true` if both coordinates of the Points are closer than *tolerance* or if their relative difference (i.e. `(a - b) / (a + b) * 2`) is smaller than *tolerance* . `false` otherwise.

Definition at line 73 of file Point.cs.

6.45.3.2 Modulus()

```
double VectSharp.Point.Modulus ( )
```

Computes the modulus of the vector represented by the Point.

Returns

The modulus of the vector represented by the Point.

Definition at line 52 of file Point.cs.

6.45.3.3 Normalize()

```
Point VectSharp.Point.Normalize ( )
```

Normalises a Point.

Returns

The normalised Point.

Definition at line 61 of file Point.cs.

6.45.4 Member Data Documentation

6.45.4.1 X

```
double VectSharp.Point.X
```

Horizontal (x) coordinate, measured to the right of the origin.

Definition at line 30 of file Point.cs.

6.45.4.2 Y

```
double VectSharp.Point.Y
```

Vertical (y) coordinate, measured to the bottom of the origin.

Definition at line 35 of file Point.cs.

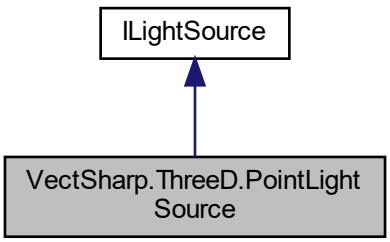
The documentation for this struct was generated from the following file:

- VectSharp/Point.cs

6.46 VectSharp.ThreeD.PointLightSource Class Reference

Represents a point light source.

Inheritance diagram for VectSharp.ThreeD.PointLightSource:



Public Member Functions

- PointLightSource (double intensity, Point3D position)

  Creates a new PointLightSource instance.
- LightIntensity GetLightAt (Point3D point)

  Computes the light intensity at the specified point, without taking into account any obstructions.
- double GetObstruction (Point3D point, IEnumerable< Triangle3DElement > shadowingTriangles)

  Determines the amount of obstruction of the light that results at point due to the specified shadowingTriangles .

Properties

- bool CastsShadow = true `[get, set]`
- Point3D Position `[get, set]`

  The position of the light source.
- double Intensity `[get, set]`

  The base intensity of the light.
- double DistanceAttenuationExponent = 2 `[get, set]`

  An exponent determining how fast the light attenuates with increasing distance. Set to 0 to disable distance attenuation.

6.46.1 Detailed Description

Represents a point light source.

Definition at line 184 of file Lights.cs.

6.46.2 Constructor & Destructor Documentation

6.46.2.1 PointLightSource()

```
VectSharp.ThreeD.PointLightSource.PointLightSource (
            double intensity,
            Point3D position )
```

Creates a new PointLightSource instance.

Parameters

intensity	The intensity of the light.
position	The position of the light source.

Definition at line 209 of file Lights.cs.

6.46.3 Property Documentation

6.46.3.1 DistanceAttenuationExponent

```
double VectSharp.ThreeD.PointLightSource.DistanceAttenuationExponent = 2  [get], [set]
```

An exponent determining how fast the light attenuates with increasing distance. Set to 0 to disable distance attenuation.

Definition at line 202 of file Lights.cs.

6.46.3.2 Intensity

```
double VectSharp.ThreeD.PointLightSource.Intensity  [get], [set]
```

The base intensity of the light.

Definition at line 197 of file Lights.cs.

6.46.3.3 Position

```
Point3D VectSharp.ThreeD.PointLightSource.Position  [get], [set]
```

The position of the light source.

Definition at line 192 of file Lights.cs.

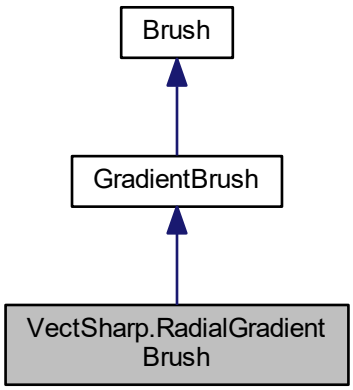
The documentation for this class was generated from the following file:

- VectSharp.ThreeD/Lights.cs

6.47 VectSharp.RadialGradientBrush Class Reference

Represents a brush painting with a radial gradient.

Inheritance diagram for VectSharp.RadialGradientBrush:



Public Member Functions

- RadialGradientBrush (Point focalPoint, Point centre, double radius, params GradientStop[ ] gradientStops)

  Creates a new RadialGradientBrush with the specified focal point, centre, radius and gradient stops.
- RadialGradientBrush (Point focalPoint, Point centre, double radius, IEnumerable< GradientStop > gradientStops)

  Creates a new RadialGradientBrush with the specified focal point, centre, radius and gradient stops.
- override Brush MultiplyOpacity (double opacity)

  Returns a brush corresponding the current instance, with the specified opacity multiplication applied.

Properties

- Point FocalPoint `[get]`

  The focal point of the gradient (i.e. the point within the circle where the gradient starts).
- Point Centre `[get]`

  Represents the centre of the gradient.
- double Radius `[get]`

  The radius of the gradient.

Additional Inherited Members

6.47.1 Detailed Description

Represents a brush painting with a radial gradient.

Definition at line 367 of file Brush.cs.

6.47.2 Constructor & Destructor Documentation

6.47.2.1 RadialGradientBrush() `[1/2]`

```
VectSharp.RadialGradientBrush.RadialGradientBrush (
            Point focalPoint,
            Point centre,
            double radius,
            params GradientStop[] gradientStops )
```

Creates a new RadialGradientBrush with the specified focal point, centre, radius and gradient stops.

Parameters

focalPoint	The focal point of the gradient. Note that this is relative to the current coordinate system when the gradient is used.
centre	The centre of the gradient. Note that this is relative to the current coordinate system when the gradient is used.
radius	The radius of the gradient. Note that this is relative to the current coordinate system when the gradient is used.
gradientStops	The colour stops in the gradient.

Definition at line 391 of file Brush.cs.

6.47.2.2 RadialGradientBrush() [2/2]

```
VectSharp.RadialGradientBrush.RadialGradientBrush (
            Point focalPoint,
            Point centre,
            double radius,
            IEnumerable< GradientStop > gradientStops )
```

Creates a new RadialGradientBrush with the specified focal point, centre, radius and gradient stops.

Parameters

focalPoint	The focal point of the gradient. Note that this is relative to the current coordinate system when the gradient is used.
centre	The centre of the gradient. Note that this is relative to the current coordinate system when the gradient is used.
radius	The radius of the gradient. Note that this is relative to the current coordinate system when the gradient is used.
gradientStops	The colour stops in the gradient.

Definition at line 430 of file Brush.cs.

6.47.3 Property Documentation

6.47.3.1 Centre

```
Point VectSharp.RadialGradientBrush.Centre  [get]
```

Represents the centre of the gradient.

Definition at line 377 of file Brush.cs.

6.47.3.2 FocalPoint

```
Point VectSharp.RadialGradientBrush.FocalPoint  [get]
```

The focal point of the gradient (i.e. the point within the circle where the gradient starts).

Definition at line 372 of file Brush.cs.

6.47.3.3 Radius

```
double VectSharp.RadialGradientBrush.Radius  [get]
```

The radius of the gradient.

Definition at line 382 of file Brush.cs.

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

- VectSharp/Brush.cs

6.48 VectSharp.Raster.Raster Class Reference

Contains methods to render a page to a PNG image.

Static Public Member Functions

- static void SaveAsPNG (this Page page, string fileName, double scale=1)
  Render the page to a PNG file.
- static void SaveAsPNG (this Page page, Stream stream, double scale=1)
  Render the page to a PNG stream.

6.48.1 Detailed Description

Contains methods to render a page to a PNG image.

Definition at line 27 of file Raster.cs.

6.48.2 Member Function Documentation

6.48.2.1 SaveAsPNG() [1/2]

```
static void VectSharp.Raster.Raster.SaveAsPNG (
            this Page page,
            Stream stream,
            double scale = 1 )  [static]
```

Render the page to a PNG stream.

Parameters

page	The Page to render.
stream	The stream to which the PNG data will be written.
scale	The scale to be used when rasterising the page. This will determine the width and height of the image file.

Definition at line 59 of file Raster.cs.

6.48.2.2 SaveAsPNG() [2/2]

```
static void VectSharp.Raster.Raster.SaveAsPNG (
            this Page page,
            string fileName,
            double scale = 1 )  [static]
```

Render the page to a PNG file.

Parameters

page	The Page to render.
fileName	The full path to the file to save. If it exists, it will be overwritten.
scale	The scale to be used when rasterising the page. This will determine the width and height of the image file.

Definition at line 36 of file Raster.cs.

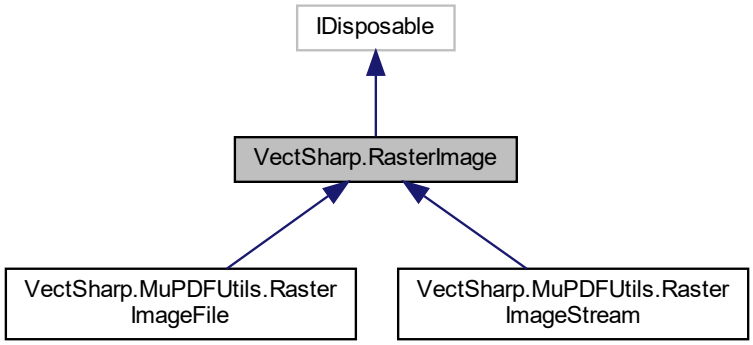
The documentation for this class was generated from the following file:

- VectSharp.Raster/Raster.cs

6.49 VectSharp.RasterImage Class Reference

Represents a raster image, created from raw pixel data. Consider using the derived classes included in the NuGet package "VectSharp.MuPDFUtils" if you need to load a raster image from a file or a Stream.

Inheritance diagram for VectSharp.RasterImage:



Public Member Functions

- RasterImage (IntPtr pixelData, int width, int height, bool hasAlpha, bool interpolate)

  Creates a new RasterImage instance from the specified pixel data in RGB or RGBA format.
- RasterImage (ref DisposableIntPtr pixelData, int width, int height, bool hasAlpha, bool interpolate)

  Creates a new RasterImage instance from the specified pixel data in RGB or RGBA format.
- RasterImage (byte[ ] data, int width, int height, PixelFormats pixelFormat, bool interpolate)

  Creates a new RasterImage instance copying the specified pixel data.
- void ClearPNGCache ()

  Disposes the PNGStream. Also useful if is is necessary to regenerate it, e.g. because the underlying image pixel data has changed.
- void Dispose ()

Properties

- IntPtr ImageDataAddress `[get]`

  The memory address of the image pixel data.
- IDisposable DataHolder `[get]`

  An IDisposable that will be disposed when the image is disposed.
- string Id `[get]`

  A univocal identifier for this image.
- bool HasAlpha `[get]`

  Determines whether the image has an alpha channel.
- int Width `[get]`

  The width in pixels of the image.
- int Height `[get]`

  The height in pixels of the image.
- bool Interpolate `[get]`

  Determines whether the image should be interpolated when it is resized.
- MemoryStream PNGStream `[get]`

  Contains a representation of the image in PNG format. Generated at the first access and cached until the image is disposed.

6.49.1 Detailed Description

Represents a raster image, created from raw pixel data. Consider using the derived classes included in the NuGet package "VectSharp.MuPDFUtils" if you need to load a raster image from a file or a Stream.

Definition at line 98 of file RasterImage.cs.

6.49.2 Constructor & Destructor Documentation

6.49.2.1 RasterImage() [1/3]

```
VectSharp.RasterImage.RasterImage (
            IntPtr pixelData,
            int width,
            int height,
            bool hasAlpha,
            bool interpolate )
```

Creates a new RasterImage instance from the specified pixel data in RGB or RGBA format.

Parameters

pixelData	The address of the image pixel data in RGB or RGBA format.
width	The width in pixels of the image.
height	The height in pixels of the image.
hasAlpha	true if the image is in RGBA format, false if it is in RGB format.
interpolate	Whether the image should be interpolated when it is resized.

Definition at line 170 of file RasterImage.cs.

6.49.2.2 RasterImage() [2/3]

```
VectSharp.RasterImage.RasterImage (
            ref DisposableIntPtr pixelData,
            int width,
            int height,
            bool hasAlpha,
            bool interpolate )
```

Creates a new RasterImage instance from the specified pixel data in RGB or RGBA format.

Parameters

pixelData	The address of the image pixel data in RGB or RGBA format wrapped in a DisposableIntPtr. The RasterImage will take ownership of this memory.
width	The width in pixels of the image.
height	The height in pixels of the image.
hasAlpha	true if the image is in RGBA format, false if it is in RGB format.
interpolate	Whether the image should be interpolated when it is resized.

Definition at line 188 of file RasterImage.cs.

6.49.2.3 RasterImage() [3/3]

```
VectSharp.RasterImage.RasterImage (
            byte[] data,
            int width,
            int height,
            PixelFormats pixelFormat,
            bool interpolate )
```

Creates a new RasterImage instance copying the specified pixel data.

Parameters

data	The image pixel data that will be copied.

Parameters

width	The width in pixels of the image.
height	The height in pixels of the image.
pixelFormat	The format of the pixel data.
interpolate	Whether the image should be interpolated when it is resized.

Definition at line 207 of file RasterImage.cs.

6.49.3 Member Function Documentation

6.49.3.1 ClearPNGCache()

```
void VectSharp.RasterImage.ClearPNGCache ( )
```

Disposes the PNGStream. Also useful if is is necessary to regenerate it, e.g. because the underlying image pixel data has changed.

Definition at line 261 of file RasterImage.cs.

6.49.4 Property Documentation

6.49.4.1 DataHolder

```
IDisposable VectSharp.RasterImage.DataHolder  [get]
```

An IDisposable that will be disposed when the image is disposed.

Definition at line 108 of file RasterImage.cs.

6.49.4.2 HasAlpha

```
bool VectSharp.RasterImage.HasAlpha  [get]
```

Determines whether the image has an alpha channel.

Definition at line 118 of file RasterImage.cs.

6.49.4.3 Height

```
int VectSharp.RasterImage.Height  [get]
```

The height in pixels of the image.

Definition at line 128 of file RasterImage.cs.

6.49.4.4 Id

```
string VectSharp.RasterImage.Id  [get]
```

A univocal identifier for this image.

Definition at line 113 of file RasterImage.cs.

6.49.4.5 ImageDataAddress

```
IntPtr VectSharp.RasterImage.ImageDataAddress  [get]
```

The memory address of the image pixel data.

Definition at line 103 of file RasterImage.cs.

6.49.4.6 Interpolate

```
bool VectSharp.RasterImage.Interpolate  [get]
```

Determines whether the image should be interpolated when it is resized.

Definition at line 133 of file RasterImage.cs.

6.49.4.7 PNGStream

```
MemoryStream VectSharp.RasterImage.PNGStream  [get]
```

Contains a representation of the image in PNG format. Generated at the first access and cached until the image is disposed.

Definition at line 140 of file RasterImage.cs.

6.49.4.8 Width

```
int VectSharp.RasterImage.Width  [get]
```

The width in pixels of the image.

Definition at line 123 of file RasterImage.cs.

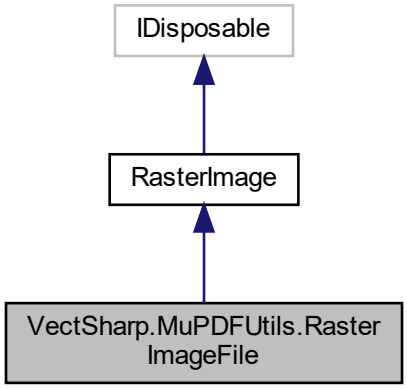
The documentation for this class was generated from the following file:

- VectSharp/RasterImage.cs

6.50 VectSharp.MuPDFUtils.RasterImageFile Class Reference

A RasterImage created from a file.

Inheritance diagram for VectSharp.MuPDFUtils.RasterImageFile:



Public Member Functions

- RasterImageFile (string fileName, int pageNumber=0, double scale=1, bool alpha=true, bool interpolate=true)
  Creates a new RasterImage from the specified file.

Additional Inherited Members

6.50.1 Detailed Description

A RasterImage created from a file.

Definition at line 28 of file RasterImages.cs.

6.50.2 Constructor & Destructor Documentation

6.50.2.1 RasterImageFile()

```
VectSharp.MuPDFUtils.RasterImageFile.RasterImageFile (
            string fileName,
            int pageNumber = 0,
            double scale = 1,
            bool alpha = true,
            bool interpolate = true )
```

Creates a new RasterImage from the specified file.

Parameters

fileName	The path to the file containing the image.
pageNumber	The number of the page in the file from which the image should be created, starting at 0. Only useful for multi-page formats, such as PDF.
scale	The scale factor at which to render the image.
alpha	A boolean value indicating whether transparency (alpha) data from the image should be preserved or not.
interpolate	A boolean value indicating whether the image should be interpolated when it is resized or not.

Definition at line 38 of file RasterImages.cs.

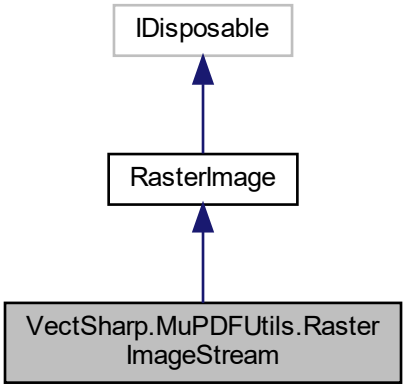
The documentation for this class was generated from the following file:

- VectSharp.MuPDFUtils/RasterImages.cs

6.51 VectSharp.MuPDFUtils.RasterImageStream Class Reference

A RasterImage created from a stream.

Inheritance diagram for VectSharp.MuPDFUtils.RasterImageStream:



Public Member Functions

- RasterImageStream (Stream imageStream, InputFileTypes fileType, int pageNumber=0, double scale=1, bool alpha=true, bool interpolate=true)

  Creates a new RasterImage from the specified stream.
- RasterImageStream (IntPtr imageAddress, long imageLength, InputFileTypes fileType, int pageNumber=0, double scale=1, bool alpha=true, bool interpolate=true)

  Creates a new RasterImage from the specified stream.

Additional Inherited Members

6.51.1 Detailed Description

A RasterImage created from a stream.

Definition at line 69 of file RasterImages.cs.

6.51.2 Constructor & Destructor Documentation

6.51.2.1 RasterImageStream() [1/2]

```
VectSharp.MuPDFUtils.RasterImageStream.RasterImageStream (
            Stream imageStream,
            InputFileTypes fileType,
            int pageNumber = 0,
            double scale = 1,
            bool alpha = true,
            bool interpolate = true )
```

Creates a new RasterImage from the specified stream.

Parameters

imageStream	The stream containing the image data.
fileType	The type of the image contained in the stream.
pageNumber	The number of the page in the file from which the image should be created, starting at 0. Only useful for multi-page formats, such as PDF.
scale	The scale factor at which to render the image.
alpha	A boolean value indicating whether transparency (alpha) data from the image should be preserved or not.
interpolate	A boolean value indicating whether the image should be interpolated when it is resized or not.

Definition at line 80 of file RasterImages.cs.

6.51.2.2 RasterImageStream() [2/2]

```
VectSharp.MuPDFUtils.RasterImageStream.RasterImageStream (
            IntPtr imageAddress,
            long imageLength,
            InputFileTypes fileType,
            int pageNumber = 0,
            double scale = 1,
            bool alpha = true,
            bool interpolate = true )
```

Creates a new RasterImage from the specified stream.

Parameters

imageAddress	A pointer to the address where the image data is contained.
imageLength	The length in bytes of the image data.
fileType	The type of the image contained in the stream.
pageNumber	The number of the page in the file from which the image should be created, starting at 0. Only useful for multi-page formats, such as PDF.
scale	The scale factor at which to render the image.
alpha	A boolean value indicating whether transparency (alpha) data from the image should be preserved or not.
interpolate	A boolean value indicating whether the image should be interpolated when it is resized or not.

Definition at line 148 of file RasterImages.cs.

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

- VectSharp.MuPDFUtils/RasterImages.cs

6.52 VectSharp.Canvas.RenderAction Class Reference

Represents a light-weight rendering action.

Public Types

- enum ActionTypes { ActionTypes.Path, ActionTypes.Text, ActionTypes.RasterImage }

  Types of rendering actions.

Public Member Functions

- void BringToFront ()

  Brings the render action to the front of the rendering queue. This method can only be invoked after the output has been fully initialised.
- void SendToBack ()

  Brings the render action to the back of the rendering queue. This method can only be invoked after the output has been fully initialised.

Static Public Member Functions

- static RenderAction PathAction (Geometry geometry, Pen stroke, IBrush fill, Avalonia.Matrix transform, Geometry clippingPath, string tag=null)

  Creates a new RenderAction representing a path.
- static RenderAction TextAction (Avalonia.Media.FormattedText text, IBrush fill, Avalonia.Matrix transform, Geometry clippingPath, string tag=null)

  Creates a new RenderAction representing text.
- static RenderAction ImageAction (string imageId, Avalonia.Rect sourceRect, Avalonia.Rect destinationRect, Avalonia.Matrix transform, Geometry clippingPath, string tag=null)

  Creates a new RenderAction representing an image.

Properties

- ActionTypes ActionType `[get]`

  Type of the rendering action.
- Geometry Geometry `[get, set]`

  Geometry that needs to be rendered (null if the action type is ActionTypes.Text). If you change this, you need to invalidate the Parent's visual.
- Avalonia.Media.FormattedText Text `[get, set]`

  Text that needs to be rendered (null if the action type is ActionTypes.Path). If you change this, you need to invalidate the Parent's visual.
- Pen Stroke `[get, set]`

  Rendering stroke (null if the action type is ActionTypes.Text or if the rendered action only has a Fill). If you change this, you need to invalidate the Parent's visual.
- IBrush Fill `[get, set]`

  Rendering fill (null if the rendered action only has a Stroke). If you change this, you need to invalidate the Parent's visual.
- string ImageId `[get, set]`

  Univocal identifier of the image that needs to be drawn.
- Avalonia.? Rect ImageSource `[get, set]`

  The source rectangle of the image.
- Avalonia.? Rect ImageDestination `[get, set]`

  The destination rectangle of the image.
- Geometry ClippingPath `[get, set]`

  The current clipping path.

- Avalonia.Matrix InverseTransform = Avalonia.Matrix.Identity `[get]`

  Inverse transformation matrix.
- Avalonia.Matrix Transform `[get, set]`

  Rendering transformation matrix. If you change this, you need to invalidate the Parent's visual.
- string Tag `[get, set]`

  A tag to access the RenderAction.
- Avalonia.Controls.Canvas Parent `[get]`

  The container of this RenderAction.

Events

- EventHandler< Avalonia.Input.PointerEventArgs > PointerEnter

  Raised when the pointer enters the area covered by the RenderAction.
- EventHandler< Avalonia.Input.PointerEventArgs > PointerLeave

  Raised when the pointer leaves the area covered by the RenderAction.
- EventHandler< Avalonia.Input.PointerPressedEventArgs > PointerPressed

  Raised when the pointer is pressed while over the area covered by the RenderAction.
- EventHandler< Avalonia.Input.PointerReleasedEventArgs > PointerReleased

  Raised when the pointer is released after a PointerPressed event.

6.52.1 Detailed Description

Represents a light-weight rendering action.

Definition at line 1186 of file AvaloniaContext.cs.

6.52.2 Member Enumeration Documentation

6.52.2.1 ActionTypes

`enum VectSharp.Canvas.RenderAction.ActionTypes [strong]`

Types of rendering actions.

Enumerator

Path	The render action represents a path object.
Text	The render action represents a text object.
RasterImage	The render action represents a raster image.

Definition at line 1191 of file AvaloniaContext.cs.

6.52.3 Member Function Documentation

6.52.3.1 BringToFront()

```
void VectSharp.Canvas.RenderAction.BringToFront ( )
```

Brings the render action to the front of the rendering queue. This method can only be invoked after the output has been fully initialised.

Definition at line 1412 of file AvaloniaContext.cs.

6.52.3.2 ImageAction()

```
static RenderAction VectSharp.Canvas.RenderAction.ImageAction (
            string imageId,
            Avalonia.Rect sourceRect,
            Avalonia.Rect destinationRect,
            Avalonia.Matrix transform,
            Geometry clippingPath,
            string tag = null )  [static]
```

Creates a new RenderAction representing an image.

Parameters

imageId	The univocal identifier of the image to draw.
sourceRect	The source rectangle of the image.
destinationRect	The destination rectangle of the image.
transform	The transform that will be applied to the image.
clippingPath	The clipping path.
tag	A tag to access the RenderAction. If this is null this RenderAction is not visible in the hit test.

Returns

A new RenderAction representing an image.

Definition at line 1395 of file AvaloniaContext.cs.

6.52.3.3 PathAction()

```
static RenderAction VectSharp.Canvas.RenderAction.PathAction (
            Geometry geometry,
            Pen stroke,
```

```
            IBrush fill,
            Avalonia.Matrix transform,
            Geometry clippingPath,
            string tag = null )  [static]
```

Creates a new RenderAction representing a path.

Parameters

geometry	The geometry to be rendered.
stroke	The stroke of the path (can be null).
fill	The fill of the path (can be null).
transform	The transform that will be applied to the path.
clippingPath	The clipping path.
tag	A tag to access the RenderAction. If this is null this RenderAction is not visible in the hit test.

Returns

A new RenderAction representing a path.

Definition at line 1348 of file AvaloniaContext.cs.

6.52.3.4 SendToBack()

```
void VectSharp.Canvas.RenderAction.SendToBack ( )
```

Brings the render action to the back of the rendering queue. This method can only be invoked after the output has been fully initialised.

Definition at line 1420 of file AvaloniaContext.cs.

6.52.3.5 TextAction()

```
static RenderAction VectSharp.Canvas.RenderAction.TextAction (
            Avalonia.Media.FormattedText text,
            IBrush fill,
            Avalonia.Matrix transform,
            Geometry clippingPath,
            string tag = null )  [static]
```

Creates a new RenderAction representing text.

Parameters

text	The text to be rendered.
fill	The fill of the text (can be null).
transform	The transform that will be applied to the text.
clippingPath	The clipping path.
tag	A tag to access the RenderAction. If this is null this RenderAction is not visible in the hit test.

Returns

A new RenderAction representing text.

Definition at line 1371 of file AvaloniaContext.cs.

6.52.4 Property Documentation

6.52.4.1 ActionType

```
ActionTypes VectSharp.Canvas.RenderAction.ActionType  [get]
```

Type of the rendering action.

Definition at line 1212 of file AvaloniaContext.cs.

6.52.4.2 ClippingPath

```
Geometry VectSharp.Canvas.RenderAction.ClippingPath  [get], [set]
```

The current clipping path.

Definition at line 1252 of file AvaloniaContext.cs.

6.52.4.3 Fill

```
IBrush VectSharp.Canvas.RenderAction.Fill  [get], [set]
```

Rendering fill (null if the rendered action only has a Stroke). If you change this, you need to invalidate the Parent's visual.

Definition at line 1232 of file AvaloniaContext.cs.

6.52.4.4 Geometry

```
Geometry VectSharp.Canvas.RenderAction.Geometry  [get], [set]
```

Geometry that needs to be rendered (null if the action type is ActionTypes.Text). If you change this, you need to invalidate the Parent's visual.

Definition at line 1217 of file AvaloniaContext.cs.

6.52.4.5 ImageDestination

```
Avalonia.?  Rect VectSharp.Canvas.RenderAction.ImageDestination  [get], [set]
```

The destination rectangle of the image.

Definition at line 1247 of file AvaloniaContext.cs.

6.52.4.6 ImageId

```
string VectSharp.Canvas.RenderAction.ImageId  [get], [set]
```

Univocal identifier of the image that needs to be drawn.

Definition at line 1237 of file AvaloniaContext.cs.

6.52.4.7 ImageSource

```
Avalonia.?  Rect VectSharp.Canvas.RenderAction.ImageSource  [get], [set]
```

The source rectangle of the image.

Definition at line 1242 of file AvaloniaContext.cs.

6.52.4.8 InverseTransform

```
Avalonia.Matrix VectSharp.Canvas.RenderAction.InverseTransform = Avalonia.Matrix.Identity
[get]
```

Inverse transformation matrix.

Definition at line 1259 of file AvaloniaContext.cs.

6.52.4.9 Parent

```
Avalonia.Controls.Canvas VectSharp.Canvas.RenderAction.Parent  [get]
```

The container of this RenderAction.

Definition at line 1284 of file AvaloniaContext.cs.

6.52.4.10 Stroke

```
Pen VectSharp.Canvas.RenderAction.Stroke  [get], [set]
```

Rendering stroke (null if the action type is ActionTypes.Text or if the rendered action only has a Fill). If you change this, you need to invalidate the Parent's visual.

Definition at line 1227 of file AvaloniaContext.cs.

6.52.4.11 Tag

```
string VectSharp.Canvas.RenderAction.Tag  [get], [set]
```

A tag to access the RenderAction.

Definition at line 1277 of file AvaloniaContext.cs.

6.52.4.12 Text

```
Avalonia.Media.FormattedText VectSharp.Canvas.RenderAction.Text  [get], [set]
```

Text that needs to be rendered (null if the action type is ActionTypes.Path). If you change this, you need to invalidate the Parent's visual.

Definition at line 1222 of file AvaloniaContext.cs.

6.52.4.13 Transform

```
Avalonia.Matrix VectSharp.Canvas.RenderAction.Transform  [get], [set]
```

Rendering transformation matrix. If you change this, you need to invalidate the Parent's visual.

Definition at line 1264 of file AvaloniaContext.cs.

6.52.5 Event Documentation

6.52.5.1 PointerEnter

```
EventHandler<Avalonia.Input.PointerEventArgs> VectSharp.Canvas.RenderAction.PointerEnter
```

Raised when the pointer enters the area covered by the RenderAction.

Definition at line 1295 of file AvaloniaContext.cs.

6.52.5.2 PointerLeave

```
EventHandler<Avalonia.Input.PointerEventArgs> VectSharp.Canvas.RenderAction.PointerLeave
```

Raised when the pointer leaves the area covered by the RenderAction.

Definition at line 1300 of file AvaloniaContext.cs.

6.52.5.3 PointerPressed

```
EventHandler<Avalonia.Input.PointerPressedEventArgs> VectSharp.Canvas.RenderAction.Pointer↩
Pressed
```

Raised when the pointer is pressed while over the area covered by the RenderAction.

Definition at line 1305 of file AvaloniaContext.cs.

6.52.5.4 PointerReleased

```
EventHandler<Avalonia.Input.PointerReleasedEventArgs> VectSharp.Canvas.RenderAction.Pointer↩
Released
```

Raised when the pointer is released after a PointerPressed event.

Definition at line 1310 of file AvaloniaContext.cs.

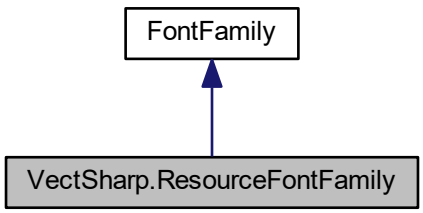
The documentation for this class was generated from the following file:

- VectSharp.Canvas/AvaloniaContext.cs

6.53 VectSharp.ResourceFontFamily Class Reference

Represents a FontFamily created from a resource stream.

Inheritance diagram for VectSharp.ResourceFontFamily:



Public Member Functions

- ResourceFontFamily (System.IO.Stream resourceStream, string resourceName)

  Create a new ResourceFontFamily from the specified resourceStream containing a TTF file, passing the specified resourceName to the

Public Attributes

- string ResourceName

  The name of the embedded resource, which will be parsed using

Additional Inherited Members

6.53.1 Detailed Description

Represents a FontFamily created from a resource stream.

Definition at line 526 of file Font.cs.

6.53.2 Constructor & Destructor Documentation

6.53.2.1 ResourceFontFamily()

```
VectSharp.ResourceFontFamily.ResourceFontFamily (
            System.IO.Stream resourceStream,
            string resourceName )
```

Create a new ResourceFontFamily from the specified *resourceStream* containing a TTF file, passing the specified *resourceName* to the

`Avalonia.Media.FontFamily.Parse(string, Uri)` " method.

Parameters

resourceStream	A resource stream containing a TTF file.
resourceName	The name of the embedded resource, which will be parsed using `Avalonia.Media.FontFamily.Parse(string, Uri)` .

Definition at line 538 of file Font.cs.

6.53.3 Member Data Documentation

6.53.3.1 ResourceName

```
string VectSharp.ResourceFontFamily.ResourceName
```

The name of the embedded resource, which will be parsed using

`Avalonia.Media.FontFamily.Parse(string, Uri)`.

Definition at line 531 of file Font.cs.

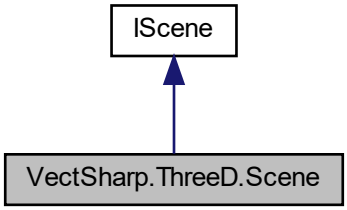
The documentation for this class was generated from the following file:

- VectSharp/Font.cs

6.54 VectSharp.ThreeD.Scene Class Reference

Represents a 3D scene.

Inheritance diagram for VectSharp.ThreeD.Scene:



Public Member Functions

- Scene ()
  Creates a new Scene.
- void AddElement (Element3D element)
  Adds the specified element to the scene.
- void AddRange (IEnumerable< Element3D > elements)
  Adds the specified elements to the scene.
- void Replace (Func< Element3D, Element3D > replacementFunction)
  Replaces each element in the scene with the element returned by the replacementFunction .
- void Replace (Func< Element3D, IEnumerable< Element3D >> replacementFunction)
  Replaces each element in the scene with the element(s) returned by the replacementFunction .

Public Attributes

- IEnumerable< Element3D > SceneElements => sceneElements

Properties

- object SceneLock `[get]`

6.54.1 Detailed Description

Represents a 3D scene.

Definition at line 66 of file Scene.cs.

6.54.2 Constructor & Destructor Documentation

6.54.2.1 Scene()

`VectSharp.ThreeD.Scene.Scene ( )`

Creates a new Scene.

Definition at line 79 of file Scene.cs.

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

- VectSharp.ThreeD/Scene.cs

6.55 VectSharp.Segment Class Reference

Represents a segment as part of a GraphicsPath.

Public Member Functions

- abstract Segment Clone ()
  Creates a copy of the Segment.
- abstract double Measure (Point previousPoint)
  Computes the length of the Segment.
- abstract Point GetPointAt (Point previousPoint, double position)
  Gets the point on the Segment at the specified (relative) position).
- abstract Point GetTangentAt (Point previousPoint, double position)
  Gets the tangent to the Segment at the specified (relative) position).
- abstract IEnumerable< Segment > Linearise (Point? previousPoint, double resolution)
  Transform the segment into a series of linear segments. Segments that are already linear are not changed.
- abstract IEnumerable< Point > GetLinearisationTangents (Point? previousPoint, double resolution)
  Gets the tanget at the points at which the segment would be linearised.
- abstract IEnumerable< Segment > Transform (Func< Point, Point > transformationFunction)
  Applies an arbitrary transformation to all of the points of the Segment.

Properties

- abstract SegmentType Type [get]

  The type of the Segment.
- Point[] Points [get]

  The points used to define the Segment.
- virtual Point Point [get]

  The end point of the Segment.

6.55.1 Detailed Description

Represents a segment as part of a GraphicsPath.

Definition at line 28 of file Segment.cs.

6.55.2 Member Function Documentation

6.55.2.1 Clone()

```
abstract Segment VectSharp.Segment.Clone ( )  [pure virtual]
```

Creates a copy of the Segment.

Returns

A copy of the Segment.

6.55.2.2 GetLinearisationTangents()

```
abstract IEnumerable<Point> VectSharp.Segment.GetLinearisationTangents (
            Point?  previousPoint,
            double resolution )  [pure virtual]
```

Gets the tanget at the points at which the segment would be linearised.

Parameters

previousPoint	The point from which the Segment starts (i.e. the endpoint of the previous Segment).
resolution	The absolute length between successive samples in curve segments.

Returns

A collection of tangents at the points in which the segment would be linearised.

6.55.2.3 GetPointAt()

```
abstract Point VectSharp.Segment.GetPointAt (
            Point previousPoint,
            double position )  [pure virtual]
```

Gets the point on the Segment at the specified (relative) *position*).

Parameters

previousPoint	The point from which the Segment starts (i.e. the endpoint of the previous Segment).
position	The relative position on the Segment (0 is the start of the Segment, 1 is the end of the Segment).

Returns

The point at the specified position.

6.55.2.4 GetTangentAt()

```
abstract Point VectSharp.Segment.GetTangentAt (
            Point previousPoint,
            double position )  [pure virtual]
```

Gets the tangent to the Segment at the specified (relative) *position*).

Parameters

previousPoint	The point from which the Segment starts (i.e. the endpoint of the previous Segment).
position	The relative position on the Segment (0 is the start of the Segment, 1 is the end of the Segment).

Returns

The tangent to the point at the specified position.

6.55.2.5 Linearise()

```
abstract IEnumerable<Segment> VectSharp.Segment.Linearise (
            Point?  previousPoint,
            double resolution )  [pure virtual]
```

Transform the segment into a series of linear segments. Segments that are already linear are not changed.

Parameters

previousPoint	The point from which the Segment starts (i.e. the endpoint of the previous Segment).
resolution	The absolute length between successive samples in curve segments.

Returns

A collection of linear segments that approximate the current segment.

6.55.2.6 Measure()

```
abstract double VectSharp.Segment.Measure (
            Point previousPoint )  [pure virtual]
```

Computes the length of the Segment.

Parameters

previousPoint	The point from which the Segment starts (i.e. the endpoint of the previous Segment).

Returns

The length of the segment.

6.55.2.7 Transform()

```
abstract IEnumerable<Segment> VectSharp.Segment.Transform (
            Func< Point, Point > transformationFunction )  [pure virtual]
```

Applies an arbitrary transformation to all of the points of the Segment.

Parameters

transformationFunction	An arbitrary transformation function.

Returns

A collection of Segments that have been transformed according to the *transformationFunction* .

6.55.3 Property Documentation

6.55.3.1 Point

```
virtual Point VectSharp.Segment.Point  [get]
```

The end point of the Segment.

Definition at line 44 of file Segment.cs.

6.55.3.2 Points

```
Point [] VectSharp.Segment.Points  [get]
```

The points used to define the Segment.

Definition at line 39 of file Segment.cs.

6.55.3.3 Type

```
abstract SegmentType VectSharp.Segment.Type  [get]
```

The type of the Segment.

Definition at line 34 of file Segment.cs.

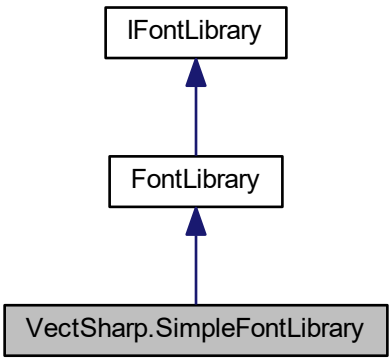
The documentation for this class was generated from the following file:

- VectSharp/Segment.cs

6.56 VectSharp.SimpleFontLibrary Class Reference

A font library that can be used to cache and resolve font family names.

Inheritance diagram for VectSharp.SimpleFontLibrary:



Public Member Functions

- SimpleFontLibrary (IFontLibrary standardFontLibrary)

  Create a new SimpleFontLibrary instance.
- SimpleFontLibrary ()

  Create a new SimpleFontLibrary instance, using the default font library to resolve the standard font families.
- SimpleFontLibrary (FontFamily timesRoman, FontFamily timesBold, FontFamily timesItalic, FontFamily timesBoldItalic, FontFamily helvetica, FontFamily helveticaBold, FontFamily helveticaOblique, FontFamily helveticaBoldOblique, FontFamily courier, FontFamily courierBold, FontFamily courierOblique, FontFamily courierBoldOblique, FontFamily symbol, FontFamily zapfdingbats)

  Create a new SimpleFontLibrary instance, with the specified replacements for the standard font families.
- SimpleFontLibrary (string timesRoman, string timesBold, string timesItalic, string timesBoldItalic, string helvetica, string helveticaBold, string helveticaOblique, string helveticaBoldOblique, string courier, string courierBold, string courierOblique, string courierBoldOblique, string symbol, string zapfdingbats)

  Create a new SimpleFontLibrary instance, with the specified replacements for the standard font families.
- void Add (string fontFamilyName, FontFamily fontFamily)

  Add the specified font family to the library with the specified name.
- void Add (FontFamily fontFamily)

  Add the specified font family to the library.
- void Add (string fileName)

  Add the font family contained in the specified True Type Font file to the library.
- void Add (string fontFamily, string fileName)

  Add the font family contained in the specified True Type Font file to the library, with the specified name. The font family is not loaded until it is requested for the first time.
- override FontFamily ResolveFontFamily (FontFamily.StandardFontFamilies standardFontFamily)

  Create a new font family from the specified standard font family name.
- override FontFamily ResolveFontFamily (string fontFamily)

  Create a new font family from the specified family name or true type file. If the family name or the true type file are not valid, an exception might be raised.

6.56.1 Detailed Description

A font library that can be used to cache and resolve font family names.

Definition at line 186 of file FontLibrary.cs.

6.56.2 Constructor & Destructor Documentation

6.56.2.1 SimpleFontLibrary() [1/4]

```
VectSharp.SimpleFontLibrary.SimpleFontLibrary (
            IFontLibrary standardFontLibrary )
```

Create a new SimpleFontLibrary instance.

Parameters

standardFontLibrary	An existing font library that will be used to resolve the standard font families.

Definition at line 198 of file FontLibrary.cs.

6.56.2.2 SimpleFontLibrary() [2/4]

```
VectSharp.SimpleFontLibrary.SimpleFontLibrary ( )
```

Create a new SimpleFontLibrary instance, using the default font library to resolve the standard font families.

Definition at line 216 of file FontLibrary.cs.

6.56.2.3 SimpleFontLibrary() [3/4]

```
VectSharp.SimpleFontLibrary.SimpleFontLibrary (
              FontFamily timesRoman,
              FontFamily timesBold,
              FontFamily timesItalic,
              FontFamily timesBoldItalic,
              FontFamily helvetica,
              FontFamily helveticaBold,
              FontFamily helveticaOblique,
              FontFamily helveticaBoldOblique,
              FontFamily courier,
              FontFamily courierBold,
              FontFamily courierOblique,
              FontFamily courierBoldOblique,
              FontFamily symbol,
              FontFamily zapfdingbats )
```

Create a new SimpleFontLibrary instance, with the specified replacements for the standard font families.

Parameters

timesRoman	The font family to use for the Times-Roman standard font.
timesBold	The font family to use for the Times-Bold standard font.
timesItalic	The font family to use for the Times-Italic standard font.
timesBoldItalic	The font family to use for the Times-BoldItalic standard font.
helvetica	The font family to use for the Helvetica standard font.
helveticaBold	The font family to use for the Helvetica-Bold standard font.
helveticaOblique	The font family to use for the Helvetica-Oblique standard font.
helveticaBoldOblique	The font family to use for the Helvetica-BoldOblique standard font.
courier	The font family to use for the Courier standard font.
courierBold	The font family to use for the Courier-Bold standard font.
courierOblique	The font family to use for the Courier-Oblique standard font.
courierBoldOblique	The font family to use for the Courier-BoldOblique standard font.
symbol	The font family to use for the Symbol standard font.
zapfdingbats	The font family to use for the Zapfdingbats standard font.

Definition at line 238 of file FontLibrary.cs.

6.56.2.4 SimpleFontLibrary() [4/4]

```
VectSharp.SimpleFontLibrary.SimpleFontLibrary (
            string timesRoman,
            string timesBold,
            string timesItalic,
            string timesBoldItalic,
            string helvetica,
            string helveticaBold,
            string helveticaOblique,
            string helveticaBoldOblique,
            string courier,
            string courierBold,
            string courierOblique,
            string courierBoldOblique,
            string symbol,
            string zapfdingbats )
```

Create a new SimpleFontLibrary instance, with the specified replacements for the standard font families.

Parameters

timesRoman	The font family to use for the Times-Roman standard font.
timesBold	The font family to use for the Times-Bold standard font.
timesItalic	The font family to use for the Times-Italic standard font.
timesBoldItalic	The font family to use for the Times-BoldItalic standard font.
helvetica	The font family to use for the Helvetica standard font.
helveticaBold	The font family to use for the Helvetica-Bold standard font.
helveticaOblique	The font family to use for the Helvetica-Oblique standard font.
helveticaBoldOblique	The font family to use for the Helvetica-BoldOblique standard font.
courier	The font family to use for the Courier standard font.
courierBold	The font family to use for the Courier-Bold standard font.
courierOblique	The font family to use for the Courier-Oblique standard font.
courierBoldOblique	The font family to use for the Courier-BoldOblique standard font.
symbol	The font family to use for the Symbol standard font.
zapfdingbats	The font family to use for the Zapfdingbats standard font.

Definition at line 293 of file FontLibrary.cs.

6.56.3 Member Function Documentation

6.56.3.1 Add() [1/4]

```
void VectSharp.SimpleFontLibrary.Add (
            FontFamily fontFamily )
```

Add the specified font family to the library.

Parameters

fontFamily	The font family to add.

Definition at line 352 of file FontLibrary.cs.

6.56.3.2 Add() [2/4]

```
void VectSharp.SimpleFontLibrary.Add (
            string fileName )
```

Add the font family contained in the specified True Type Font file to the library.

Parameters

fileName	The path to the TTF file containing the font family.

Definition at line 368 of file FontLibrary.cs.

6.56.3.3 Add() [3/4]

```
void VectSharp.SimpleFontLibrary.Add (
            string fontFamily,
            string fileName )
```

Add the font family contained in the specified True Type Font file to the library, with the specified name. The font family is not loaded until it is requested for the first time.

Parameters

fontFamily	The name of the font family.
fileName	The path to the TTF file containing the font family.

Definition at line 387 of file FontLibrary.cs.

6.56.3.4 Add() [4/4]

```
void VectSharp.SimpleFontLibrary.Add (
            string fontFamilyName,
            FontFamily fontFamily )
```

Add the specified font family to the library with the specified name.

Parameters

fontFamilyName	The name of the font family.
fontFamily	The font family to add.

Definition at line 336 of file FontLibrary.cs.

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

- VectSharp/FontLibrary.cs

6.57 VectSharp.Size Struct Reference

Represents the size of an object.

Public Member Functions

- Size (double width, double height)
  Create a new Size.

Public Attributes

- double Width
  Width of the object.
- double Height
  Height of the object.

6.57.1 Detailed Description

Represents the size of an object.

Definition at line 82 of file Point.cs.

6.57.2 Constructor & Destructor Documentation

6.57.2.1 Size()

```
VectSharp.Size.Size (
            double width,
            double height )
```

Create a new Size.

Parameters

width	The width of the object.
height	The height of the object.

Definition at line 99 of file Point.cs.

6.57.3 Member Data Documentation

6.57.3.1 Height

```
double VectSharp.Size.Height
```

Height of the object.

Definition at line 92 of file Point.cs.

6.57.3.2 Width

```
double VectSharp.Size.Width
```

Width of the object.

Definition at line 87 of file Point.cs.

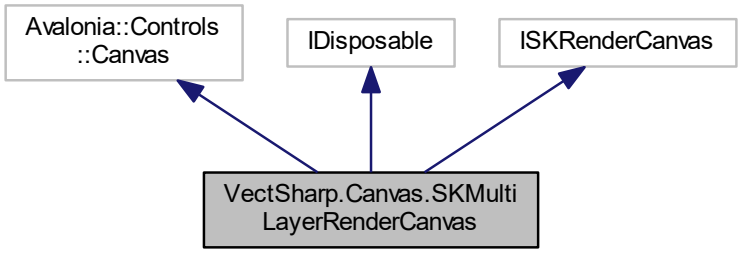
The documentation for this struct was generated from the following file:

- VectSharp/Point.cs

6.58 VectSharp.Canvas.SKMultiLayerRenderCanvas Class Reference

Represents a multi-threaded, triple-buffered canvas on which the image is drawn using SkiaSharp.

Inheritance diagram for VectSharp.Canvas.SKMultiLayerRenderCanvas:



Public Member Functions

- SKMultiLayerRenderCanvas (Document document, Colour backgroundColour, double width, double height, List< SKRenderAction > layerTransforms=null)

  Create a new SKMultiLayerRenderCanvas from a Document, where each page represents a layer.
- SKMultiLayerRenderCanvas (IEnumerable< Page > layers, Colour backgroundColour, double width, double height, List< SKRenderAction > layerTransforms=null)

  Create a new SKMultiLayerRenderCanvas from a collection of Pages, each representing a layer.
- SKMultiLayerRenderCanvas (List< SKRenderContext > contents, List< SKRenderAction > contentTransforms, Colour backgroundColour, double width, double height)

  Create a new SKMultiLayerRenderCanvas from a list of SKRenderContexts, each representing a layer.
- void UpdateWith (List< SKRenderContext > contents, List< SKRenderAction > contentTransforms, Colour backgroundColour, double width, double height)

  Replace the contents of the SKMultiLayerRenderCanvas with the specified layers.
- void UpdateLayer (int layer, SKRenderContext newContent, SKRenderAction newTransform)

  Replace a single layer with the specified content.
- void AddLayer (SKRenderContext newContent, SKRenderAction newTransform)

  Add a new layer to the image.
- void InsertLayer (int index, SKRenderContext newContent, SKRenderAction newTransform)

  Insert a new layer at the specified index.
- void RemoveLayer (int layer)

  Remove the specified layer from the image.
- void SwitchLayers (int layer1, int layer2)

  Switch the position of the two specified layers.
- void MoveLayer (int oldIndex, int newIndex)

  Move the specified layer to the specified position, shifting all other layers as necessary.
- RenderTargetBitmap RenderAtResolution (int width, int height, SKColor? background=null)

  Render the image at to a bitmap at the specified resolution.
- void InvalidateDirty ()

  Invalidate the contents of the canvas, forcing it to redraw itself.
- void InvalidateZIndex ()

  Invalidate the contents of the canvas, specifying that the order of the layers has changed.
- override void Render (DrawingContext context)
- void Dispose ()

Public Attributes

- List< List< SKRenderAction > > RenderActions

  The list of render actions, each element in this list is itself a list, containing the actions that correspond to a layer in the image.
- List< SKRenderAction > LayerTransforms

  The list of transforms associated with each layer.
- object RenderLock = new object()

  An lock for the rendering loop. The public methods of this class already lock on this, but you may need it if you want to directly manipulate the contents of the canvas.

Properties

- double PageWidth [get, set]

  The width of the page that is rendered on this canvas.
- double PageHeight [get, set]

  The height of the page that is rendered on this canvas.
- Func< long, Bitmap > Spinner = null [get, set]

  If the image to draw is not already cached, this method is called with an argument containing the number of milliseconds since the image was last rendered. The method can return a Bitmap that will be drawn on the canvas in order to let users know that the image is being rendered in background.

6.58.1 Detailed Description

Represents a multi-threaded, triple-buffered canvas on which the image is drawn using SkiaSharp.

Definition at line 43 of file SKMultiLayerRenderCanvas.cs.

6.58.2 Constructor & Destructor Documentation

6.58.2.1 SKMultiLayerRenderCanvas() [1/3]

```
VectSharp.Canvas.SKMultiLayerRenderCanvas.SKMultiLayerRenderCanvas (
            Document document,
            Colour backgroundColour,
            double width,
            double height,
            List< SKRenderAction > layerTransforms = null )
```

Create a new SKMultiLayerRenderCanvas from a Document, where each page represents a layer.

Parameters

document	The document containing the layers as Pages.
layerTransforms	A list of transforms associated with each layer. This list should contain the same number of elements as the number of pages in *document* . This is useful to manipulate the position of each layer individually. If this is null, an identity transform is applied to each layer.
backgroundColour	The background colour of the canvas.
width	The width of the canvas and the pages it contains.
height	The height of the canvas and the pages it contains.

Definition at line 98 of file SKMultiLayerRenderCanvas.cs.

6.58.2.2 SKMultiLayerRenderCanvas() [2/3]

```
VectSharp.Canvas.SKMultiLayerRenderCanvas.SKMultiLayerRenderCanvas (
            IEnumerable< Page > layers,
            Colour backgroundColour,
            double width,
            double height,
            List< SKRenderAction > layerTransforms = null )
```

Create a new SKMultiLayerRenderCanvas from a collection of Pages, each representing a layer.

Parameters

layers	The contents of the canvas. Each element in this list represents a layer.
layerTransforms	A list of transforms associated with each layer. This list should contain the same number of elements as *layers* . This is useful to manipulate the position of each layer individually. If this is null, an identity transform is applied to each layer.
backgroundColour	The background colour of the canvas.
width	The width of the canvas and the pages it contains.
height	The height of the canvas and the pages it contains.

Definition at line 108 of file SKMultiLayerRenderCanvas.cs.

6.58.2.3 SKMultiLayerRenderCanvas() [3/3]

```
VectSharp.Canvas.SKMultiLayerRenderCanvas.SKMultiLayerRenderCanvas (
            List< SKRenderContext > contents,
            List< SKRenderAction > contentTransforms,
            Colour backgroundColour,
            double width,
            double height )
```

Create a new SKMultiLayerRenderCanvas from a list of SKRenderContexts, each representing a layer.

Parameters

contents	The contents of the canvas. Each element in this list represents a layer. A Page can be converded to a SKRenderContext through the CopyToSKRenderContext method.
contentTransforms	A list of transforms associated with each layer. This list should contain the same number of elements as *contents* . This is useful to manipulate the position of each layer individually.
backgroundColour	The background colour of the canvas.
width	The width of the canvas and the page it contains.
height	The height of the canvas and the page it contains.

Definition at line 143 of file SKMultiLayerRenderCanvas.cs.

6.58.3 Member Function Documentation

6.58.3.1 AddLayer()

```
void VectSharp.Canvas.SKMultiLayerRenderCanvas.AddLayer (
            SKRenderContext newContent,
            SKRenderAction newTransform )
```

Add a new layer to the image.

Parameters

newContent	The contents of the new layer. A Page can be converded to a SKRenderContext through the CopyToSKRenderContext method.
newTransform	The transform for the new layer.

Definition at line 276 of file SKMultiLayerRenderCanvas.cs.

6.58.3.2 InsertLayer()

```
void VectSharp.Canvas.SKMultiLayerRenderCanvas.InsertLayer (
            int index,
            SKRenderContext newContent,
            SKRenderAction newTransform )
```

Insert a new layer at the specified index.

Parameters

index	The position at which the new layer will be inserted.
newContent	The contents of the new layer.
newTransform	The transform for the new layer.

Definition at line 309 of file SKMultiLayerRenderCanvas.cs.

6.58.3.3 InvalidateDirty()

```
void VectSharp.Canvas.SKMultiLayerRenderCanvas.InvalidateDirty ( )
```

Invalidate the contents of the canvas, forcing it to redraw itself.

Definition at line 992 of file SKMultiLayerRenderCanvas.cs.

6.58.3.4 InvalidateZIndex()

```
void VectSharp.Canvas.SKMultiLayerRenderCanvas.InvalidateZIndex ( )
```

Invalidate the contents of the canvas, specifying that the order of the layers has changed.

Definition at line 1001 of file SKMultiLayerRenderCanvas.cs.

6.58.3.5 MoveLayer()

```
void VectSharp.Canvas.SKMultiLayerRenderCanvas.MoveLayer (
            int oldIndex,
            int newIndex )
```

Move the specified layer to the specified position, shifting all other layers as necessary.

Parameters

oldIndex	The current index of the layer to move.
newIndex	The final index of the layer. Layers after this will be shifted by 1 in order to accommodate the moved layer.

Definition at line 389 of file SKMultiLayerRenderCanvas.cs.

6.58.3.6 RemoveLayer()

```
void VectSharp.Canvas.SKMultiLayerRenderCanvas.RemoveLayer (
            int layer )
```

Remove the specified layer from the image.

Parameters

layer	The index of the layer to remove.

Definition at line 340 of file SKMultiLayerRenderCanvas.cs.

6.58.3.7 RenderAtResolution()

```
RenderTargetBitmap VectSharp.Canvas.SKMultiLayerRenderCanvas.RenderAtResolution (
            int width,
            int height,
            SKColor?  background = null )
```

Render the image at to a bitmap at the specified resolution.

Parameters

width	The width of the rendered image. Note that the actual width of the returned image might be lower than this, depending on the aspect ratio of the image.
height	The height of the rendered image. Note that the actual height of the returned image might be lower than this, depending on the aspect ratio of the image.
background	The background colour for the image. If this is `null`, the current background colour is used.

Returns

A RenderTargetBitmap containing the image rendered at the specified resolution.

Definition at line 735 of file SKMultiLayerRenderCanvas.cs.

6.58.3.8 SwitchLayers()

```
void VectSharp.Canvas.SKMultiLayerRenderCanvas.SwitchLayers (
            int layer1,
            int layer2 )
```

Switch the position of the two specified layers.

Parameters

layer1	The index of the first layer to switch.
layer2	The index of the second layer to switch.

Definition at line 364 of file SKMultiLayerRenderCanvas.cs.

6.58.3.9 UpdateLayer()

```
void VectSharp.Canvas.SKMultiLayerRenderCanvas.UpdateLayer (
            int layer,
            SKRenderContext newContent,
            SKRenderAction newTransform )
```

Replace a single layer with the specified content.

Parameters

layer	The index of the layer to replace.
newContent	The new contents of the layer. A Page can be converded to a SKRenderContext through the CopyToSKRenderContext method.
newTransform	The new transform for the layer.

Definition at line 244 of file SKMultiLayerRenderCanvas.cs.

6.58.3.10 UpdateWith()

```
void VectSharp.Canvas.SKMultiLayerRenderCanvas.UpdateWith (
            List< SKRenderContext > contents,
            List< SKRenderAction > contentTransforms,
            Colour backgroundColour,
            double width,
            double height )
```

Replace the contents of the SKMultiLayerRenderCanvas with the specified layers.

Parameters

contents	The contents of the canvas. Each element in this list represents a layer. A Page can be converded to a SKRenderContext through the CopyToSKRenderContext method.
contentTransforms	A list of transforms associated with each layer. This list should contain the same number of elements as *contents* . This is useful to manipulate the position of each layer individually.
backgroundColour	The background colour of the canvas.
width	The width of the canvas and the page it contains.
height	The height of the canvas and the page it contains.

Definition at line 166 of file SKMultiLayerRenderCanvas.cs.

6.58.4 Member Data Documentation

6.58.4.1 LayerTransforms

```
List<SKRenderAction> VectSharp.Canvas.SKMultiLayerRenderCanvas.LayerTransforms
```

The list of transforms associated with each layer.

Definition at line 64 of file SKMultiLayerRenderCanvas.cs.

6.58.4.2 RenderActions

```
List<List<SKRenderAction> > VectSharp.Canvas.SKMultiLayerRenderCanvas.RenderActions
```

The list of render actions, each element in this list is itself a list, containing the actions that correspond to a layer in the image.

Definition at line 59 of file SKMultiLayerRenderCanvas.cs.

6.58.4.3 RenderLock

```
object VectSharp.Canvas.SKMultiLayerRenderCanvas.RenderLock = new object()
```

An lock for the rendering loop. The public methods of this class already lock on this, but you may need it if you want to directly manipulate the contents of the canvas.

Definition at line 726 of file SKMultiLayerRenderCanvas.cs.

6.58.5 Property Documentation

6.58.5.1 PageHeight

```
double VectSharp.Canvas.SKMultiLayerRenderCanvas.PageHeight  [get], [set]
```

The height of the page that is rendered on this canvas.

Definition at line 53 of file SKMultiLayerRenderCanvas.cs.

6.58.5.2 PageWidth

```
double VectSharp.Canvas.SKMultiLayerRenderCanvas.PageWidth  [get], [set]
```

The width of the page that is rendered on this canvas.

Definition at line 48 of file SKMultiLayerRenderCanvas.cs.

6.58.5.3 Spinner

```
Func<long, Bitmap> VectSharp.Canvas.SKMultiLayerRenderCanvas.Spinner = null  [get], [set]
```

If the image to draw is not already cached, this method is called with an argument containing the number of milliseconds since the image was last rendered. The method can return a Bitmap that will be drawn on the canvas in order to let users know that the image is being rendered in background.

Definition at line 70 of file SKMultiLayerRenderCanvas.cs.

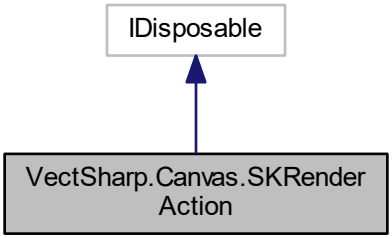
The documentation for this class was generated from the following file:

- VectSharp.Canvas/SKMultiLayerRenderCanvas.cs

6.59 VectSharp.Canvas.SKRenderAction Class Reference

Represents a light-weight rendering action.

Inheritance diagram for VectSharp.Canvas.SKRenderAction:



Public Types

- enum ActionTypes {
  ActionTypes.Path, ActionTypes.Text, ActionTypes.RasterImage, ActionTypes.Transform,
  ActionTypes.Save, ActionTypes.Restore, ActionTypes.Clip }

  Types of rendering actions.

Public Member Functions

- void InvalidateHitTestPath ()
- void InvalidateVisual ()
- void InvalidateZIndex ()
- void InvalidateAll ()
- void Dispose ()

Static Public Member Functions

- static SKRenderAction PathAction (SKPath path, SKPaint paint, string tag=null)

  Creates a new SKRenderAction representing a path.
- static SKRenderAction ClipAction (SKPath clippingPath, string tag=null)

  Creates a new SKRenderAction representing a clipping action.
- static SKRenderAction TextAction (string text, float x, float y, SKFont font, SKPaint paint, string tag=null)

  Creates a new SKRenderAction representing text.
- static SKRenderAction ImageAction (string imageId, SKRect sourceRect, SKRect destinationRect, string tag=null)

  Creates a new SKRenderAction representing an image.
- static SKRenderAction TransformAction (SKMatrix transform, string tag=null)

  Creates a new SKRenderAction representing a transform.
- static SKRenderAction SaveAction (string tag=null)

  Creates a new SKRenderAction that saves the current graphics state.
- static SKRenderAction RestoreAction (string tag=null)

  Creates a new SKRenderAction that saves the current graphics state.

Public Attributes

- bool Disposed => disposedValue

  Returns a boolean value indicating whether the current instance has been disposed.

Properties

- ActionTypes ActionType `[get]`

  Type of the rendering action.
- SKPath Path `[get, set]`

  Path that needs to be rendered (null if the action type is not ActionTypes.Path). If you change this, you probably want to call this object's InvalidateHitTestPath method.
- string Text `[get, set]`

  Text that needs to be rendered (null if the action type is not ActionTypes.Text). If you change this, you probably want to call this object's InvalidateHitTestPath method.
- SKFont Font `[get, set]`

  The font that will be used to render the text (null if the action type is not ActionTypes.Text). If you change this, you probably want to call this object's InvalidateHitTestPath method.
- float TextX `[get, set]`

  The X coordainate at which the text will be drawn (null if the action type is not ActionTypes.Text). If you change this, you probably want to call this object's InvalidateHitTestPath method.
- float TextY `[get, set]`

  The Y coordainate at which the text will be drawn (null if the action type is not ActionTypes.Text). If you change this, you probably want to call this object's InvalidateHitTestPath method.
- SKPaint Paint `[get, set]`

  Paint used to render the text or path (`null` if the action type is neither ActionTypes.Text nor ActionTypes.Path). If you change this, you probably want to call this object's InvalidateHitTestPath method.
- string ImageId `[get, set]`

  Univocal identifier of the image that needs to be drawn.
- SKRect? ImageSource `[get, set]`

  The source rectangle of the image (`null` if the action type is not ActionTypes.RasterImage). If you change this, you probably want to call this object's InvalidateVisual method.
- SKRect? ImageDestination `[get, set]`

  The destination rectangle of the image (`null` if the action type is not ActionTypes.RasterImage). If you change this, you probably want to call this object's InvalidateHitTestPath method.
- SKMatrix? Transform = null `[get, set]`

  The transformation matrix that will be applied to the current coordinate system (`null` if the action type is not ActionTypes.Transform). If you change this, you probably want to call this object's InvalidateVisual method.
- string Tag `[get]`

  A tag to access the SKRenderAction.
- uint ZIndex = 0 `[get, set]`

  The Z-index of the rendering action (an action with a higher Z-index will always appear above an action with a lower Z-index). The more different values there are for the Z-index, the slower the rendering, so keep use of this property to a minimum. If you change this, you probably want to call this object's InvalidateZIndex method.
- object Payload `[get, set]`

  An arbitrary object associated with the RenderAction.
- Avalonia.Controls.Canvas Parent `[get]`

  The container of this SKRenderAction.

Events

- EventHandler< Avalonia.Input.PointerEventArgs > PointerEnter

  Raised when the pointer enters the area covered by the SKRenderAction.
- EventHandler< Avalonia.Input.PointerEventArgs > PointerLeave

  Raised when the pointer leaves the area covered by the SKRenderAction.
- EventHandler< Avalonia.Input.PointerPressedEventArgs > PointerPressed

  Raised when the pointer is pressed while over the area covered by the SKRenderAction.
- EventHandler< Avalonia.Input.PointerReleasedEventArgs > PointerReleased

  Raised when the pointer is released after a PointerPressed event.

6.59.1 Detailed Description

Represents a light-weight rendering action.

Definition at line 29 of file SKRenderContext.cs.

6.59.2 Member Enumeration Documentation

6.59.2.1 ActionTypes

```
enum VectSharp.Canvas.SKRenderAction.ActionTypes  [strong]
```

Types of rendering actions.

Enumerator

Path	The render action represents a path object.
Text	The render action represents a text object.
RasterImage	The render action represents a raster image.
Transform	The render action represents a transformation of the coordinate space.
Save	The render action represents saving the current graphics state.
Restore	The render action represents restoring the last saved graphics state.
Clip	The render action represents an update of the current clip path.

Definition at line 41 of file SKRenderContext.cs.

6.59.3 Member Function Documentation

6.59.3.1 ClipAction()

```
static SKRenderAction VectSharp.Canvas.SKRenderAction.ClipAction (
            SKPath clippingPath,
            string tag = null )  [static]
```

Creates a new SKRenderAction representing a clipping action.

Parameters

clippingPath	The path to be used for clipping.
tag	A tag to access the SKRenderAction.

Returns

A new SKRenderAction representing a clipping action.

Definition at line 329 of file SKRenderContext.cs.

6.59.3.2 ImageAction()

```
static SKRenderAction VectSharp.Canvas.SKRenderAction.ImageAction (
            string imageId,
            SKRect sourceRect,
            SKRect destinationRect,
            string tag = null )  [static]
```

Creates a new SKRenderAction representing an image.

Parameters

imageId	The univocal identifier of the image to draw.
sourceRect	The source rectangle of the image.
destinationRect	The destination rectangle of the image.
tag	A tag to access the SKRenderAction. If this is null this SKRenderAction is not visible in the hit test.

Returns

A new SKRenderAction representing an image.

Definition at line 381 of file SKRenderContext.cs.

6.59.3.3 InvalidateAll()

```
void VectSharp.Canvas.SKRenderAction.InvalidateAll ( )
```

This methods signals to the Parent that the Z-index, shape and visual properties (e.g. the colour) of this object have changed and triggers a redraw.

If you make changes to more than one SKRenderAction contained in the same Canvas, you only need to invalidate the last one.

This method should only be called after the output has been fully initialized.

Definition at line 287 of file SKRenderContext.cs.

6.59.3.4 InvalidateHitTestPath()

```
void VectSharp.Canvas.SKRenderAction.InvalidateHitTestPath ( )
```

Signals to this object that its shape has changed a new path needs to be computed for the purpose of hit-testing. Also signals to the Parent that the visual properties of this object have changed and triggers a redraw.

This method should be called whenever the "shape" of the object represented by the SKRenderAction changes. If only the visual properties of this object have changed (e.g. the colour), call the InvalidateVisual method instead.

If you make changes to more than one SKRenderAction contained in the same Canvas, you only need to invalidate the last one.

This method should only be called after the output has been fully initialized.

Definition at line 216 of file SKRenderContext.cs.

6.59.3.5 InvalidateVisual()

```
void VectSharp.Canvas.SKRenderAction.InvalidateVisual ( )
```

This methods signals to the Parent that the visual properties (e.g. the colour) of this object have changed and triggers a redraw.

If the "shape" of the object has changed as well, call the InvalidateHitTestPath method instead. If the Z-index of the object has changed, call the InvalidateZIndex method instead. If both the "shape" and the Z-index of the object have changed, call the InvalidateAll method.

If you make changes to more than one SKRenderAction contained in the same Canvas, you only need to invalidate the last one.

This method should only be called after the output has been fully initialized.

Definition at line 266 of file SKRenderContext.cs.

6.59.3.6 InvalidateZIndex()

```
void VectSharp.Canvas.SKRenderAction.InvalidateZIndex ( )
```

This methods signals to the Parent that the Z-index and visual properties (e.g. the colour) of this object have changed and triggers a redraw.

If the "shape" of the object has changed as well, call the InvalidateAll method instead.

If you make changes to more than one SKRenderAction contained in the same Canvas, you only need to invalidate the last one.

This method should only be called after the output has been fully initialized.

Definition at line 277 of file SKRenderContext.cs.

6.59.3.7 PathAction()

```
static SKRenderAction VectSharp.Canvas.SKRenderAction.PathAction (
            SKPath path,
            SKPaint paint,
            string tag = null )  [static]
```

Creates a new SKRenderAction representing a path.

Parameters

path	The geometry to be rendered.
paint	The paint used to fill or stroke the path.
tag	A tag to access the SKRenderAction. If this is null this SKRenderAction is not visible in the hit test.

Returns

A new SKRenderAction representing a path.

Definition at line 305 of file SKRenderContext.cs.

6.59.3.8 RestoreAction()

```
static SKRenderAction VectSharp.Canvas.SKRenderAction.RestoreAction (
            string tag = null )  [static]
```

Creates a new SKRenderAction that saves the current graphics state.

Parameters

tag	A tag to access the SKRenderAction.

Returns

A new SKRenderAction that restores the last saved graphics state.

Definition at line 435 of file SKRenderContext.cs.

6.59.3.9 SaveAction()

```
static SKRenderAction VectSharp.Canvas.SKRenderAction.SaveAction (
            string tag = null )  [static]
```

Creates a new SKRenderAction that saves the current graphics state.

Parameters

tag	A tag to access the SKRenderAction.

Returns

A new SKRenderAction that saves the current graphics state.

Definition at line 421 of file SKRenderContext.cs.

6.59.3.10 TextAction()

```
static SKRenderAction VectSharp.Canvas.SKRenderAction.TextAction (
            string text,
            float x,
            float y,
            SKFont font,
            SKPaint paint,
            string tag = null )  [static]
```

Creates a new SKRenderAction representing text.

Parameters

text	The text to be rendered.
x	The X coordinate at which the text will be drawn.
y	The Y coordinate at which the text will be drawn.
font	The font to be used to render the text.
paint	The paint to be used to fill or stroke the text.
tag	A tag to access the SKRenderAction. If this is null this SKRenderAction is not visible in the hit test.

Returns

A new SKRenderAction representing text.

Definition at line 349 of file SKRenderContext.cs.

6.59.3.11 TransformAction()

```
static SKRenderAction VectSharp.Canvas.SKRenderAction.TransformAction (
            SKMatrix transform,
            string tag = null )  [static]
```

Creates a new SKRenderAction representing a transform.

Parameters

transform	The transform to apply.
tag	A tag to access the SKRenderAction.

Returns

A new SKRenderAction representing a transform.

Definition at line 406 of file SKRenderContext.cs.

6.59.4 Member Data Documentation

6.59.4.1 Disposed

```
bool VectSharp.Canvas.SKRenderAction.Disposed => disposedValue
```

Returns a boolean value indicating whether the current instance has been disposed.

Definition at line 34 of file SKRenderContext.cs.

6.59.5 Property Documentation

6.59.5.1 ActionType

```
ActionTypes VectSharp.Canvas.SKRenderAction.ActionType  [get]
```

Type of the rendering action.

Definition at line 82 of file SKRenderContext.cs.

6.59.5.2 Font

```
SKFont VectSharp.Canvas.SKRenderAction.Font  [get], [set]
```

The font that will be used to render the text (null if the action type is not ActionTypes.Text). If you change this, you probably want to call this object's InvalidateHitTestPath method.

Definition at line 100 of file SKRenderContext.cs.

6.59.5.3 ImageDestination

```
SKRect?  VectSharp.Canvas.SKRenderAction.ImageDestination  [get], [set]
```

The destination rectangle of the image (`null` if the action type is not ActionTypes.RasterImage). If you change this, you probably want to call this object's InvalidateHitTestPath method.

Definition at line 130 of file SKRenderContext.cs.

6.59.5.4 ImageId

```
string VectSharp.Canvas.SKRenderAction.ImageId  [get], [set]
```

Univocal identifier of the image that needs to be drawn.

Definition at line 120 of file SKRenderContext.cs.

6.59.5.5 ImageSource

```
SKRect?  VectSharp.Canvas.SKRenderAction.ImageSource  [get], [set]
```

The source rectangle of the image (`null` if the action type is not ActionTypes.RasterImage). If you change this, you probably want to call this object's InvalidateVisual method.

Definition at line 125 of file SKRenderContext.cs.

6.59.5.6 Paint

```
SKPaint VectSharp.Canvas.SKRenderAction.Paint  [get], [set]
```

Paint used to render the text or path (`null` if the action type is neither ActionTypes.Text nor ActionTypes.Path). If you change this, you probably want to call this object's InvalidateHitTestPath method.

Definition at line 115 of file SKRenderContext.cs.

6.59.5.7 Parent

```
Avalonia.Controls.Canvas VectSharp.Canvas.SKRenderAction.Parent  [get]
```

The container of this SKRenderAction.

Definition at line 159 of file SKRenderContext.cs.

6.59.5.8 Path

```
SKPath VectSharp.Canvas.SKRenderAction.Path  [get], [set]
```

Path that needs to be rendered (null if the action type is not ActionTypes.Path). If you change this, you probably want to call this object's InvalidateHitTestPath method.

Definition at line 87 of file SKRenderContext.cs.

6.59.5.9 Payload

```
object VectSharp.Canvas.SKRenderAction.Payload  [get], [set]
```

An arbitrary object associated with the RenderAction.

Definition at line 152 of file SKRenderContext.cs.

6.59.5.10 Tag

```
string VectSharp.Canvas.SKRenderAction.Tag  [get]
```

A tag to access the SKRenderAction.

Definition at line 140 of file SKRenderContext.cs.

6.59.5.11 Text

```
string VectSharp.Canvas.SKRenderAction.Text  [get], [set]
```

Text that needs to be rendered (null if the action type is not ActionTypes.Text). If you change this, you probably want to call this object's InvalidateHitTestPath method.

Definition at line 95 of file SKRenderContext.cs.

6.59.5.12 TextX

```
float VectSharp.Canvas.SKRenderAction.TextX  [get], [set]
```

The X coordainate at which the text will be drawn (null if the action type is not ActionTypes.Text). If you change this, you probably want to call this object's InvalidateHitTestPath method.

Definition at line 105 of file SKRenderContext.cs.

6.59.5.13 TextY

```
float VectSharp.Canvas.SKRenderAction.TextY  [get], [set]
```

The Y coordainate at which the text will be drawn (null if the action type is not ActionTypes.Text). If you change this, you probably want to call this object's InvalidateHitTestPath method.

Definition at line 110 of file SKRenderContext.cs.

6.59.5.14 Transform

```
SKMatrix?  VectSharp.Canvas.SKRenderAction.Transform = null  [get], [set]
```

The transformation matrix that will be applied to the current coordinate system (`null` if the action type is not ActionTypes.Transform). If you change this, you probably want to call this object's InvalidateVisual method.

Definition at line 135 of file SKRenderContext.cs.

6.59.5.15 ZIndex

```
uint VectSharp.Canvas.SKRenderAction.ZIndex = 0  [get], [set]
```

The Z-index of the rendering action (an action with a higher Z-index will always appear above an action with a lower Z-index). The more different values there are for the Z-index, the slower the rendering, so keep use of this property to a minimum. If you change this, you probably want to call this object's InvalidateZIndex method.

Definition at line 147 of file SKRenderContext.cs.

6.59.6 Event Documentation

6.59.6.1 PointerEnter

```
EventHandler<Avalonia.Input.PointerEventArgs> VectSharp.Canvas.SKRenderAction.PointerEnter
```

Raised when the pointer enters the area covered by the SKRenderAction.

Definition at line 170 of file SKRenderContext.cs.

6.59.6.2 PointerLeave

```
EventHandler<Avalonia.Input.PointerEventArgs> VectSharp.Canvas.SKRenderAction.PointerLeave
```

Raised when the pointer leaves the area covered by the SKRenderAction.

Definition at line 175 of file SKRenderContext.cs.

6.59.6.3 PointerPressed

```
EventHandler<Avalonia.Input.PointerPressedEventArgs> VectSharp.Canvas.SKRenderAction.Pointer↩
Pressed
```

Raised when the pointer is pressed while over the area covered by the SKRenderAction.

Definition at line 180 of file SKRenderContext.cs.

6.59.6.4 PointerReleased

```
EventHandler<Avalonia.Input.PointerReleasedEventArgs> VectSharp.Canvas.SKRenderAction.Pointer↩
Released
```

Raised when the pointer is released after a PointerPressed event.

Definition at line 185 of file SKRenderContext.cs.

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

- VectSharp.Canvas/SKRenderContext.cs

6.60 VectSharp.Canvas.SKRenderContext Class Reference

Represents a page that has been prepared for fast rendering using the SkiaSharp renderer.

6.60.1 Detailed Description

Represents a page that has been prepared for fast rendering using the SkiaSharp renderer.

Definition at line 515 of file SKRenderContext.cs.

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

- VectSharp.Canvas/SKRenderContext.cs

6.61 VectSharp.Canvas.SKRenderContextInterpreter Class Reference

Contains methods to render a Page to an Avalonia.Controls.Canvas using the SkiaSharp renderer.

Static Public Member Functions

- static SKMultiLayerRenderCanvas PaintToSKCanvas (this Document document, double? width=null, double? height=null, Colour? background=null, AvaloniaContextInterpreter.TextOptions textOption=AvaloniaContextInterpreter.TextOptio

  Render a Document to an Avalonia.Controls.Canvas using the SkiaSharp renderer. Each page corresponds to a layer in the image.
- static SKMultiLayerRenderCanvas PaintToSKCanvas (this Document document, Dictionary< string, Func< SKRenderAction, IEnumerable< SKRenderAction >>> taggedActions, bool removeTagged↩ ActionsAfterExecution=true, double? width=null, double? height=null, Colour? background=null, AvaloniaContextInterpreter.TextOptions textOption=AvaloniaContextInterpreter.TextOptions.ConvertIfNecessary)

  Render a Document to an Avalonia.Controls.Canvas using the SkiaSharp renderer. Each page corresponds to a layer in the image.
- static SKMultiLayerRenderCanvas PaintToSKCanvas (this Document document, Dictionary< string, Func< SKRenderAction, IEnumerable< SKRenderAction >>> taggedActions, Dictionary< string,(SKBitmap, bool)> images, bool removeTaggedActionsAfterExecution=true, double? width=null, double? height=null, Colour? background=null, AvaloniaContextInterpreter.TextOptions textOption=AvaloniaContextInterpreter.TextOptions.ConvertI

  Render a Document to an Avalonia.Controls.Canvas using the SkiaSharp renderer. Each page corresponds to a layer in the image.
- static SKMultiLayerRenderCanvas PaintToSKCanvas (this Page page, AvaloniaContextInterpreter.TextOptions textOption=AvaloniaContextInterpreter.TextOptions.ConvertIfNecessary)

  Render a Page to an Avalonia.Controls.Canvas using the SkiaSharp renderer.
- static SKMultiLayerRenderCanvas PaintToSKCanvas (this Page page, Dictionary< string, Func< SKRenderAction, IEnumerable< SKRenderAction >>> taggedActions, bool removeTaggedActionsAfter↩ Execution=true, AvaloniaContextInterpreter.TextOptions textOption=AvaloniaContextInterpreter.TextOptions.ConvertIfNecessar

  Render a Page to an Avalonia.Controls.Canvas using the SkiaSharpRenderer.
- static SKMultiLayerRenderCanvas PaintToSKCanvas (this Page page, Dictionary< string, Func< SKRenderAction, IEnumerable< SKRenderAction >>> taggedActions, Dictionary< string,(SKBitmap, bool)> images, bool removeTaggedActionsAfterExecution=true, AvaloniaContextInterpreter.TextOptions textOption=AvaloniaContextInterpreter.TextOptions.ConvertIfNecessary)

  Render a Page to an Avalonia.Controls.Canvas using the SkiaSharpRenderer.
- static SKRenderContext CopyToSKRenderContext (this Page page, AvaloniaContextInterpreter.TextOptions textOption=AvaloniaContextInterpreter.TextOptions.ConvertIfNecessary)

  Render a Page to a SKRenderContext. This can be drawn using the SkiaSharpRenderer by adding it to a SKMultiLayerRenderCanvas.
- static SKRenderContext CopyToSKRenderContext (this Page page, Dictionary< string, Func< SKRenderAction, IEnumerable< SKRenderAction >>> taggedActions, bool removeTaggedActionsAfter↩ Execution=true, AvaloniaContextInterpreter.TextOptions textOption=AvaloniaContextInterpreter.TextOptions.ConvertIfNecessar

Render a Page to a SKRenderContext. This can be drawn using the SkiaSharpRenderer by adding it to a SKMultiLayerRenderCanvas.

- static SKRenderContext CopyToSKRenderContext (this Page page, Dictionary< string, Func< SKRenderAction, IEnumerable< SKRenderAction >>> taggedActions, Dictionary< string,(SKBitmap, bool)> images, bool removeTaggedActionsAfterExecution=true, AvaloniaContextInterpreter.TextOptions textOption=AvaloniaContextInterpreter.TextOptions.ConvertIfNecessary)

  Render a Page to a SKRenderContext. This can be drawn using the SkiaSharpRenderer by adding it to a SKMultiLayerRenderCanvas.

6.61.1 Detailed Description

Contains methods to render a Page to an Avalonia.Controls.Canvas using the SkiaSharp renderer.

Definition at line 1121 of file SKRenderContext.cs.

6.61.2 Member Function Documentation

6.61.2.1 CopyToSKRenderContext() [1/3]

```
static SKRenderContext VectSharp.Canvas.SKRenderContextInterpreter.CopyToSKRenderContext (
            this Page page,
            AvaloniaContextInterpreter.TextOptions textOption = AvaloniaContextInterpreter.TextOptions.Conver
) [static]
```

Render a Page to a SKRenderContext. This can be drawn using the SkiaSharpRenderer by adding it to a SKMultiLayerRenderCanvas.

Parameters

page	The Page to render.
textOption	Defines whether text items should be converted into paths when drawing.

Returns

A SKRenderContext containing the rendered graphics objects.

Definition at line 1235 of file SKRenderContext.cs.

6.61.2.2 CopyToSKRenderContext() [2/3]

```
static SKRenderContext VectSharp.Canvas.SKRenderContextInterpreter.CopyToSKRenderContext (
            this Page page,
            Dictionary< string, Func< SKRenderAction, IEnumerable< SKRenderAction >>> tagged↩
Actions,
```

```
            bool removeTaggedActionsAfterExecution = true,
            AvaloniaContextInterpreter.TextOptions textOption = AvaloniaContextInterpreter.TextOptions.Conver
)  [static]
```

Render a Page to a SKRenderContext. This can be drawn using the SkiaSharpRenderer by adding it to a SKMultiLayerRenderCanvas.

Parameters

page	The Page to render.
taggedActions	A Dictionary containing the actions that will be performed on items with the corresponding tag. These should be functions that accept one parameter of type SKRenderAction and return an IEnumerable<SKRenderAction> of the render actions that will actually be added to the plot.
removeTaggedActionsAfterExecution	Whether the actions should be removed from *taggedActions* after their execution. Set to false if the same action should be performed on multiple items with the same tag.
textOption	Defines whether text items should be converted into paths when drawing.

Returns

A SKRenderContext containing the rendered graphics objects.

Definition at line 1249 of file SKRenderContext.cs.

6.61.2.3 CopyToSKRenderContext() [3/3]

```
static SKRenderContext VectSharp.Canvas.SKRenderContextInterpreter.CopyToSKRenderContext (
            this Page page,
            Dictionary< string, Func< SKRenderAction, IEnumerable< SKRenderAction >>> taggedActions,
            Dictionary< string,(SKBitmap, bool)> images,
            bool removeTaggedActionsAfterExecution = true,
            AvaloniaContextInterpreter.TextOptions textOption = AvaloniaContextInterpreter.TextOptions.Conver
)  [static]
```

Render a Page to a SKRenderContext. This can be drawn using the SkiaSharpRenderer by adding it to a SKMultiLayerRenderCanvas.

Parameters

page	The Page to render.
taggedActions	A Dictionary containing the actions that will be performed on items with the corresponding tag. These should be functions that accept one parameter of type SKRenderAction and return an IEnumerable<SKRenderAction> of the render actions that will actually be added to the plot.

Parameters

images	A dictionary that associates to each raster image path (or data URL) the image rendered as a SKBitmap and a boolean value indicating whether it should be drawn as "pixelated" or not. This will be populated automatically as the page is rendered. If you are rendering multiple Pages (or you are rendering the same page multiple times), it will be beneficial to keep a reference to this dictionary and pass it again on further rendering requests; otherwise, you can just pass an empty dictionary.
removeTaggedActionsAfterExecution	Whether the actions should be removed from *taggedActions* after their execution. Set to false if the same action should be performed on multiple items with the same tag.
textOption	Defines whether text items should be converted into paths when drawing.

Returns

A SKRenderContext containing the rendered graphics objects.

Definition at line 1265 of file SKRenderContext.cs.

6.61.2.4 PaintToSKCanvas() [1/6]

```
static SKMultiLayerRenderCanvas VectSharp.Canvas.SKRenderContextInterpreter.PaintToSKCanvas (
            this Document document,
            Dictionary< string, Func< SKRenderAction, IEnumerable< SKRenderAction >>> taggedActions,
            bool removeTaggedActionsAfterExecution = true,
            double?  width = null,
            double?  height = null,
            Colour?  background = null,
            AvaloniaContextInterpreter.TextOptions textOption = AvaloniaContextInterpreter.TextOptions.Conver
)  [static]
```

Render a Document to an Avalonia.Controls.Canvas using the SkiaSharp renderer. Each page corresponds to a layer in the image.

Parameters

document	The Document to render.
taggedActions	A Dictionary containing the actions that will be performed on items with the corresponding tag. These should be functions that accept one parameter of type SKRenderAction and return an IEnumerable<SKRenderAction> of the render actions that will actually be added to the plot.
removeTaggedActionsAfterExecution	Whether the actions should be removed from *taggedActions* after their execution. Set to false if the same action should be performed on multiple items with the same tag.
width	The width of the document. If this is `null`, the width of the largest page is used.

Parameters

height	The height of the document. If this is `null`, the height of the largest page is used.
background	The background colour of the document. If this is `null`, a transparent background is used.
textOption	Defines whether text items should be converted into paths when drawing.

Returns

An Avalonia.Controls.Canvas containing the rendered graphics objects.

Definition at line 1164 of file SKRenderContext.cs.

6.61.2.5 PaintToSKCanvas() [2/6]

```
static SKMultiLayerRenderCanvas VectSharp.Canvas.SKRenderContextInterpreter.PaintToSKCanvas (
            this Document document,
            Dictionary< string, Func< SKRenderAction, IEnumerable< SKRenderAction >>> taggedActions,
            Dictionary< string,(SKBitmap, bool)> images,
            bool removeTaggedActionsAfterExecution = true,
            double?  width = null,
            double?  height = null,
            Colour?  background = null,
            AvaloniaContextInterpreter.TextOptions textOption = AvaloniaContextInterpreter.TextOptions.Conver
)  [static]
```

Render a Document to an Avalonia.Controls.Canvas using the SkiaSharp renderer. Each page corresponds to a layer in the image.

Parameters

document	The Document to render.
taggedActions	A Dictionary containing the actions that will be performed on items with the corresponding tag. These should be functions that accept one parameter of type SKRenderAction and return an IEnumerable<SKRenderAction> of the render actions that will actually be added to the plot.
images	A dictionary that associates to each raster image path (or data URL) the image rendered as a SKBitmap and a boolean value indicating whether it should be drawn as "pixelated" or not. This will be populated automatically as the page is rendered. If you are rendering multiple Pages (or you are rendering the same page multiple times), it will be beneficial to keep a reference to this dictionary and pass it again on further rendering requests; otherwise, you can just pass an empty dictionary.
removeTaggedActionsAfterExecution	Whether the actions should be removed from *taggedActions* after their execution. Set to false if the same action should be performed on multiple items with the same tag.

Parameters

width	The width of the document. If this is `null`, the width of the largest page is used.
height	The height of the document. If this is `null`, the height of the largest page is used.
background	The background colour of the document. If this is `null`, a transparent background is used.
textOption	Defines whether text items should be converted into paths when drawing.

Returns

An Avalonia.Controls.Canvas containing the rendered graphics objects.

Definition at line 1183 of file SKRenderContext.cs.

6.61.2.6 PaintToSKCanvas() [3/6]

```
static SKMultiLayerRenderCanvas VectSharp.Canvas.SKRenderContextInterpreter.PaintToSKCanvas (
            this Document document,
            double?  width = null,
            double?  height = null,
            Colour?  background = null,
            AvaloniaContextInterpreter.TextOptions textOption = AvaloniaContextInterpreter.TextOptions.Conver
)  [static]
```

Render a Document to an Avalonia.Controls.Canvas using the SkiaSharp renderer. Each page corresponds to a layer in the image.

Parameters

document	The Document to render.
width	The width of the document. If this is `null`, the width of the largest page is used.
height	The height of the document. If this is `null`, the height of the largest page is used.
background	The background colour of the document. If this is `null`, a transparent background is used.
textOption	Defines whether text items should be converted into paths when drawing.

Returns

An Avalonia.Controls.Canvas containing the rendered graphics objects.

Definition at line 1147 of file SKRenderContext.cs.

6.61.2.7 PaintToSKCanvas() [4/6]

```
static SKMultiLayerRenderCanvas VectSharp.Canvas.SKRenderContextInterpreter.PaintToSKCanvas (
            this Page page,
            AvaloniaContextInterpreter.TextOptions textOption = AvaloniaContextInterpreter.TextOptions.Conver
)  [static]
```

Render a Page to an Avalonia.Controls.Canvas using the SkiaSharp renderer.

Parameters

page	The Page to render.
textOption	Defines whether text items should be converted into paths when drawing.

Returns

An Avalonia.Controls.Canvas containing the rendered graphics objects.

Definition at line 1194 of file SKRenderContext.cs.

6.61.2.8 PaintToSKCanvas() [5/6]

```
static SKMultiLayerRenderCanvas VectSharp.Canvas.SKRenderContextInterpreter.PaintToSKCanvas (
            this Page page,
            Dictionary< string, Func< SKRenderAction, IEnumerable< SKRenderAction >>> taggedActions,
            bool removeTaggedActionsAfterExecution = true,
            AvaloniaContextInterpreter.TextOptions textOption = AvaloniaContextInterpreter.TextOptions.Conver
)  [static]
```

Render a Page to an Avalonia.Controls.Canvas using the SkiaSharpRenderer.

Parameters

page	The Page to render.
taggedActions	A Dictionary containing the actions that will be performed on items with the corresponding tag. These should be functions that accept one parameter of type SKRenderAction and return an IEnumerable<SKRenderAction> of the render actions that will actually be added to the plot.
removeTaggedActionsAfterExecution	Whether the actions should be removed from *taggedActions* after their execution. Set to false if the same action should be performed on multiple items with the same tag.
textOption	Defines whether text items should be converted into paths when drawing.

Returns

An Avalonia.Controls.Canvas containing the rendered graphics objects.

Definition at line 1208 of file SKRenderContext.cs.

6.61.2.9 PaintToSKCanvas() [6/6]

```
static SKMultiLayerRenderCanvas VectSharp.Canvas.SKRenderContextInterpreter.PaintToSKCanvas (
            this Page page,
            Dictionary< string, Func< SKRenderAction, IEnumerable< SKRenderAction >>> taggedActions,
            Dictionary< string,(SKBitmap, bool)> images,
            bool removeTaggedActionsAfterExecution = true,
            AvaloniaContextInterpreter.TextOptions textOption = AvaloniaContextInterpreter.TextOptions.Conver
)  [static]
```

Render a Page to an Avalonia.Controls.Canvas using the SkiaSharpRenderer.

Parameters

page	The Page to render.
taggedActions	A Dictionary containing the actions that will be performed on items with the corresponding tag. These should be functions that accept one parameter of type SKRenderAction and return an IEnumerable<SKRenderAction> of the render actions that will actually be added to the plot.
images	A dictionary that associates to each raster image path (or data URL) the image rendered as a SKBitmap and a boolean value indicating whether it should be drawn as "pixelated" or not. This will be populated automatically as the page is rendered. If you are rendering multiple Pages (or you are rendering the same page multiple times), it will be beneficial to keep a reference to this dictionary and pass it again on further rendering requests; otherwise, you can just pass an empty dictionary.
removeTaggedActionsAfterExecution	Whether the actions should be removed from *taggedActions* after their execution. Set to false if the same action should be performed on multiple items with the same tag.
textOption	Defines whether text items should be converted into paths when drawing.

Returns

An Avalonia.Controls.Canvas containing the rendered graphics objects.

Definition at line 1224 of file SKRenderContext.cs.

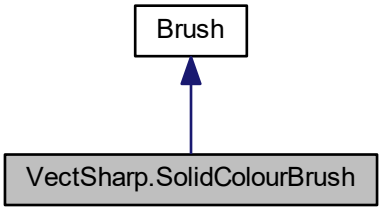
The documentation for this class was generated from the following file:

- VectSharp.Canvas/SKRenderContext.cs

6.62 VectSharp.SolidColourBrush Class Reference

Represents a brush painting with a single solid colour.

Inheritance diagram for VectSharp.SolidColourBrush:



Public Member Functions

- SolidColourBrush (Colour colour)

  Creates a new SolidColourBrush with the specified colour .
- override Brush MultiplyOpacity (double opacity)

  Returns a brush corresponding the current instance, with the specified opacity multiplication applied.

Static Public Member Functions

- static implicit operator SolidColourBrush (Colour colour)

  Implicitly converts a Colour into a SolidColourBrush.

Public Attributes

- double R => Colour.R

  Red component of the colour. Range: [0, 1].
- double G => Colour.G

  Green component of the colour. Range: [0, 1].
- double B => Colour.B

  Blue component of the colour. Range: [0, 1].
- double A => Colour.A

  Alpha component of the colour. Range: [0, 1].

Properties

- Colour Colour `[get]`

  The colour of the brush.

6.62.1 Detailed Description

Represents a brush painting with a single solid colour.

Definition at line 54 of file Brush.cs.

6.62.2 Constructor & Destructor Documentation

6.62.2.1 SolidColourBrush()

```
VectSharp.SolidColourBrush.SolidColourBrush (
            Colour colour )
```

Creates a new SolidColourBrush with the specified *colour* .

Parameters

colour	The Colour to use for the brush.

Definition at line 85 of file Brush.cs.

6.62.3 Member Function Documentation

6.62.3.1 operator SolidColourBrush()

```
static implicit VectSharp.SolidColourBrush.operator SolidColourBrush (
            Colour colour )  [static]
```

Implicitly converts a Colour into a SolidColourBrush.

Parameters

colour	The Colour to use for the brush.

Definition at line 100 of file Brush.cs.

6.62.4 Member Data Documentation

6.62.4.1 A

```
double VectSharp.SolidColourBrush.A => Colour.A
```

Alpha component of the colour. Range: [0, 1].

Definition at line 79 of file Brush.cs.

6.62.4.2 B

```
double VectSharp.SolidColourBrush.B => Colour.B
```

Blue component of the colour. Range: [0, 1].

Definition at line 74 of file Brush.cs.

6.62.4.3 G

```
double VectSharp.SolidColourBrush.G => Colour.G
```

Green component of the colour. Range: [0, 1].

Definition at line 69 of file Brush.cs.

6.62.4.4 R

```
double VectSharp.SolidColourBrush.R => Colour.R
```

Red component of the colour. Range: [0, 1].

Definition at line 64 of file Brush.cs.

6.62.5 Property Documentation

6.62.5.1 Colour

```
Colour VectSharp.SolidColourBrush.Colour  [get]
```

The colour of the brush.

Definition at line 59 of file Brush.cs.

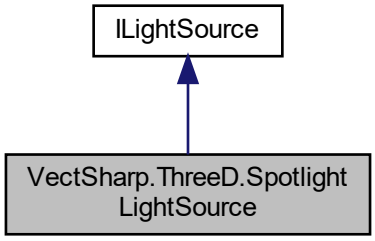
The documentation for this class was generated from the following file:

- VectSharp/Brush.cs

6.63 VectSharp.ThreeD.SpotlightLightSource Class Reference

Represents a conic spotlight.

Inheritance diagram for VectSharp.ThreeD.SpotlightLightSource:



Public Member Functions

- SpotlightLightSource (double intensity, Point3D position, NormalizedVector3D direction, double beamWidth↩ Angle, double cutoffAngle)

  Creates a new SpotlightLightSource instance.
- LightIntensity GetLightAt (Point3D point)

  Computes the light intensity at the specified point, without taking into account any obstructions.
- double GetObstruction (Point3D point, IEnumerable< Triangle3DElement > shadowingTriangles)

  Determines the amount of obstruction of the light that results at point due to the specified shadowingTriangles .

Properties

- bool CastsShadow = true `[get, set]`
- Point3D Position `[get, set]`

  The position of the light source.
- NormalizedVector3D Direction `[get, set]`

  The direction of the cone axis.
- double Intensity `[get, set]`

  The base intensity of the light.
- double BeamWidthAngle `[get, set]`

  The angular size of the light cone, in radians.
- double CutoffAngle `[get, set]`

  The angular size of the cutoff cone, in radians.
- double DistanceAttenuationExponent = 2 `[get, set]`

  An exponent determining how fast the light attenuates with increasing distance. Set to 0 to disable distance attenuation.
- double AngleAttenuationExponent = 1 `[get, set]`

  An exponent determining how fast the light attenuates between the main light cone and the cutoff cone.

6.63.1 Detailed Description

Represents a conic spotlight.

Definition at line 256 of file Lights.cs.

6.63.2 Constructor & Destructor Documentation

6.63.2.1 SpotlightLightSource()

```
VectSharp.ThreeD.SpotlightLightSource.SpotlightLightSource (
            double intensity,
            Point3D position,
            NormalizedVector3D direction,
            double beamWidthAngle,
            double cutoffAngle )
```

Creates a new SpotlightLightSource instance.

Parameters

intensity	The intensity of the light.
position	The position of the light source.
direction	The direction of the cone's axis.
beamWidthAngle	The angular size of the light cone, in radians.
cutoffAngle	The angular size of the cutoff cone, in radians.

Definition at line 304 of file Lights.cs.

6.63.3 Property Documentation

6.63.3.1 AngleAttenuationExponent

```
double VectSharp.ThreeD.SpotlightLightSource.AngleAttenuationExponent = 1  [get], [set]
```

An exponent determining how fast the light attenuates between the main light cone and the cutoff cone.

Definition at line 294 of file Lights.cs.

6.63.3.2 BeamWidthAngle

```
double VectSharp.ThreeD.SpotlightLightSource.BeamWidthAngle  [get], [set]
```

The angular size of the light cone, in radians.

Definition at line 279 of file Lights.cs.

6.63.3.3 CutoffAngle

```
double VectSharp.ThreeD.SpotlightLightSource.CutoffAngle  [get], [set]
```

The angular size of the cutoff cone, in radians.

Definition at line 284 of file Lights.cs.

6.63.3.4 Direction

```
NormalizedVector3D VectSharp.ThreeD.SpotlightLightSource.Direction  [get], [set]
```

The direction of the cone axis.

Definition at line 269 of file Lights.cs.

6.63.3.5 DistanceAttenuationExponent

```
double VectSharp.ThreeD.SpotlightLightSource.DistanceAttenuationExponent = 2  [get], [set]
```

An exponent determining how fast the light attenuates with increasing distance. Set to 0 to disable distance attenuation.

Definition at line 289 of file Lights.cs.

6.63.3.6 Intensity

```
double VectSharp.ThreeD.SpotlightLightSource.Intensity  [get], [set]
```

The base intensity of the light.

Definition at line 274 of file Lights.cs.

6.63.3.7 Position

```
Point3D VectSharp.ThreeD.SpotlightLightSource.Position  [get], [set]
```

The position of the light source.

Definition at line 264 of file Lights.cs.

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

- VectSharp.ThreeD/Lights.cs

6.64 VectSharp.SVG.SVGContextInterpreter Class Reference

Contains methods to render a Page as an SVG file.

Public Types

- enum TextOptions { TextOptions.EmbedFonts, TextOptions.SubsetFonts, TextOptions.ConvertIntoPaths, TextOptions.DoNotEmbed }

  Defines whether the used fonts should be included in the file.

Static Public Member Functions

- static void SaveAsSVG (this Page page, string fileName, TextOptions textOption=TextOptions.SubsetFonts, Dictionary< string, string > linkDestinations=null)

  Render the page to an SVG file.
- static void SaveAsSVG (this Page page, Stream stream, TextOptions textOption=TextOptions.SubsetFonts, Dictionary< string, string > linkDestinations=null)

  Render the page to an SVG stream.

6.64.1 Detailed Description

Contains methods to render a Page as an SVG file.

Definition at line 1105 of file SVGContext.cs.

6.64.2 Member Enumeration Documentation

6.64.2.1 TextOptions

```
enum VectSharp.SVG.SVGContextInterpreter.TextOptions  [strong]
```

Defines whether the used fonts should be included in the file.

Enumerator

EmbedFonts	Embeds the full font files.
SubsetFonts	Embeds subsetted font files containing only the glyphs for the characters that have been used.
ConvertIntoPaths	Does not embed any font file and converts all text items into paths.
DoNotEmbed	Does not embed any font file, but still encodes text items as such.

Definition at line 1126 of file SVGContext.cs.

6.64.3 Member Function Documentation

6.64.3.1 SaveAsSVG() [1/2]

```
static void VectSharp.SVG.SVGContextInterpreter.SaveAsSVG (
            this Page page,
            Stream stream,
            TextOptions textOption = TextOptions.SubsetFonts,
            Dictionary< string, string > linkDestinations = null )  [static]
```

Render the page to an SVG stream.

Parameters

page	The Page to render.
stream	The stream to which the SVG data will be written.
textOption	Defines whether the used fonts should be included in the file.
linkDestinations	A dictionary associating element tags to link targets. If this is provided, objects that have been drawn with a tag contained in the dictionary will become hyperlink to the destination specified in the dictionary. If the destination starts with a hash (#), it is interpreted as the tag of another object in the current document; otherwise, it is interpreted as an external URI.

Definition at line 1156 of file SVGContext.cs.

6.64.3.2 SaveAsSVG() [2/2]

```
static void VectSharp.SVG.SVGContextInterpreter.SaveAsSVG (
            this Page page,
            string fileName,
            TextOptions textOption = TextOptions.SubsetFonts,
            Dictionary< string, string > linkDestinations = null )  [static]
```

Render the page to an SVG file.

Parameters

page	The Page to render.
fileName	The full path to the file to save. If it exists, it will be overwritten.
textOption	Defines whether the used fonts should be included in the file.
linkDestinations	A dictionary associating element tags to link targets. If this is provided, objects that have been drawn with a tag contained in the dictionary will become hyperlink to the destination specified in the dictionary. If the destination starts with a hash (#), it is interpreted as the tag of another object in the current document; otherwise, it is interpreted as an external URI.

Definition at line 1115 of file SVGContext.cs.

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

- VectSharp.SVG/SVGContext.cs

6.65 VectSharp.Markdown.SyntaxHighlighter Class Reference

Contains methods to perform syntax highlighting.

Static Public Member Functions

- static List< List< FormattedString > > GetSyntaxHighlightedLines (string sourceCode, string language)
  Performs syntax highlighting for a specified language on some source code.

6.65.1 Detailed Description

Contains methods to perform syntax highlighting.

Definition at line 73 of file SyntaxHighlighting.cs.

6.65.2 Member Function Documentation

6.65.2.1 GetSyntaxHighlightedLines()

```
static List<List<FormattedString> > VectSharp.Markdown.SyntaxHighlighter.GetSyntaxHighlighted↩
Lines (
            string sourceCode,
            string language )  [static]
```

Performs syntax highlighting for a specified language on some source code.

Parameters

sourceCode	The source code to be highlighted.
language	The name of the language to use for the highlighting.

Returns

A list of lists of FormattedStrings. Each list of FormattedStrings represents a line.

Definition at line 129 of file SyntaxHighlighting.cs.

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

- VectSharp.Markdown/SyntaxHighlighting.cs

6.66 VectSharp.TrueTypeFile Class Reference

Represents a font file in TrueType format. Reference: http://stevehanov.ca/blog/?id=143, https://developer.apple.com/fonts/TrueType-Reference-Manual/, https://docs.microsoft.com/en-us/typography/opentype/spec/

Classes

- struct Bearings

  Represents the left- and right-side bearings of a glyph.
- struct TrueTypePoint

  Represents a point in a TrueType path description.
- struct VerticalMetrics

  Represents the maximum heigth above and depth below the baseline of a glyph.

Public Member Functions

- void Destroy ()

  Remove this TrueType file from the cache, clear the tables and release the FontStream. Only call this when the actual file that was used to create this object needs to be changed!
- TrueTypeFile SubsetFont (string charactersToInclude, bool consolidateAt32=false, Dictionary< char, char > outputEncoding=null)

  Create a subset of the TrueType file, containing only the glyphs for the specified characters.
- string GetFontFamilyName ()

  Obtains the font family name from the TrueType file.
- string GetFontName ()

  Obtains the PostScript font name from the TrueType file.
- ushort GetFirstCharIndex ()

  Returns the index of the first character glyph represented by the font.
- ushort GetLastCharIndex ()

  Returns the index of the last character glyph represented by the font.
- bool IsItalic ()

Determines whether the typeface is Italic or Oblique or not.

- bool IsOblique ()

  Determines whether the typeface is Oblique or not.
- bool IsBold ()

  Determines whether the typeface is Bold or not.
- bool IsFixedPitch ()

  Determines whether the typeface is fixed-pitch (aka monospaces) or not.
- bool IsSerif ()

  Determines whether the typeface is serifed or not.
- bool IsScript ()

  Determines whether the typeface is a script typeface or not.
- int GetGlyphIndex (char glyph)

  Determines the index of the glyph corresponding to a certain character.
- TrueTypePoint[ ][ ] GetGlyphPath (int glyphIndex, double size)

  Get the path that describes the shape of a glyph.
- TrueTypePoint[ ][ ] GetGlyphPath (char glyph, double size)

  Get the path that describes the shape of a glyph.
- double Get1000EmGlyphWidth (char glyph)

  Computes the advance width of a glyph, in thousandths of em unit.
- double Get1000EmGlyphWidth (int glyphIndex)

  Computes the advance width of a glyph, in thousandths of em unit.
- double Get1000EmWinAscent ()

  Computes the font's Win ascent, in thousandths of em unit.
- double Get1000EmAscent ()

  Computes the font ascent, in thousandths of em unit.
- double Get1000EmDescent ()

  Computes the font descent, in thousandths of em unit.
- double Get1000EmYMax ()

  Computes the maximum height over the baseline of the font, in thousandths of em unit.
- double Get1000EmYMin ()

  Computes the maximum depth below the baseline of the font, in thousandths of em unit.
- double Get1000EmXMax ()

  Computes the maximum distance to the right of the glyph origin of the font, in thousandths of em unit.
- double Get1000EmXMin ()

  Computes the maximum distance to the left of the glyph origin of the font, in thousandths of em unit.
- Bearings Get1000EmGlyphBearings (char glyph)

  Computes the left- and right- side bearings of a glyph, in thousandths of em unit.
- VerticalMetrics Get1000EmGlyphVerticalMetrics (char glyph)

  Computes the vertical metrics of a glyph, in thousandths of em unit.

Properties

- Stream FontStream `[get]`

  A stream pointing to the TrueType file source (either on disk or in memory). Never dispose this stream directly; if you really need to, call Destroy instead.

6.66.1 Detailed Description

Represents a font file in TrueType format. Reference: http://stevehanov.ca/blog/?id=143, https://developer.apple.com/fonts/TrueType-Reference-Manual/, https://docs.↩microsoft.com/en-us/typography/opentype/spec/

Definition at line 30 of file TrueType.cs.

6.66.2 Member Function Documentation

6.66.2.1 Destroy()

```
void VectSharp.TrueTypeFile.Destroy ( )
```

Remove this TrueType file from the cache, clear the tables and release the FontStream. Only call this when the actual file that was used to create this object needs to be changed!

Definition at line 53 of file TrueType.cs.

6.66.2.2 Get1000EmAscent()

```
double VectSharp.TrueTypeFile.Get1000EmAscent ( )
```

Computes the font ascent, in thousandths of em unit.

Returns

The font ascent in thousandths of em unit.

Definition at line 2098 of file TrueType.cs.

6.66.2.3 Get1000EmDescent()

```
double VectSharp.TrueTypeFile.Get1000EmDescent ( )
```

Computes the font descent, in thousandths of em unit.

Returns

The font descent in thousandths of em unit.

Definition at line 2108 of file TrueType.cs.

6.66.2.4 Get1000EmGlyphBearings()

```
Bearings VectSharp.TrueTypeFile.Get1000EmGlyphBearings (
            char glyph )
```

Computes the left- and right- side bearings of a glyph, in thousandths of em unit.

Parameters

glyph	The glyph whose bearings are to be computed.

Returns

The left- and right- side bearings of the glyph in thousandths of em unit

Definition at line 2190 of file TrueType.cs.

6.66.2.5 Get1000EmGlyphVerticalMetrics()

```
VerticalMetrics VectSharp.TrueTypeFile.Get1000EmGlyphVerticalMetrics (
            char glyph )
```

Computes the vertical metrics of a glyph, in thousandths of em unit.

Parameters

glyph	The glyph whose vertical metrics are to be computed.

Returns

The vertical metrics of a glyph, in thousandths of em unit.

Definition at line 2238 of file TrueType.cs.

6.66.2.6 Get1000EmGlyphWidth() [1/2]

```
double VectSharp.TrueTypeFile.Get1000EmGlyphWidth (
            char glyph )
```

Computes the advance width of a glyph, in thousandths of em unit.

Parameters

glyph	The glyph whose advance width is to be computed.

Returns

The advance width of the glyph in thousandths of em unit.

Definition at line 2049 of file TrueType.cs.

6.66.2.7 Get1000EmGlyphWidth() [2/2]

```
double VectSharp.TrueTypeFile.Get1000EmGlyphWidth (
            int glyphIndex )
```

Computes the advance width of a glyph, in thousandths of em unit.

Parameters

glyphIndex	The index of the glyph whose advance width is to be computed.

Returns

The advance width of the glyph in thousandths of em unit.

Definition at line 2067 of file TrueType.cs.

6.66.2.8 Get1000EmWinAscent()

```
double VectSharp.TrueTypeFile.Get1000EmWinAscent ( )
```

Computes the font's Win ascent, in thousandths of em unit.

Returns

The font's Win ascent in thousandths of em unit.

Definition at line 2078 of file TrueType.cs.

6.66.2.9 Get1000EmXMax()

```
double VectSharp.TrueTypeFile.Get1000EmXMax ( )
```

Computes the maximum distance to the right of the glyph origin of the font, in thousandths of em unit.

Returns

The maximum distance to the right of the glyph origin of the font in thousandths of em unit.

Definition at line 2135 of file TrueType.cs.

6.66.2.10 Get1000EmXMin()

```
double VectSharp.TrueTypeFile.Get1000EmXMin ( )
```

Computes the maximum distance to the left of the glyph origin of the font, in thousandths of em unit.

Returns

The maximum distance to the left of the glyph origin of the font in thousandths of em unit.

Definition at line 2144 of file TrueType.cs.

6.66.2.11 Get1000EmYMax()

```
double VectSharp.TrueTypeFile.Get1000EmYMax ( )
```

Computes the maximum height over the baseline of the font, in thousandths of em unit.

Returns

The maximum height over the baseline of the font in thousandths of em unit.

Definition at line 2117 of file TrueType.cs.

6.66.2.12 Get1000EmYMin()

```
double VectSharp.TrueTypeFile.Get1000EmYMin ( )
```

Computes the maximum depth below the baseline of the font, in thousandths of em unit.

Returns

The maximum depth below the baseline of the font in thousandths of em unit.

Definition at line 2126 of file TrueType.cs.

6.66.2.13 GetFirstCharIndex()

```
ushort VectSharp.TrueTypeFile.GetFirstCharIndex ( )
```

Returns the index of the first character glyph represented by the font.

Returns

The index of the first character glyph represented by the font.

Definition at line 1887 of file TrueType.cs.

6.66.2.14 GetFontFamilyName()

```
string VectSharp.TrueTypeFile.GetFontFamilyName ( )
```

Obtains the font family name from the TrueType file.

Returns

The font family name, if available; `null` otherwise.

Definition at line 1840 of file TrueType.cs.

6.66.2.15 GetFontName()

```
string VectSharp.TrueTypeFile.GetFontName ( )
```

Obtains the PostScript font name from the TrueType file.

Returns

The PostScript font name, if available; `null` otherwise.

Definition at line 1868 of file TrueType.cs.

6.66.2.16 GetGlyphIndex()

```
int VectSharp.TrueTypeFile.GetGlyphIndex (
            char glyph )
```

Determines the index of the glyph corresponding to a certain character.

Parameters

glyph	The character whose glyph is sought.

Returns

The index of the glyph in the TrueType file.

Definition at line 1977 of file TrueType.cs.

6.66.2.17 GetGlyphPath() [1/2]

```
TrueTypePoint [][] VectSharp.TrueTypeFile.GetGlyphPath (
            char glyph,
            double size )
```

Get the path that describes the shape of a glyph.

Parameters

glyph	The glyph whose path is sought.
size	The font size to be used for the font coordinates.

Returns

An array of contours, each of which is itself an array of TrueType points.

Definition at line 2039 of file TrueType.cs.

6.66.2.18 GetGlyphPath() [2/2]

```
TrueTypePoint [][] VectSharp.TrueTypeFile.GetGlyphPath (
            int glyphIndex,
            double size )
```

Get the path that describes the shape of a glyph.

Parameters

glyphIndex	The index of the glyph whose path is sought.
size	The font size to be used for the font coordinates.

Returns

An array of contours, each of which is itself an array of TrueType points.

Definition at line 2028 of file TrueType.cs.

6.66.2.19 GetLastCharIndex()

```
ushort VectSharp.TrueTypeFile.GetLastCharIndex ( )
```

Returns the index of the last character glyph represented by the font.

Returns

The index of the last character glyph represented by the font.

Definition at line 1898 of file TrueType.cs.

6.66.2.20 IsBold()

```
bool VectSharp.TrueTypeFile.IsBold ( )
```

Determines whether the typeface is Bold or not.

Returns

A bool indicating whether the typeface is Bold or not

Definition at line 1932 of file TrueType.cs.

6.66.2.21 IsFixedPitch()

```
bool VectSharp.TrueTypeFile.IsFixedPitch ( )
```

Determines whether the typeface is fixed-pitch (aka monospaces) or not.

Returns

A bool indicating whether the typeface is fixed-pitch (aka monospaces) or not.

Definition at line 1943 of file TrueType.cs.

6.66.2.22 IsItalic()

```
bool VectSharp.TrueTypeFile.IsItalic ( )
```

Determines whether the typeface is Italic or Oblique or not.

Returns

A bool indicating whether the typeface is Italic or Oblique or not.

Definition at line 1910 of file TrueType.cs.

6.66.2.23 IsOblique()

```
bool VectSharp.TrueTypeFile.IsOblique ( )
```

Determines whether the typeface is Oblique or not.

Returns

A bool indicating whether the typeface is Oblique or not.

Definition at line 1921 of file TrueType.cs.

6.66.2.24 IsScript()

```
bool VectSharp.TrueTypeFile.IsScript ( )
```

Determines whether the typeface is a script typeface or not.

Returns

A bool indicating whether the typeface is a script typeface or not.

Definition at line 1965 of file TrueType.cs.

6.66.2.25 IsSerif()

```
bool VectSharp.TrueTypeFile.IsSerif ( )
```

Determines whether the typeface is serifed or not.

Returns

A bool indicating whether the typeface is serifed or not.

Definition at line 1954 of file TrueType.cs.

6.66.2.26 SubsetFont()

```
TrueTypeFile VectSharp.TrueTypeFile.SubsetFont (
            string charactersToInclude,
            bool consolidateAt32 = false,
            Dictionary< char, char > outputEncoding = null )
```

Create a subset of the TrueType file, containing only the glyphs for the specified characters.

Parameters

charactersToInclude	A string containing the characters for which the glyphs should be included.
consolidateAt32	If true, the character map is rearranged so that the included glyphs start at the unicode U+0032 control point.
outputEncoding	If *consolidateAt32* is true, entries will be added to this dictionary mapping the original characters to the new map (that starts at U+0033).

Returns

Definition at line 556 of file TrueType.cs.

6.66.3 Property Documentation

6.66.3.1 FontStream

```
Stream VectSharp.TrueTypeFile.FontStream  [get]
```

A stream pointing to the TrueType file source (either on disk or in memory). Never dispose this stream directly; if you really need to, call Destroy instead.

Definition at line 47 of file TrueType.cs.

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

- VectSharp/TrueType.cs

6.67 VectSharp.TrueTypeFile.TrueTypePoint Struct Reference

Represents a point in a TrueType path description.

Public Attributes

- double X

  The horizontal coordinate of the point.
- double Y

  The vertical coordinate of the point.
- bool IsOnCurve

  Whether the point is a point on the curve, or a control point of a quadratic Bezier curve.

6.67.1 Detailed Description

Represents a point in a TrueType path description.

Definition at line 1354 of file TrueType.cs.

6.67.2 Member Data Documentation

6.67.2.1 IsOnCurve

```
bool VectSharp.TrueTypeFile.TrueTypePoint.IsOnCurve
```

Whether the point is a point on the curve, or a control point of a quadratic Bezier curve.

Definition at line 1369 of file TrueType.cs.

6.67.2.2 X

```
double VectSharp.TrueTypeFile.TrueTypePoint.X
```

The horizontal coordinate of the point.

Definition at line 1359 of file TrueType.cs.

6.67.2.3 Y

```
double VectSharp.TrueTypeFile.TrueTypePoint.Y
```

The vertical coordinate of the point.

Definition at line 1364 of file TrueType.cs.

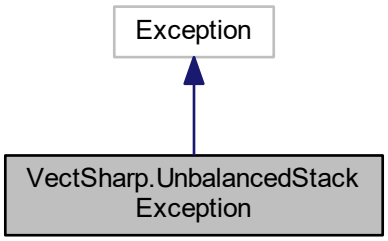
The documentation for this struct was generated from the following file:

- VectSharp/TrueType.cs

6.68 VectSharp.UnbalancedStackException Class Reference

The exception that is thrown when an unbalanced graphics state stack occurs.

Inheritance diagram for VectSharp.UnbalancedStackException:



6.68.1 Detailed Description

The exception that is thrown when an unbalanced graphics state stack occurs.

Definition at line 245 of file Graphics.cs.

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

- VectSharp/Graphics.cs

6.69 VectSharp.TrueTypeFile.VerticalMetrics Struct Reference

Represents the maximum heigth above and depth below the baseline of a glyph.

Public Attributes

- int YMin

  The maximum depth below the baseline of the glyph.
- int YMax

  The maximum height above the baseline of the glyph.

6.69.1 Detailed Description

Represents the maximum heigth above and depth below the baseline of a glyph.

Definition at line 2207 of file TrueType.cs.

6.69.2 Member Data Documentation

6.69.2.1 YMax

```
int VectSharp.TrueTypeFile.VerticalMetrics.YMax
```

The maximum height above the baseline of the glyph.

Definition at line 2217 of file TrueType.cs.

6.69.2.2 YMin

```
int VectSharp.TrueTypeFile.VerticalMetrics.YMin
```

The maximum depth below the baseline of the glyph.

Definition at line 2212 of file TrueType.cs.

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

- VectSharp/TrueType.cs

Index