

Farouq Adepetu's Rendering Engine

Generated by Doxygen 1.9.4



<b>1 Namespace Index</b>	<b>1</b>
1.1 Namespace List	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Namespace Documentation</b>	<b>7</b>
4.1 FACamera Namespace Reference	7
4.1.1 Detailed Description	7
4.2 FARender Namespace Reference	7
4.2.1 Detailed Description	8
4.3 FAWindow Namespace Reference	8
4.3.1 Detailed Description	8
<b>5 Class Documentation</b>	<b>9</b>
5.1 FACamera::Camera Class Reference	9
5.1.1 Detailed Description	11
5.1.2 Constructor & Destructor Documentation	11
5.1.2.1 Camera()	11
5.1.3 Member Function Documentation	11
5.1.3.1 Backward()	12
5.1.3.2 Down()	12
5.1.3.3 Foward()	12
5.1.3.4 GetAngularVelocity()	12
5.1.3.5 GetAspectRatio()	12
5.1.3.6 GetCameraPosition()	12
5.1.3.7 GetCameraVelocity()	13
5.1.3.8 GetPerspectiveProjectionMatrix()	13
5.1.3.9 GetVerticalFov()	13
5.1.3.10 GetViewPerspectiveProjectionMatrix()	13
5.1.3.11 GetViewTransformationMatrix()	13
5.1.3.12 GetX()	13
5.1.3.13 GetY()	14
5.1.3.14 GetZ()	14
5.1.3.15 GetZFar()	14
5.1.3.16 GetZNear()	14
5.1.3.17 KeyboardInput()	14
5.1.3.18 Left()	14
5.1.3.19 LookAt()	15
5.1.3.20 MouseInput()	15
5.1.3.21 Right()	15

5.1.3.22 RotateCameraLeftRight()	15
5.1.3.23 RotateCameraUpDown()	15
5.1.3.24 SetAngularVelocity()	16
5.1.3.25 SetAspectRatio()	16
5.1.3.26 SetCameraPosition()	16
5.1.3.27 SetCameraVelocity()	16
5.1.3.28 SetVerticalFov()	16
5.1.3.29 SetX()	16
5.1.3.30 SetY()	17
5.1.3.31 SetZ()	17
5.1.3.32 SetZFar()	17
5.1.3.33 SetZNear()	17
5.1.3.34 Up()	17
5.1.3.35 UpdatePerspectiveProjectionMatrix()	17
5.1.3.36 UpdateViewMatrix()	18
5.1.3.37 UpdateViewPerspectiveProjectionMatrix()	18
5.2 FColor::Color Class Reference	18
5.2.1 Detailed Description	19
5.2.2 Constructor & Destructor Documentation	19
5.2.2.1 Color() [1/2]	19
5.2.2.2 Color() [2/2]	19
5.2.3 Member Function Documentation	20
5.2.3.1 GetAlpha()	20
5.2.3.2 GetBlue()	20
5.2.3.3 GetColor()	20
5.2.3.4 GetGreen()	20
5.2.3.5 GetRed()	20
5.2.3.6 operator*=( ) [1/2]	21
5.2.3.7 operator*=( ) [2/2]	21
5.2.3.8 operator+=( )	21
5.2.3.9 operator-=( )	21
5.2.3.10 SetAlpha()	21
5.2.3.11 SetBlue()	22
5.2.3.12 SetColor()	22
5.2.3.13 SetGreen()	22
5.2.3.14 SetRed()	22
5.3 FRender::ConstantBuffer Class Reference	22
5.3.1 Detailed Description	23
5.3.2 Constructor & Destructor Documentation	23
5.3.2.1 ~ConstantBuffer()	23
5.3.3 Member Function Documentation	23
5.3.3.1 CopyData()	23

5.3.3.2 CreateConstantBuffer()	24
5.3.3.3 CreateConstantBufferView()	24
5.4 FARender::DepthStencilBuffer Class Reference	24
5.4.1 Detailed Description	25
5.4.2 Constructor & Destructor Documentation	25
5.4.2.1 DepthStencilBuffer()	25
5.4.3 Member Function Documentation	25
5.4.3.1 ClearDepthStencilBuffer()	25
5.4.3.2 CreateDepthStencilBufferAndView()	25
5.4.3.3 GetDepthStencilFormat()	26
5.4.3.4 ResetBuffer()	26
5.5 FARender::DeviceResources Class Reference	26
5.5.1 Detailed Description	27
5.5.2 Constructor & Destructor Documentation	27
5.5.2.1 ~DeviceResources()	28
5.5.3 Member Function Documentation	28
5.5.3.1 AfterTextDraw()	28
5.5.3.2 BeforeTextDraw()	28
5.5.3.3 DisableMSAA()	28
5.5.3.4 DisableText()	28
5.5.3.5 EnableMSAA()	29
5.5.3.6 EnableText()	29
5.5.3.7 Execute()	29
5.5.3.8 FlushCommandQueue()	29
5.5.3.9 GetBackBufferFormat()	29
5.5.3.10 GetCBVSize()	30
5.5.3.11 GetCommandList()	30
5.5.3.12 GetCurrentFrame()	30
5.5.3.13 GetDepthStencilFormat()	30
5.5.3.14 GetDevice()	30
5.5.3.15 GetInstance()	30
5.5.3.16 GetNumFrames()	31
5.5.3.17 GetTextResources()	31
5.5.3.18 IsMSAAEnabled()	31
5.5.3.19 IsTextEnabled()	31
5.5.3.20 NextFrame()	31
5.5.3.21 Present()	31
5.5.3.22 Resize()	32
5.5.3.23 RTBufferTransition()	32
5.5.3.24 Signal()	32
5.5.3.25 UpdateCurrentFrameFenceValue()	32
5.5.3.26 WaitForGPU()	32

5.6 DirectXException Class Reference . . . . .	33
5.7 FARender::DrawSettings Struct Reference . . . . .	33
5.7.1 Detailed Description . . . . .	33
5.8 FARender::IndexBuffer Class Reference . . . . .	33
5.8.1 Detailed Description . . . . .	34
5.8.2 Member Function Documentation . . . . .	34
5.8.2.1 CreateIndexBuffer() . . . . .	34
5.8.2.2 CreateIndexBufferView() . . . . .	34
5.8.2.3 GetIndexBufferView() . . . . .	34
5.9 FARender::MultiSampling Class Reference . . . . .	35
5.9.1 Detailed Description . . . . .	35
5.9.2 Constructor & Destructor Documentation . . . . .	35
5.9.2.1 MultiSampling() . . . . .	36
5.9.3 Member Function Documentation . . . . .	36
5.9.3.1 ClearDepthStencilBuffer() . . . . .	36
5.9.3.2 ClearRenderTargetBuffer() . . . . .	36
5.9.3.3 CreateDepthStencilBufferAndView() . . . . .	36
5.9.3.4 CreateRenderTargetBufferAndView() . . . . .	37
5.9.3.5 GetRenderTargetBuffer() . . . . .	37
5.9.3.6 ResetBuffers() . . . . .	37
5.9.3.7 Transition() . . . . .	37
5.10 FARender::RenderScene Class Reference . . . . .	37
5.10.1 Detailed Description . . . . .	40
5.10.2 Member Function Documentation . . . . .	40
5.10.2.1 AddDrawArgument() . . . . .	40
5.10.2.2 AddIndices() [1/2] . . . . .	40
5.10.2.3 AddIndices() [2/2] . . . . .	41
5.10.2.4 AddVertices() [1/2] . . . . .	41
5.10.2.5 AddVertices() [2/2] . . . . .	41
5.10.2.6 AfterDraw() . . . . .	41
5.10.2.7 AfterDrawObjects() . . . . .	41
5.10.2.8 AfterDrawText() . . . . .	41
5.10.2.9 BeforeDrawObjects() . . . . .	42
5.10.2.10 BeforeDrawText() . . . . .	42
5.10.2.11 CopyData() . . . . .	42
5.10.2.12 CreateCBVHeap() . . . . .	42
5.10.2.13 CreateConstantBuffer() . . . . .	42
5.10.2.14 CreateConstantBufferView() . . . . .	43
5.10.2.15 CreateDrawArgument() . . . . .	43
5.10.2.16 CreateDrawSettings() . . . . .	43
5.10.2.17 CreateIndexBuffer() . . . . .	43
5.10.2.18 CreateText() . . . . .	43

5.10.2.19 DisableMSAA()	44
5.10.2.20 DisableText()	44
5.10.2.21 DrawObjects()	44
5.10.2.22 EnableMSAA()	44
5.10.2.23 EnableText()	45
5.10.2.24 ExecuteAndFlush()	45
5.10.2.25 IsMSAAEnabled()	45
5.10.2.26 IsTextEnabled()	45
5.10.2.27 NextFrame()	45
5.10.2.28 RemoveDrawArgument()	45
5.10.2.29 RemoveDrawSettings()	46
5.10.2.30 RemoveText()	46
5.10.2.31 RenderText()	46
5.10.2.32 Resize()	46
5.10.2.33 SetPrimitive()	46
5.10.2.34 SetPSO()	47
5.10.2.35 SetRootSignature()	47
5.11 FARender::RenderTargetBuffer Class Reference	47
5.11.1 Detailed Description	48
5.11.2 Constructor & Destructor Documentation	48
5.11.2.1 RenderTargetBuffer()	48
5.11.3 Member Function Documentation	48
5.11.3.1 ClearRenderTargetBuffer()	48
5.11.3.2 CreateRenderTargetBufferAndView()	48
5.11.3.3 GetRenderTargetBuffer() [1/2]	49
5.11.3.4 GetRenderTargetBuffer() [2/2]	49
5.11.3.5 GetRenderTargetFormat()	49
5.11.3.6 ResetBuffer()	49
5.12 FARender::SwapChain Class Reference	49
5.12.1 Detailed Description	50
5.12.2 Constructor & Destructor Documentation	50
5.12.2.1 SwapChain()	51
5.12.3 Member Function Documentation	51
5.12.3.1 ClearCurrentBackBuffer()	51
5.12.3.2 ClearDepthStencilBuffer()	51
5.12.3.3 CreateDepthStencilBufferAndView()	51
5.12.3.4 CreateRenderTargetBuffersAndViews()	52
5.12.3.5 GetBackBufferFormat()	52
5.12.3.6 GetCurrentBackBuffer()	52
5.12.3.7 GetCurrentBackBufferIndex()	52
5.12.3.8 GetDepthStencilFormat()	52
5.12.3.9 GetNumRenderTargetBuffers()	52

5.12.3.10 GetRenderTargetBuffers()	53
5.12.3.11 Present()	53
5.12.3.12 ResetBuffers()	53
5.12.3.13 ResizeSwapChain()	53
5.12.3.14 Transition()	53
5.13 FAREnder::Text Class Reference	54
5.13.1 Detailed Description	54
5.13.2 Constructor & Destructor Documentation	54
5.13.2.1 Text()	54
5.13.3 Member Function Documentation	55
5.13.3.1 GetTextColor()	55
5.13.3.2 GetTextLocation()	55
5.13.3.3 GetTextSize()	55
5.13.3.4 GetTextString()	55
5.13.3.5 SetTextColor()	55
5.13.3.6 SetTextLocation()	56
5.13.3.7 SetTextSize()	56
5.13.3.8 SetTextString()	56
5.14 FAREnder::TextResources Class Reference	56
5.14.1 Detailed Description	57
5.14.2 Constructor & Destructor Documentation	57
5.14.2.1 TextResources()	57
5.14.3 Member Function Documentation	57
5.14.3.1 AfterRenderText()	57
5.14.3.2 BeforeRenderText()	57
5.14.3.3 GetDirect2DDeviceContext()	57
5.14.3.4 GetDirectWriteFactory()	58
5.14.3.5 ResetBuffers()	58
5.14.3.6 ResizeBuffers()	58
5.15 FATime::Time Class Reference	58
5.15.1 Constructor & Destructor Documentation	59
5.15.1.1 Time()	59
5.15.2 Member Function Documentation	59
5.15.2.1 DeltaTime()	59
5.15.2.2 Reset()	59
5.15.2.3 Start()	59
5.15.2.4 Stop()	59
5.15.2.5 Tick()	60
5.15.2.6 TotalTime()	60
5.16 Time Class Reference	60
5.16.1 Detailed Description	60
5.17 FAREnder::VertexBuffer Class Reference	60



5.17.1 Detailed Description	61
5.17.2 Member Function Documentation	61
5.17.2.1 CreateVertexBuffer()	61
5.17.2.2 CreateVertexBufferView()	61
5.17.2.3 GetVertexBufferView()	61
5.18 FAWindow::Window Class Reference	61
5.18.1 Detailed Description	62
5.18.2 Constructor & Destructor Documentation	62
5.18.2.1 Window() [1/2]	62
5.18.2.2 Window() [2/2]	63
5.18.3 Member Function Documentation	63
5.18.3.1 GetHeight()	63
5.18.3.2 GetWidth()	63
5.18.3.3 GetWindowHandle()	63
5.18.3.4 SetHeight()	63
5.18.3.5 SetWidth()	63
<b>6 File Documentation</b>	<b>65</b>
6.1 Direct3DLink.h	65
6.2 FABuffer.h File Reference	65
6.2.1 Detailed Description	66
6.3 FABuffer.h	66
6.4 FACamera.h File Reference	67
6.4.1 Detailed Description	68
6.4.2 Typedef Documentation	68
6.4.2.1 vec2	68
6.5 FACamera.h	68
6.6 FAColor.h File Reference	70
6.6.1 Detailed Description	70
6.6.2 Function Documentation	70
6.6.2.1 operator*() [1/3]	71
6.6.2.2 operator*() [2/3]	71
6.6.2.3 operator*() [3/3]	71
6.6.2.4 operator+()	71
6.6.2.5 operator-()	71
6.7 FAColor.h	72
6.8 FADeviceResources.h File Reference	72
6.8.1 Detailed Description	73
6.9 FADeviceResources.h	73
6.10 FADirectXException.h	74
6.11 FAMultiSampling.h	75
6.12 FARenderScene.h File Reference	76

6.12.1 Detailed Description . . . . .	76
6.13 FARenderScene.h . . . . .	77
6.14 FASwapChain.h . . . . .	79
6.15 FAText.h File Reference . . . . .	80
6.15.1 Detailed Description . . . . .	81
6.16 FAText.h . . . . .	81
6.17 FATextResources.h . . . . .	81
6.18 FATime.h File Reference . . . . .	82
6.18.1 Detailed Description . . . . .	82
6.19 FATime.h . . . . .	82
6.20 FAWindow.h File Reference . . . . .	83
6.20.1 Detailed Description . . . . .	83
6.21 FAWindow.h . . . . .	84
<b>Index</b>	<b>85</b>

# Chapter 1

## Namespace Index

### 1.1 Namespace List

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

<a href="#">FACamera</a>		
	Has <a href="#">Camera</a> class . . . . .	<a href="#">7</a>
<a href="#">FARender</a>		
	Has classes that are used for rendering objects and text through the Direct3D 12 API . . . . .	<a href="#">7</a>
<a href="#">FAWindow</a>		
	Has <a href="#">Window</a> class . . . . .	<a href="#">8</a>



## Chapter 2

# Class Index

### 2.1 Class List

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

#### [FACamera::Camera](#)

Simple first person style camera class that lets the viewer explore the 3D scene. It keeps track of the camera coordinate system relative to the world space so that the view matrix can be constructed.

It keeps track of the viewing frustum of the camera so that the projection matrix can be obtained.

9

#### [FAColor::Color](#)

This class stores a RGBA color in a 4D vector using floats. The range of each component is [0.0, 1.0]. The first componet is red, second component is green, third component is blue and the 4th component is alpha . . . . . 18

#### [FARender::ConstantBuffer](#)

This class stores constant data in a Direct3D 12 upload buffer . . . . . 22

#### [FARender::DepthStencilBuffer](#)

A wrapper for depth stencil buffer resources. Uses DirectD 12 API . . . . . 24

#### [FARender::DeviceResources](#)

A wrapper for resources that are needed to render objects and text using the Direct3D 12 API . 26

#### [DirectXException](#)

. . . . . 33

#### [FARender::DrawSettings](#)

Holds a array of objects that use the same PSO, root signature and primitive . . . . . 33

#### [FARender::IndexBuffer](#)

This class stores indices in a Direct3D 12 default buffer . . . . . 33

#### [FARender::MultiSampling](#)

A wrapper for multisampling resources. Uses DirectD 12 API . . . . . 35

#### [FARender::RenderScene](#)

This class is used to render a scene using Direct3D 12 API . . . . . 37

#### [FARender::RenderTargetBuffer](#)

A wrapper for render target buffer resources. Uses DirectD 12 API . . . . . 47

#### [FARender::SwapChain](#)

A wrapper for swap chain resources. Uses DirectD 12 API and DXGI API . . . . . 49

#### [FARender::Text](#)

This class is used to help render text. Stores the location of the text, the text string, text size and the color of the text . . . . . 54

[FARender::TextResources](#)

A wrapper for resources that are needed to render text using Direct3D 11on12, Direct2D and DirectWrite . . . . . 56

[FATime::Time](#) . . . . . 58[Time](#)

This class is used to get the time between each frame. You can stop start, reset and get the total time . . . . . 60

[FARender::VertexBuffer](#)

This class stores vertices in a Direct3D 12 default buffer . . . . . 60

[FAWindow::Window](#)

The window class is used to make a [Window](#) using Windows API . . . . . 61

## Chapter 3

# File Index

### 3.1 File List

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

<a href="#">Direct3DLink.h</a> . . . . .	??
<a href="#">FABuffer.h</a>	
File has classes VertexBuffer, IndexBuffer and ConstantBuffer under namespace <a href="#">FARender</a> . .	65
<a href="#">FACamera.h</a>	
File that has namespace <a href="#">FACamera</a> . Withn the namespace is the class Camera . . . . .	67
<a href="#">FAColor.h</a>	
File has class Color under namespace <a href="#">FAColor</a> . . . . .	70
<a href="#">FADeviceResources.h</a>	
File has class DeviceResources under namespace <a href="#">FARender</a> . . . . .	72
<a href="#">FADirectXException.h</a> . . . . .	??
<a href="#">FAMultiSampling.h</a> . . . . .	??
<a href="#">FARenderScene.h</a>	
File has class RenderScene under namespace <a href="#">FARender</a> . . . . .	76
<a href="#">FASwapChain.h</a> . . . . .	??
<a href="#">FAText.h</a>	
File has class Text under namespace <a href="#">FARender</a> . . . . .	80
<a href="#">FATextResources.h</a> . . . . .	??
<a href="#">FATime.h</a>	
File that has namespace <a href="#">FATime</a> . Withn the namespace is the class <a href="#">Time</a> . . . . .	82
<a href="#">FAWindow.h</a>	
File that has namespace <a href="#">FAWindow</a> . Withn the namespace is the class Window . . . . .	83





## Chapter 4

# Namespace Documentation

### 4.1 FACamera Namespace Reference

Has [Camera](#) class.

#### Classes

- class [Camera](#)

*Simple first person style camera class that lets the viewer explore the 3D scene.*

*It keeps track of the camera coordinate system relative to the world space so that the view matrix can be constructed.*

*It keeps track of the viewing frustum of the camera so that the projection matrix can be obtained.*

.

#### 4.1.1 Detailed Description

Has [Camera](#) class.

### 4.2 FARender Namespace Reference

Has classes that are used for rendering objects and text through the Direct3D 12 API.

#### Classes

- class [ConstantBuffer](#)

*This class stores constant data in a Direct3D 12 upload buffer.*

- class [DepthStencilBuffer](#)

*A wrapper for depth stencil buffer resources. Uses DirectD 12 API.*

- class [DeviceResources](#)

*A wrapper for resources that are needed to render objects and text using the Direct3D 12 API.*

- struct [DrawSettings](#)

*Holds a array of objects that use the same PSO, root signature and primitive.*

- class [IndexBuffer](#)  
*This class stores indices in a Direct3D 12 default buffer.*
- class [MultiSampling](#)  
*A wrapper for multisampling resources. Uses DirectD 12 API.*
- class [RenderScene](#)  
*This class is used to render a scene using Direct3D 12 API.*
- class [RenderTargetBuffer](#)  
*A wrapper for render target buffer resources. Uses DirectD 12 API.*
- class [SwapChain](#)  
*A wrapper for swap chain resources. Uses DirectD 12 API and DXGI API.*
- class [Text](#)  
*This class is used to help render text. Stores the location of the text, the text string, text size and the color of the text.*
- class [TextResources](#)  
*A wrapper for resources that are needed to render text using Direct3D 11on12, Direct2D and DirectWrite.*
- class [VertexBuffer](#)  
*This class stores vertices in a Direct3D 12 default buffer.*

#### 4.2.1 Detailed Description

Has classes that are used for rendering objects and text through the Direct3D 12 API.

### 4.3 FAWindow Namespace Reference

Has [Window](#) class.

#### Classes

- class [Window](#)  
*The window class is used to make a [Window](#) using Windows API.*

#### 4.3.1 Detailed Description

Has [Window](#) class.

## Chapter 5

# Class Documentation

### 5.1 FACamera::Camera Class Reference

Simple first person style camera class that lets the viewer explore the 3D scene.

It keeps track of the camera coordinate system relative to the world space so that the view matrix can be constructed.

It keeps track of the viewing frustum of the camera so that the projection matrix can be obtained.

```
#include "FACamera.h"
```

#### Public Member Functions

- [Camera](#) (vec3 cameraPosition=vec3(0.0f, 0.0f, 0.0f), vec3 x=vec3(1.0f, 0.0f, 0.0f), vec3 y=vec3(0.0f, 1.0f, 0.0f), vec3 z=vec3(0.0f, 0.0f, 1.0f), float znear=1.0f, float zfar=100.f, float aspectRatio=1.0f, float vFov=45.0f, float cameraVelocity=10.0f, float angularVelocity=0.25f)  
*Constructor.*
- const vec3 & [GetCameraPosition](#) () const  
*Returns a constant reference to the position of the camera in world coordinates.*
- const vec3 & [GetX](#) () const  
*Returns a constant reference to the x-axis of the camera.*
- const vec3 & [GetY](#) () const  
*Returns a constant reference to the y-axis of the camera.*
- const vec3 & [GetZ](#) () const  
*Returns a constant reference to the z-axis of the camera.*
- const mat4 & [GetViewTransformationMatrix](#) () const  
*Returns a constant reference to the view transformation matrix of this camera.*
- float [GetCameraVelocity](#) () const  
*Returns the camera's velocity.*
- float [GetAngularVelocity](#) () const  
*Returns the camera's angular velocity.*
- void [LookAt](#) (vec3 cameraPosition, vec3 target, vec3 up)  
*Defines the camera space using UVN.*
- float [GetZNear](#) () const  
*Returns the near value of the frustum.*
- float [GetZFar](#) () const

- Returns the far value of the frustrum.*

  - float [GetVerticalFov](#) () const

*Returns the vertical field of view of the frustrum in degrees.*
- float [GetAspectRatio](#) () const

*Returns the aspect ratio of the frustrum.*
- void [SetCameraPosition](#) (const vec3 &position)

*Sets the camera's position to the specified position.*
- void [SetX](#) (const vec3 &x)

*Sets the camera's x-axis to the specified vector.*
- void [SetY](#) (const vec3 &y)

*Sets the camera's y-axis to the specified vector.*
- void [SetZ](#) (const vec3 &z)

*Sets the camera's z-axis to the specified vector.*
- void [SetCameraVelocity](#) (float velocity)

*Sets the camera's velocity to the specified velocity.*
- void [SetAngularVelocity](#) (float velocity)

*Sets the camera's angular velocity to the specified angular velocity.*
- void [SetZNear](#) (float znear)

*Sets the camera's near plane z value to the specified value.*
- void [SetZFar](#) (float zfar)

*Sets the camera's far plane z value to the specified value.*
- void [SetVerticalFov](#) (float fov)

*Sets the camera's vertical field of view to the specified vertical field of view .*
- void [SetAspectRatio](#) (float ar)

*Sets the camera's aspect ratio to the specified aspect ratio.*
- const mat4 & [GetPerspectiveProjectionMatrix](#) () const

*Returns a constant reference to the perspective projection transformation matrix of this camera.*
- const mat4 & [GetViewPerspectiveProjectionMatrix](#) () const

*Returns a constant reference to the view perspective projection transformation matrix of this camera.*
- void [UpdateViewMatrix](#) ()

*After modifying the camera position and/or orientation, call this to rebuild the view transformation matrix.*
- void [UpdatePerspectiveProjectionMatrix](#) ()

*After modifying any of the frustrum properties, call this to rebuild the perspective projection transformation matrix.*
- void [UpdateViewPerspectiveProjectionMatrix](#) ()

*After modifying view and/or perspective projection transformation matrix, call this to rebuild the view perspective projection transformation matrix.*
- void [Left](#) (float dt)

*Moves the camera left along the camera's x-axis.*
- void [Right](#) (float dt)

*Moves the camera right along the camera's x-axis.*
- void [Forward](#) (float dt)

*Moves the camera forward along the camera's z-axis.*
- void [Backward](#) (float dt)

*Moves the camera backward along the camera's z-axis.*
- void [Up](#) (float dt)

*Moves the camera up along the camera's y-axis.*
- void [Down](#) (float dt)

*Moves the camera down along the camera's y-axis.*
- void [RotateCameraLeftRight](#) (float xDiff)

*Rotates the camera to look left and right.*
- void [RotateCameraUpDown](#) (float yDiff)

*Rotates the camera to look up and down.*

- void [KeyboardInput](#) (float dt)

*Polls keyboard input and moves the camera. Moves the camera forward/backward if w/s or up/down arrow was pressed. Moves the camera left/right if a/d or left/right arrow was pressed. Moves the camera up/down if space/crtl was pressed.*

- void [MouseInput](#) ()

*Rotates camera on mouse movement.*

### 5.1.1 Detailed Description

Simple first person style camera class that lets the viewer explore the 3D scene.

It keeps track of the camera coordinate system relative to the world space so that the view matrix can be constructed.

It keeps track of the viewing frustum of the camera so that the projection matrix can be obtained.

### 5.1.2 Constructor & Destructor Documentation

#### 5.1.2.1 Camera()

```
FACamera::Camera::Camera (
    vec3 cameraPosition = vec3(0.0f, 0.0f, 0.0f),
    vec3 x = vec3(1.0f, 0.0f, 0.0f),
    vec3 y = vec3(0.0f, 1.0f, 0.0f),
    vec3 z = vec3(0.0f, 0.0f, 1.0f),
    float znear = 1.0f,
    float zfar = 100.f,
    float aspectRatio = 1.0f,
    float vFov = 45.0f,
    float cameraVelocity = 10.0f,
    float angularVelocity = 0.25f )
```

Constructor.

Creates a new camera.

Sets the origin of the camera space to the given cameraPosition.

Sets the axis of the camera space to the given x, y and z vectors.

The origin and basis vectors of the camera space should be relative to world space.

Sets the frustum properties for perspective projection to the given znear, zar, aspectRatio and fov values.

vFov should be in degrees.

The constant velocity of the camera when moved is set to the given cameraVelocity; The angular velocity of the camera is set the to specified angularVelocity.

### 5.1.3 Member Function Documentation

#### 5.1.3.1 Backward()

```
void FACamera::Camera::Backward (
    float dt )
```

Moves the camera backward along the camera's z-axis.

#### 5.1.3.2 Down()

```
void FACamera::Camera::Down (
    float dt )
```

Moves the camera down along the camera's y-axis.

#### 5.1.3.3 Foward()

```
void FACamera::Camera::Foward (
    float dt )
```

Moves the camera foward along the camera's z-axis.

#### 5.1.3.4 GetAngularVelocity()

```
float FACamera::Camera::GetAngularVelocity ( ) const
```

Returns the camera's angular velocity.

#### 5.1.3.5 GetAspectRatio()

```
float FACamera::Camera::GetAspectRatio ( ) const
```

Returns the aspect ratio of the frustum.

#### 5.1.3.6 GetCameraPosition()

```
const vec3 & FACamera::Camera::GetCameraPosition ( ) const
```

Returns a constant reference to the position of the camera in world coordinates.

#### 5.1.3.7 GetCameraVelocity()

```
float FACamera::Camera::GetCameraVelocity ( ) const
```

Returns the camera's velocity.

#### 5.1.3.8 GetPerspectiveProjectionMatrix()

```
const mat4 & FACamera::Camera::GetPerspectiveProjectionMatrix ( ) const
```

Returns a constant reference to the perspective projection transformation matrix of this camera.

#### 5.1.3.9 GetVerticalFov()

```
float FACamera::Camera::GetVerticalFov ( ) const
```

Returns the vertical field of view of the frustum in degrees.

#### 5.1.3.10 GetViewPerspectiveProjectionMatrix()

```
const mat4 & FACamera::Camera::GetViewPerspectiveProjectionMatrix ( ) const
```

Returns a constant reference to the view perspective projection transformation matrix of this camera.

#### 5.1.3.11 GetViewTransformationMatrix()

```
const mat4 & FACamera::Camera::GetViewTransformationMatrix ( ) const
```

Returns a constant reference to the view transformation matrix of this camera.

#### 5.1.3.12 GetX()

```
const vec3 & FACamera::Camera::GetX ( ) const
```

Returns a constant reference to the x-axis of the camera.

#### 5.1.3.13 GetY()

```
const vec3 & FACamera::Camera::GetY ( ) const
```

Returns a constant reference to the y-axis of the camera.

#### 5.1.3.14 GetZ()

```
const vec3 & FACamera::Camera::GetZ ( ) const
```

Returns a constant reference to the z-axis of the camera.

#### 5.1.3.15 GetZFar()

```
float FACamera::Camera::GetZFar ( ) const
```

Returns the far value of the frustum.

#### 5.1.3.16 GetZNear()

```
float FACamera::Camera::GetZNear ( ) const
```

Returns the near value of the frustum.

#### 5.1.3.17 KeyboardInput()

```
void FACamera::Camera::KeyboardInput (
    float dt )
```

Polls keyboard input and moves the camera. Moves the camera forward/backward if w/s or up/down arrow was pressed. Moves the camera left/right if a/d or left/right arrow was pressed. Moves the camera up/down if space/crtl was pressed.

#### 5.1.3.18 Left()

```
void FACamera::Camera::Left (
    float dt )
```

Moves the camera left along the camera's x-axis.



#### 5.1.3.19 LookAt()

```
void FACamera::Camera::LookAt (
    vec3 cameraPosition,
    vec3 target,
    vec3 up )
```

Defines the camera space using UVN.

#### 5.1.3.20 MouseInput()

```
void FACamera::Camera::MouseInput ( )
```

Rotates camera on mouse movement.

#### 5.1.3.21 Right()

```
void FACamera::Camera::Right (
    float dt )
```

Moves the camera right along the camera's x-axis.

#### 5.1.3.22 RotateCameraLeftRight()

```
void FACamera::Camera::RotateCameraLeftRight (
    float xDiff )
```

Rotates the camera to look left and right.

#### 5.1.3.23 RotateCameraUpDown()

```
void FACamera::Camera::RotateCameraUpDown (
    float yDiff )
```

Rotates the camera to look up and down.

#### 5.1.3.24 SetAngularVelocity()

```
void FACamera::Camera::SetAngularVelocity (
    float velcoity )
```

Sets the camera's angular velocity to the specified angular velocity.

#### 5.1.3.25 SetAspectRatio()

```
void FACamera::Camera::SetAspectRatio (
    float ar )
```

Sets the camera's aspect ratio to the specified aspect ratio.

#### 5.1.3.26 SetCameraPosition()

```
void FACamera::Camera::SetCameraPosition (
    const vec3 & position )
```

Sets the camera's position to the specified position.

#### 5.1.3.27 SetCameraVelocity()

```
void FACamera::Camera::SetCameraVelocity (
    float velocity )
```

Sets the camera's velocity to the specified velocity.

#### 5.1.3.28 SetVerticalFov()

```
void FACamera::Camera::SetVerticalFov (
    float fov )
```

Sets the camera's vertical field of view to the specified vertical field of view .

#### 5.1.3.29 SetX()

```
void FACamera::Camera::SetX (
    const vec3 & x )
```

Sets the camera's x-axis to the specified vector.

#### 5.1.3.30 SetY()

```
void FACamera::Camera::SetY (
    const vec3 & y )
```

Sets the camera's y-axis to the specified vector.

#### 5.1.3.31 SetZ()

```
void FACamera::Camera::SetZ (
    const vec3 & z )
```

Sets the camera's z-axis to the specified vector.

#### 5.1.3.32 SetZFar()

```
void FACamera::Camera::SetZFar (
    float zfar )
```

Sets the camera's far plane z value to the specified value.

#### 5.1.3.33 SetZNear()

```
void FACamera::Camera::SetZNear (
    float znear )
```

Sets the camera's near plane z value to the specified value.

#### 5.1.3.34 Up()

```
void FACamera::Camera::Up (
    float dt )
```

Moves the camera up along the camera's y-axis.

#### 5.1.3.35 UpdatePerspectiveProjectionMatrix()

```
void FACamera::Camera::UpdatePerspectiveProjectionMatrix ( )
```

After modifying any of the frustrum properties, call this to rebuild the perspective projection transformation matrix.

### 5.1.3.36 UpdateViewMatrix()

```
void FACamera::Camera::UpdateViewMatrix ( )
```

After modifying the camera position and/or orientation, call this to rebuild the view transformation matrix.

### 5.1.3.37 UpdateViewPerspectiveProjectionMatrix()

```
void FACamera::Camera::UpdateViewPerspectiveProjectionMatrix ( )
```

After modifying view and/or perspective projection transformation matrix, call this to rebuild the view perspective projection transformation matrix.

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

- [FACamera.h](#)

## 5.2 FAColor::Color Class Reference

This class stores a RGBA color in a 4D vector using floats. The range of each component is [0.0, 1.0]. The first componet is red, second component is green, third component is blue and the 4th component is alpha.

```
#include "FAColor.h"
```

### Public Member Functions

- [Color](#) (float r=0.0f, float g=0.0f, float b=0.0f, float a=1.0f)  
*Default Constructor. Initializes the color to the specified RGBA values.*
- [Color](#) (const FAMath::Vector4D &color)  
*Overloaded Constructor. Initializes the color to the specified color.*
- const FAMath::Vector4D & [GetColor](#) () const  
*Returns the color.*
- float [GetRed](#) () const  
*Returns the value of the red component.*
- float [GetGreen](#) () const  
*Returns the value of the blue component.*
- float [GetBlue](#) () const  
*Returns the value of the green component.*
- float [GetAlpha](#) () const  
*Returns the value of the alpha component.*
- void [SetColor](#) (const FAMath::Vector4D &color)  
*Sets the color to the specified color.*
- void [SetRed](#) (float r)  
*Sets the red component to the specified float value.*
- void [SetGreen](#) (float g)  
*Sets the green component to the specified float value.*
- void [SetBlue](#) (float b)

- Sets the blue component to the specified float value.*

  - void **SetAlpha** (float a)

*Sets the alpha component to the specified float value.*
  - **Color & operator+=** (const **Color** &c)

*Adds this objects color to the specified color and stores the result in this object. Does component-wise addition. If any of the resultant components are > 1.0f, they are set to 1.0f.*
  - **Color & operator-=** (const **Color** &c)

*Subtracts the specified color from this objects color and stores the result in this object. Does component-wise subtraction. If any of the resultant components are < 0.0f, they are set to 0.0f.*
  - **Color & operator\*=** (float k)

*Multiplies this objects color by the specified float value k and stores the result in this object. If k < 0.0f, no multiplication happens and this objects color does not get modified. If any of the resultant components are > 1.0f, they are set to 1.0f.*
  - **Color & operator\*=** (const **Color** &c)

*Multiplies this objects color by the specified color c and stores the result in this object. If any of the resultant components are > 1.0f, they are set to 1.0f. Does component-wise multiplication.*

### 5.2.1 Detailed Description

This class stores a RGBA color in a 4D vector using floats. The range of each component is [0.0, 1.0]. The first componet is red, second component is green, third component is blue and the 4th component is alpha.

### 5.2.2 Constructor & Destructor Documentation

#### 5.2.2.1 Color() [1/2]

```
FColor::Color::Color (
    float r = 0.0f,
    float g = 0.0f,
    float b = 0.0f,
    float a = 1.0f )
```

Default Constructor. Initializes the color to the specified RGBA values.

#### 5.2.2.2 Color() [2/2]

```
FColor::Color::Color (
    const FMath::Vector4D & color )
```

Overloaded Constructor. Initializes the color to the specified color.

## 5.2.3 Member Function Documentation

### 5.2.3.1 GetAlpha()

```
float FColor::Color::GetAlpha ( ) const
```

Returns the value of the alpha component.

### 5.2.3.2 GetBlue()

```
float FColor::Color::GetBlue ( ) const
```

Returns the value of the green component.

### 5.2.3.3 GetColor()

```
const FMath::Vector4D & FColor::Color::GetColor ( ) const
```

Returns the color.

### 5.2.3.4 GetGreen()

```
float FColor::Color::GetGreen ( ) const
```

Returns the value of the blue component.

### 5.2.3.5 GetRed()

```
float FColor::Color::GetRed ( ) const
```

Returns the value of the red component.

### 5.2.3.6 operator\*=( ) [1/2]

```
Color & FColor::Color::operator*= (
    const Color & c )
```

Multiplies this objects color by the specified color c and stores the result in this object. If any of the resultant components are  $> 1.0f$ , they are set to  $1.0f$ . Does component-wise multiplication.

### 5.2.3.7 operator\*=( ) [2/2]

```
Color & FColor::Color::operator*= (
    float k )
```

Multiplies this objects color by the specified float value k and stores the result in this object. If  $k < 0.0f$ , no multiplication happens and this objects color does not get modified. If any of the resultant components are  $> 1.0f$ , they are set to  $1.0f$ .

.

### 5.2.3.8 operator+=( )

```
Color & FColor::Color::operator+= (
    const Color & c )
```

Adds this objects color to the specified color and stores the result in this object. Does component-wise addition. If any of the resultant components are  $> 1.0f$ , they are set to  $1.0f$ .

### 5.2.3.9 operator-=( )

```
Color & FColor::Color::operator-= (
    const Color & c )
```

Subtracts the specified color from this objects color and stores the result in this object. Does component-wise subtraction. If any of the resultant components are  $< 0.0f$ , they are set to  $0.0f$ .

### 5.2.3.10 SetAlpha()

```
void FColor::Color::SetAlpha (
    float a )
```

Sets the alpha component to the specified float value.

#### 5.2.3.11 SetBlue()

```
void FColor::Color::SetBlue (
    float b )
```

Sets the blue component to the specified float value.

#### 5.2.3.12 SetColor()

```
void FColor::Color::SetColor (
    const FAMath::Vector4D & color )
```

Sets the color to the specified color.

#### 5.2.3.13 SetGreen()

```
void FColor::Color::SetGreen (
    float g )
```

Sets the green component to the specified float value.

#### 5.2.3.14 SetRed()

```
void FColor::Color::SetRed (
    float r )
```

Sets the red component to the specified float value.

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

- [FColor.h](#)

## 5.3 FRender::ConstantBuffer Class Reference

This class stores constant data in a Direct3D 12 upload buffer.

```
#include "FABuffer.h"
```



## Public Member Functions

- **ConstantBuffer** (const [ConstantBuffer](#) &)=delete
- **ConstantBuffer & operator=** (const [ConstantBuffer](#) &)=delete
- **~ConstantBuffer** ()  
*Unmaps the pointer to the constant buffer.*
- void **CreateConstantBuffer** (const Microsoft::WRL::ComPtr< ID3D12Device > &device, const UINT &numOfBytes)  
*Creates and maps the constant buffer. The number of bytes allocated should be a multiple of 256 bytes.*
- void **CreateConstantBufferView** (const Microsoft::WRL::ComPtr< ID3D12Device > &device, const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > &cbvHeap, UINT cbvSize, UINT cBufferIndex, UINT cbvHeapIndex, UINT numBytes)  
*Creates and maps the constant buffer view and stores it in the specified descriptor heap.*
- void **CopyData** (UINT index, UINT byteSize, const void \*data, UINT64 numBytes)  
*Copies data from the given data into the constant buffer. Uses 0-indexing.*

### 5.3.1 Detailed Description

This class stores constant data in a Direct3D 12 upload buffer.

### 5.3.2 Constructor & Destructor Documentation

#### 5.3.2.1 ~ConstantBuffer()

```
FARender::ConstantBuffer::~ConstantBuffer ( )
```

Unmaps the pointer to the constant buffer.

### 5.3.3 Member Function Documentation

#### 5.3.3.1 CopyData()

```
void FAREnder::ConstantBuffer::CopyData (
    UINT index,
    UINT byteSize,
    const void * data,
    UINT64 numBytes )
```

Copies data from the given data into the constant buffer. Uses 0-indexing.

### 5.3.3.2 CreateConstantBuffer()

```
void FARender::ConstantBuffer::CreateConstantBuffer (
    const Microsoft::WRL::ComPtr< ID3D12Device > & device,
    const UINT & numOfBytes )
```

Creates and maps the constant buffer. The number of bytes allocated should be a multiple of 256 bytes.

### 5.3.3.3 CreateConstantBufferView()

```
void FARender::ConstantBuffer::CreateConstantBufferView (
    const Microsoft::WRL::ComPtr< ID3D12Device > & device,
    const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > & cbvHeap,
    UINT cbvSize,
    UINT cBufferIndex,
    UINT cbvHeapIndex,
    UINT numBytes )
```

Creates and maps the constant buffer view and stores it in the specified descriptor heap.

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

- [FABuffer.h](#)

## 5.4 FARender::DepthStencilBuffer Class Reference

A wrapper for depth stencil buffer resources. Uses DirectD 12 API.

```
#include "FABuffer.h"
```

### Public Member Functions

- [DepthStencilBuffer](#) (DXGI\_FORMAT format=DXGI\_FORMAT\_D24\_UNORM\_S8\_UINT)  
*Default Constructor.*
- DXGI\_FORMAT [GetDepthStencilFormat](#) () const  
*Returns the format of the depth stencil buffer.*
- void [CreateDepthStencilBufferAndView](#) (const Microsoft::WRL::ComPtr< ID3D12Device > &device, const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > &dsvHeap, unsigned int indexWhereToStoreView, unsigned int dsvSize, unsigned int width, unsigned int height, unsigned int sampleCount=1)  
*Creates the depth stencil buffer and view.*
- void [ResetBuffer](#) ()  
*Resest the depth stencil buffer.*
- void [ClearDepthStencilBuffer](#) (const Microsoft::WRL::ComPtr< ID3D12GraphicsCommandList > &commandList, const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > &dsvHeap, unsigned int indexOfView, unsigned int dsvSize, float clearValue)  
*Clears the depth stencil buffer with the specified clear value.*

### 5.4.1 Detailed Description

A wrapper for depth stencil buffer resources. Uses DirectD 12 API.

### 5.4.2 Constructor & Destructor Documentation

#### 5.4.2.1 DepthStencilBuffer()

```
FARender::DepthStencilBuffer::DepthStencilBuffer (
    DXGI_FORMAT format = DXGI_FORMAT_D24_UNORM_S8_UINT )
```

Default Constructor.

### 5.4.3 Member Function Documentation

#### 5.4.3.1 ClearDepthStencilBuffer()

```
void FARender::DepthStencilBuffer::ClearDepthStencilBuffer (
    const Microsoft::WRL::ComPtr< ID3D12GraphicsCommandList > & commandList,
    const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > & dsvHeap,
    unsigned int indexOfView,
    unsigned int dsvSize,
    float clearValue )
```

Clears the depth stencil buffer with the specified clear value.

#### 5.4.3.2 CreateDepthStencilBufferAndView()

```
void FARender::DepthStencilBuffer::CreateDepthStencilBufferAndView (
    const Microsoft::WRL::ComPtr< ID3D12Device > & device,
    const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > & dsvHeap,
    unsigned int indexOfWhereToStoreView,
    unsigned int dsvSize,
    unsigned int width,
    unsigned int height,
    unsigned int sampleCount = 1 )
```

Creates the depth stencil buffer and view.

### 5.4.3.3 GetDepthStencilFormat()

```
DXGI_FORMAT FARender::DepthStencilBuffer::GetDepthStencilFormat ( ) const
```

Returns the format of the depth stencil buffer.

### 5.4.3.4 ResetBuffer()

```
void FARender::DepthStencilBuffer::ResetBuffer ( )
```

Resest the depth stencil buffer.

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

- [FABuffer.h](#)

## 5.5 FARender::DeviceResources Class Reference

A wrapper for resources that are needed to render objects and text using the Direct3D 12 API.

```
#include "FADeviceResources.h"
```

### Public Member Functions

- **DeviceResources** (const [DeviceResources](#) &)=delete
- **DeviceResources** & **operator=** (const [DeviceResources](#) &)=delete
- **~DeviceResources** ()  
*Flushes the command queue.*
- const Microsoft::WRL::ComPtr< ID3D12Device > & **GetDevice** () const  
*Returns a constant reference to the ID3D12Device objet.*
- const Microsoft::WRL::ComPtr< ID3D12GraphicsCommandList > & **GetCommandList** () const  
*Returns a constant reference to the ID3D12GraphicsCommandList objet.*
- DXGI\_FORMAT **GetBackBufferFormat** () const  
*Returns a constant reference to the back buffer format.*
- DXGI\_FORMAT **GetDepthStencilFormat** () const  
*Returns a constant reference to the depth stencil format.*
- unsigned int **GetCBVSize** () const  
*The size of a constant buffer view.*
- unsigned int **GetNumFrames** () const  
*Returns the number of frames.*
- unsigned int **GetCurrentFrame** () const  
*Returns the current frame.*
- const [TextResources](#) & **GetTextResources** () const  
*Returns a constant reference to the TextResources object.*
- bool **IsMSAAEnabled** () const  
*Returns true if MSAA is enabled, false otherwise.*

- void [DisableMSAA](#) (unsigned int width, unsigned int height, HWND windowHandle)  
*Disables MSAA.*
- void [EnableMSAA](#) (unsigned int width, unsigned int height, HWND windowHandle)  
*Enables MSAA.*
- bool [IsTextEnabled](#) () const  
*Returns true if text is enabled, false otherwise.*
- void [DisableText](#) (unsigned int width, unsigned int height, HWND windowHandle)  
*Disables text.*
- void [EnableText](#) (unsigned int width, unsigned int height, HWND windowHandle)  
*Enables text.*
- void [UpdateCurrentFrameFenceValue](#) ()  
*Updates the current frames fence value.*
- void [FlushCommandQueue](#) ()  
*Synchronizes the CPU and GPU. Use this function to make sure all of the commands in command list are executed by the GPU before the CPU writes in new commands.*
- void [WaitForGPU](#) () const  
*Waits for the GPU to execute all of the commands of the current frame. Signal should have been called before this function is called.*
- void [Signal](#) ()  
*Adds an instruction to the GPU to set the fence value to the current fence value.*
- void [Resize](#) (int width, int height, const HWND &handle)  
*Call when the window gets resized. Call when you initialize your program.*
- void [RTBufferTransition](#) ()  
*Transitions the render target buffer.*
- void [BeforeTextDraw](#) ()  
*Prepares to render text.*
- void [AfterTextDraw](#) ()  
*Executes the text commands.*
- void [Execute](#) () const  
*Executes the command list.*
- void [Present](#) ()  
*Swaps the front and back buffers.*
- void [Draw](#) ()
- void [NextFrame](#) ()  
*Updates the current frame value to go to the next frame.*

## Static Public Member Functions

- static [DeviceResources](#) & [GetInstance](#) (unsigned int width, unsigned int height, HWND windowHandle, unsigned int numFrames)  
*Call to make an object of [DeviceResources](#). This only allows one instance to exist.*

### 5.5.1 Detailed Description

A wrapper for resources that are needed to render objects and text using the Direct3D 12 API.

### 5.5.2 Constructor & Destructor Documentation

#### 5.5.2.1 ~DeviceResources()

```
FARender::DeviceResources::~~DeviceResources ( )
```

Flushes the command queue.

### 5.5.3 Member Function Documentation

#### 5.5.3.1 AfterTextDraw()

```
void FARender::DeviceResources::AfterTextDraw ( )
```

Executes the text commands.

#### 5.5.3.2 BeforeTextDraw()

```
void FARender::DeviceResources::BeforeTextDraw ( )
```

Prepares to render text.

#### 5.5.3.3 DisableMSAA()

```
void FARender::DeviceResources::DisableMSAA (
    unsigned int width,
    unsigned int height,
    HWND windowHandle )
```

Disables MSAA.

#### 5.5.3.4 DisableText()

```
void FARender::DeviceResources::DisableText (
    unsigned int width,
    unsigned int height,
    HWND windowHandle )
```

Disables text.

#### 5.5.3.5 EnableMSAA()

```
void FARender::DeviceResources::EnableMSAA (
    unsigned int width,
    unsigned int height,
    HWND windowHandle )
```

Enables MSAA.

#### 5.5.3.6 EnableText()

```
void FARender::DeviceResources::EnableText (
    unsigned int width,
    unsigned int height,
    HWND windowHandle )
```

Enables text.

#### 5.5.3.7 Execute()

```
void FARender::DeviceResources::Execute ( ) const
```

Executes the command list.

#### 5.5.3.8 FlushCommandQueue()

```
void FARender::DeviceResources::FlushCommandQueue ( )
```

Synchronizes the CPU and GPU. Use this function to make sure all of the commands in command list are executed by the GPU before the CPU writes in new commands.

#### 5.5.3.9 GetBackBufferFormat()

```
DXGI_FORMAT FARender::DeviceResources::GetBackBufferFormat ( ) const
```

Returns a constant reference to the back buffer format.

#### 5.5.3.10 GetCBVSize()

```
unsigned int FARender::DeviceResources::GetCBVSize ( ) const
```

The size of a constant buffer view.

#### 5.5.3.11 GetCommandList()

```
const Microsoft::WRL::ComPtr< ID3D12GraphicsCommandList > & FARender::DeviceResources::GetCommandList ( ) const
```

Returns a constant reference to the ID3D12GraphicsCommandList object.

#### 5.5.3.12 GetCurrentFrame()

```
unsigned int FARender::DeviceResources::GetCurrentFrame ( ) const
```

Returns the current frame.

#### 5.5.3.13 GetDepthStencilFormat()

```
DXGI_FORMAT FARender::DeviceResources::GetDepthStencilFormat ( ) const
```

Returns a constant reference to the depth stencil format.

#### 5.5.3.14 GetDevice()

```
const Microsoft::WRL::ComPtr< ID3D12Device > & FARender::DeviceResources::GetDevice ( ) const
```

Returns a constant reference to the ID3D12Device object.

#### 5.5.3.15 GetInstance()

```
static DeviceResources & FARender::DeviceResources::GetInstance (
    unsigned int width,
    unsigned int height,
    HWND windowHandle,
    unsigned int numFrames ) [static]
```

Call to make an object of [DeviceResources](#). This only allows one instance to exist.



#### 5.5.3.16 GetNumFrames()

```
unsigned int FAREnder::DeviceResources::GetNumFrames ( ) const
```

Returns the number of frames.

#### 5.5.3.17 GetTextResources()

```
const TextResources & FAREnder::DeviceResources::GetTextResources ( ) const
```

Returns a constant reference to the [TextResources](#) object.

#### 5.5.3.18 IsMSAAEnabled()

```
bool FAREnder::DeviceResources::IsMSAAEnabled ( ) const
```

Returns true if MSAA is enabled, false otherwise.

#### 5.5.3.19 IsTextEnabled()

```
bool FAREnder::DeviceResources::IsTextEnabled ( ) const
```

Returns true if text is enabled, false otherwise.

#### 5.5.3.20 NextFrame()

```
void FAREnder::DeviceResources::NextFrame ( )
```

Updates the current frame value to go to the next frame.

#### 5.5.3.21 Present()

```
void FAREnder::DeviceResources::Present ( )
```

Swaps the front and back buffers.

#### 5.5.3.22 Resize()

```
void FARender::DeviceResources::Resize (
    int width,
    int height,
    const HWND & handle )
```

Call when the window gets resized. Call when you initialize your program.

#### 5.5.3.23 RTBufferTransition()

```
void FARender::DeviceResources::RTBufferTransition ( )
```

Transitions the render target buffer.

#### 5.5.3.24 Signal()

```
void FARender::DeviceResources::Signal ( )
```

Adds an instruction to the GPU to set the fence value to the current fence value.

#### 5.5.3.25 UpdateCurrentFrameFenceValue()

```
void FARender::DeviceResources::UpdateCurrentFrameFenceValue ( )
```

Updates the current frames fence value.

#### 5.5.3.26 WaitForGPU()

```
void FARender::DeviceResources::WaitForGPU ( ) const
```

Waits for the GPU to execute all of the commands of the current frame. Signal should have been called before this function is called.

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

- [FADeviceResources.h](#)

## 5.6 DirectXException Class Reference

### Public Member Functions

- **DirectXException** (HRESULT hr, const std::wstring &functionName, const std::wstring &fileName, int line↵  
Number)
- std::wstring **ErrorMsg** () const

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

- [FADirectXException.h](#)

## 5.7 FARender::DrawSettings Struct Reference

Holds a array of objects that use the same PSO, root signature and primitive.

```
#include "FARenderScene.h"
```

### Public Attributes

- Microsoft::WRL::ComPtr< ID3D12PipelineState > **pipelineState**
- Microsoft::WRL::ComPtr< ID3D12RootSignature > **rootSig**
- D3D\_PRIMITIVE\_TOPOLOGY **prim**
- std::vector< FAShapes::DrawArguments > **drawArgs**

### 5.7.1 Detailed Description

Holds a array of objects that use the same PSO, root signature and primitive.

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

- [FARenderScene.h](#)

## 5.8 FARender::IndexBuffer Class Reference

This class stores indices in a Direct3D 12 default buffer.

```
#include "FABuffer.h"
```

## Public Member Functions

- **IndexBuffer** (const [IndexBuffer](#) &)=delete
- **IndexBuffer & operator=** (const [IndexBuffer](#) &)=delete
- const D3D12\_INDEX\_BUFFER\_VIEW & [GetIndexBufferView](#) ()  
*Returns a constant reference to the vertex buffer view.*
- void [CreateIndexBuffer](#) (const Microsoft::WRL::ComPtr< ID3D12Device > &device, const Microsoft::WRL::ComPtr< ID3D12GraphicsCommandList > &commandList, const void \*data, UINT numBytes)  
*Creates the vertex buffer and stores all of the specified vertices in the vertex buffer.*
- void [CreateIndexBufferView](#) (UINT numBytes, DXGI\_FORMAT format)  
*Creates the vertex buffer view and stores it.*

### 5.8.1 Detailed Description

This class stores indices in a Direct3D 12 default buffer.

### 5.8.2 Member Function Documentation

#### 5.8.2.1 CreateIndexBuffer()

```
void FARender::IndexBuffer::CreateIndexBuffer (
    const Microsoft::WRL::ComPtr< ID3D12Device > & device,
    const Microsoft::WRL::ComPtr< ID3D12GraphicsCommandList > & commandList,
    const void * data,
    UINT numBytes )
```

Creates the vertex buffer and stores all of the specified vertices in the vertex buffer.

#### 5.8.2.2 CreateIndexBufferView()

```
void FARender::IndexBuffer::CreateIndexBufferView (
    UINT numBytes,
    DXGI_FORMAT format )
```

Creates the vertex buffer view and stores it.

#### 5.8.2.3 GetIndexBufferView()

```
const D3D12_INDEX_BUFFER_VIEW & FARender::IndexBuffer::GetIndexBufferView ( )
```

Returns a constant reference to the vertex buffer view.

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

- [FABuffer.h](#)

## 5.9 FAREnder::MultiSampling Class Reference

A wrapper for multisampling resources. Uses DirectD 12 API.

```
#include "FAMultiSampling.h"
```

### Public Member Functions

- [MultiSampling](#) (const Microsoft::WRL::ComPtr< ID3D12Device > &device, DXGI\_FORMAT rtFormat, DXGI\_FORMAT dsFormat, unsigned int sampleCount)  
*Constructor. Checks if the specified format and sample count are supported by the specified device for multi-sampling. Throws a runtime\_error if they are not supported.*
- const Microsoft::WRL::ComPtr< ID3D12Resource > & [GetRenderTargetBuffer](#) ()  
*Returns the MSAA render target buffer.*
- DXGI\_FORMAT [GetRenderTargetFormat](#) ()
- DXGI\_FORMAT [GetDepthStencilFormat](#) ()
- void [ResetBuffers](#) ()  
*Resets the MSAA render target buffer and MSAA depth stencil buffer.*
- void [CreateRenderTargetBufferAndView](#) (const Microsoft::WRL::ComPtr< ID3D12Device > &device, const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > &rtvHeap, unsigned int indexWhereToStoreView, unsigned int rtvSize, unsigned int width, unsigned int height)  
*Creates the MSAA render target buffer and a view to it.*
- void [CreateDepthStencilBufferAndView](#) (const Microsoft::WRL::ComPtr< ID3D12Device > &device, const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > &dsvHeap, unsigned int indexWhereToStoreView, unsigned int dsvSize, unsigned int width, unsigned int height)  
*Creates the MSAA depth stencil buffer and a view to it.*
- void [Transition](#) (const Microsoft::WRL::ComPtr< ID3D12GraphicsCommandList > &commandList, D3D12\_RESOURCE\_STATES before, D3D12\_RESOURCE\_STATES after)  
*Transitions the MSAA render target buffer from the specified before state to the specified after state.*
- void [ClearRenderTargetBuffer](#) (const Microsoft::WRL::ComPtr< ID3D12GraphicsCommandList > &commandList, const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > &rtvHeap, unsigned int indexOfView, unsigned int rtvSize, const float \*clearValue)  
*Clears the MSAA render target buffer with the specified clear value.*
- void [ClearDepthStencilBuffer](#) (const Microsoft::WRL::ComPtr< ID3D12GraphicsCommandList > &commandList, const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > &dsvHeap, unsigned int indexOfView, unsigned int dsvSize, float clearValue)  
*Clears the MSAA depth stencil buffer with the specified clear value.*

### 5.9.1 Detailed Description

A wrapper for multisampling resources. Uses DirectD 12 API.

### 5.9.2 Constructor & Destructor Documentation

### 5.9.2.1 MultiSampling()

```
FARender::MultiSampling::MultiSampling (
    const Microsoft::WRL::ComPtr< ID3D12Device > & device,
    DXGI_FORMAT rtFormat,
    DXGI_FORMAT dsFormat,
    unsigned int sampleCount )
```

Constructor. Checks if the specified format and sample count are supported by the specified device for multi-sampling. Throws a runtime\_error if they are not supported.

## 5.9.3 Member Function Documentation

### 5.9.3.1 ClearDepthStencilBuffer()

```
void FAREnder::MultiSampling::ClearDepthStencilBuffer (
    const Microsoft::WRL::ComPtr< ID3D12GraphicsCommandList > & commandList,
    const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > & dsvHeap,
    unsigned int indexOfView,
    unsigned int dsvSize,
    float clearValue )
```

Clears the MSAA depth stencil buffer with the specified clear value.

### 5.9.3.2 ClearRenderTargetBuffer()

```
void FAREnder::MultiSampling::ClearRenderTargetBuffer (
    const Microsoft::WRL::ComPtr< ID3D12GraphicsCommandList > & commandList,
    const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > & rtvHeap,
    unsigned int indexOfView,
    unsigned int rtvSize,
    const float * clearValue )
```

Clears the MSAA render target buffer with the specified clear value.

### 5.9.3.3 CreateDepthStencilBufferAndView()

```
void FAREnder::MultiSampling::CreateDepthStencilBufferAndView (
    const Microsoft::WRL::ComPtr< ID3D12Device > & device,
    const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > & dsvHeap,
    unsigned int indexWhereToStoreView,
    unsigned int dsvSize,
    unsigned int width,
    unsigned int height )
```

Creates the MSAA depth stencil buffer and a view to it.

#### 5.9.3.4 CreateRenderTargetBufferAndView()

```
void FARender::MultiSampling::CreateRenderTargetBufferAndView (
    const Microsoft::WRL::ComPtr< ID3D12Device > & device,
    const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > & rtvHeap,
    unsigned int indexOfWhereToStoreView,
    unsigned int rtvSize,
    unsigned int width,
    unsigned int height )
```

Creates the MSAA render target buffer and a view to it.

#### 5.9.3.5 GetRenderTargetBuffer()

```
const Microsoft::WRL::ComPtr< ID3D12Resource > & FARender::MultiSampling::GetRenderTargetBuffer ( )
```

Returns the MSAA render target buffer.

#### 5.9.3.6 ResetBuffers()

```
void FARender::MultiSampling::ResetBuffers ( )
```

Resets the MSAA render target buffer and MSAA depth stencil buffer.

#### 5.9.3.7 Transition()

```
void FARender::MultiSampling::Transition (
    const Microsoft::WRL::ComPtr< ID3D12GraphicsCommandList > & commandList,
    D3D12_RESOURCE_STATES before,
    D3D12_RESOURCE_STATES after )
```

Transitions the MSAA render target buffer from the specified before state to the specified after state.

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

- FAMultiSampling.h

## 5.10 FARender::RenderScene Class Reference

This class is used to render a scene using Direct3D 12 API.

```
#include "FARenderScene.h"
```

## Public Member Functions

- **RenderScene** (unsigned int width, unsigned int height, HWND windowHandle)
- **RenderScene** (const [RenderScene](#) &)=delete
- **RenderScene & operator=** (const [RenderScene](#) &)=delete
- const [DeviceResources](#) & **GetDeviceResources** () const
- FAShapes::DrawArguments & **GetDrawArguments** (unsigned int drawSettingsIndex, unsigned int drawArgsIndex)
- const FAShapes::DrawArguments & **GetDrawArguments** (unsigned int drawSettingsIndex, unsigned int drawArgsIndex) const
- [FACamera::Camera](#) & **GetCamera** ()
- const [FACamera::Camera](#) & **GetCamera** () const
- [FARender::Text](#) & **GetText** (unsigned int textIndex)
- const [FARender::Text](#) & **GetText** (unsigned int textIndex) const
- void **LoadShader** (const std::wstring &filename)
- void **LoadShaderAndCompile** (const std::wstring &filename, const std::string &entryPointName, const std::string &target)
- void **RemoveShader** (unsigned int index)
- void **CreateInputElementDescription** (const char \*semanticName, unsigned int semanticIndex, DXGI\_FORMAT format, unsigned int inputSlot, unsigned int byteOffset, D3D12\_INPUT\_CLASSIFICATION inputSlotClassification=D3D12\_INPUT\_CLASSIFICATION\_PER\_VERTEX\_DATA, unsigned int instanceStepRate=0)
- void **RemoveInputElementDescription** (unsigned int index)
- void **CreatePSO** (unsigned int drawSettingsIndex, D3D12\_FILL\_MODE fillMode, BOOL enableMultisample, unsigned int vsIndex, unsigned int psIndex, const D3D12\_PRIMITIVE\_TOPOLOGY\_TYPE &primitiveType, UINT sampleCount)
- void **CreateRootSignature** (unsigned int drawSettingsIndex)
- void **CreateVertexBuffer** ()
- void [CreateIndexBuffer](#) ()  
*Creates an index buffer with the specified name and stores all of the added indices. Also creates a view to the index buffer.*  
*Execute commands and flush the command queue after calling createVertexBuffer() and createIndexBuffer().*
- void [CreateCBVHeap](#) (UINT numDescriptors, UINT shaderRegister)  
*Creates the CBV heap.*
- void [CreateConstantBuffer](#) (UINT numOfBytes)  
*Creates a constant buffer for each frame.*
- void [CreateConstantBufferView](#) (UINT index, UINT numBytes)  
*Creates a constant buffer view for each frame and stores it in the CBV heap.*
- void [SetPSO](#) (unsigned int drawSettingsIndex, const Microsoft::WRL::ComPtr< ID3D12PipelineState > &pso)  
*Sets the PSO in the specified [DrawSettings](#) structure to the specified pso. If the index to the specified [DrawSettings](#) structure is out of bounds an out\_of\_range exception is thrown.*
- void [SetRootSignature](#) (unsigned int drawSettingsIndex, const Microsoft::WRL::ComPtr< ID3D12RootSignature > &rootSignature)  
*Sets the root signature in the specified [DrawSettings](#) structure to the specified root signature. If the index to the specified [DrawSettings](#) structure is out of bounds an out\_of\_range exception is thrown.*
- void [SetPrimitive](#) (unsigned int drawSettingsIndex, const D3D\_PRIMITIVE\_TOPOLOGY &primitive)  
*Sets the Primitive in the specified [DrawSettings](#) structure to the specified primitive. If the index to the specified [DrawSettings](#) structure is out of bounds an out\_of\_range exception is thrown.*
- void [AddDrawArgument](#) (unsigned int drawSettingsIndex, const FAShapes::DrawArguments &drawArg)  
*Adds the specified draw argument structure to the DrawArguments vector of the specified [DrawSettings](#) structure. If the index to the specified [DrawSettings](#) structure is out of bounds an out\_of\_range exception is thrown.*
- void [CreateDrawArgument](#) (unsigned int drawSettingsIndex, unsigned int indexCount, unsigned int locationOfFirstIndex, int indexOfFirstVertex, int indexOfConstantData)



- Creates a DrawArgument structure with the specified values. The created DrawArgument structure is stored in the DrawArguments vector of the specified DrawSettings structure. If the index to the specified DrawSettings structure is out of bounds an out\_of\_range exception is thrown.*
- void [RemoveDrawArgument](#) (unsigned int drawSettingsIndex, unsigned int drawArgIndex)
 

*Removes the specified DrawArgument structure in the DrawArguments vector of the specified DrawSettings structure. If the index to the specified DrawSettings structure or if the index to the specified DrawArguments structure is out of bounds an out\_of\_range exception is thrown.*
  - void [CreateDrawSettings](#) ()
 

*Creates a DrawSettings.*
  - void [RemoveDrawSettings](#) (unsigned int drawSettingsIndex)
 

*Removes the specified DrawSettings structure. If the index to the specified DrawSettings structure is out of bounds an out\_of\_range exception is thrown.*
  - void [CreateText](#) (FAMath::Vector4D textLocation, const std::wstring &textString, float textSize, const FColor::Color textColor)
 

*Creates a Text object with the specified properties and stores it. For text location the first two values in the vector is the top-left location of the rectangle and the last two values are the bottom-right location of the rectangle.*
  - void [RemoveText](#) (unsigned int textIndex)
 

*Removes the specified Text object. If the index to the specified DrawSettings structure is out of bounds an out\_of\_range exception is thrown.*
  - void [AddVertices](#) (const std::vector< FAShapes::Vertex > &vertices)
 

*Adds the specified vertices to the vertex list.*
  - void [AddVertices](#) (const FAShapes::Vertex \*vertices, unsigned int numVertices)
 

*Adds the specified vertices to the vertex list.*
  - void [AddIndices](#) (const std::vector< unsigned int > &indices)
 

*Adds the specified vertices to the index list.*
  - void [AddIndices](#) (const unsigned int \*indices, unsigned int numIndices)
 

*Adds the specified vertices to the index list.*
  - void [BeforeDrawObjects](#) ()
 

*Puts all of the commands needed in the command list before drawing the objects of the scene. Call before calling the first drawObjects function.*
  - void [DrawObjects](#) (unsigned int drawSettingsIndex)
 

*Draws all of the objects that use the same PSO, root signature and primitive. Call in between a beforeDrawObjects function and a afterDrawObjects function.*
  - void [AfterDrawObjects](#) ()
 

*Transitions the render target buffer to the correct state and excutes commands.*
  - void [BeforeDrawText](#) ()
 

*Puts all of the commands needed in the command list before drawing the text of the scene. Call before calling the first RenderText function.*
  - void [RenderText](#) (unsigned int textIndex)
 

*Draws the specified Text object. Call in between a BeforeDrawText function and a AfterDrawText function.*
  - void [AfterDrawText](#) ()
 

*Transitions the render target buffer and executes all of the text drawing commands. Call after calling all the RenderText functions.*
  - void [AfterDraw](#) ()
 

*Presents and signals (puts a fence command in the command queue). Call after drawing all your objects and text.*
  - void [ExecuteAndFlush](#) ()
 

*Executes the commands to fill the vertex and index buffer with data and flushes the queue.*
  - void [NextFrame](#) ()
 

*Moves to next frame and waits for the GPU to finish executing the next frame's commands.*
  - void [Resize](#) (unsigned int width, unsigned int height, HWND windowHandle)
 

*Resizes the DeviceResources resources when the window gets resized.*

- void [CopyData](#) (UINT index, UINT byteSize, const void \*data, UINT64 numOfBytes)  
*Copies the specified data into the constant buffer.*
- bool [IsMSAAEnabled](#) () const  
*Returns true if MSAA is enabled, false otherwise.*
- void [DisableMSAA](#) (unsigned int width, unsigned int height, HWND windowHandle)  
*Disables MSAA.*
- void [EnableMSAA](#) (unsigned int width, unsigned int height, HWND windowHandle)  
*Enables MSAA.*
- bool [IsTextEnabled](#) () const  
*Returns true if text is enabled, false otherwise.*
- void [DisableText](#) (unsigned int width, unsigned int height, HWND windowHandle)  
*Disables text.*
- void [EnableText](#) (unsigned int width, unsigned int height, HWND windowHandle)  
*Enables text.*

### 5.10.1 Detailed Description

This class is used to render a scene using Direct3D 12 API.

### 5.10.2 Member Function Documentation

#### 5.10.2.1 AddDrawArgument()

```
void FARender::RenderScene::AddDrawArgument (
    unsigned int drawSettingsIndex,
    const FAShapes::DrawArguments & drawArg )
```

Adds the specified draw argument structure to the DrawArguments vector of the specified [DrawSettings](#) structure. If the index to the specified [DrawSettings](#) structure is out of bounds an out\_of\_range exception is thrown.

#### 5.10.2.2 AddIndices() [1/2]

```
void FARender::RenderScene::AddIndices (
    const std::vector< unsigned int > & indices )
```

Adds the specified vertices to the index list.

### 5.10.2.3 AddIndices() [2/2]

```
void FARender::RenderScene::AddIndices (
    const unsigned int * indices,
    unsigned int numIndices )
```

Adds the specified vertices to the index list.

### 5.10.2.4 AddVertices() [1/2]

```
void FARender::RenderScene::AddVertices (
    const FAShapes::Vertex * vertices,
    unsigned int numVertices )
```

Adds the specified vertices to the vertex list.

### 5.10.2.5 AddVertices() [2/2]

```
void FARender::RenderScene::AddVertices (
    const std::vector< FAShapes::Vertex > & vertices )
```

Adds the specified vertices to the vertex list.

### 5.10.2.6 AfterDraw()

```
void FARender::RenderScene::AfterDraw ( )
```

Presents and signals (puts a fence command in the command queue). Call after drawing all your objects and text.

### 5.10.2.7 AfterDrawObjects()

```
void FARender::RenderScene::AfterDrawObjects ( )
```

Transitions the render target buffer to the correct state and excutes commands.

### 5.10.2.8 AfterDrawText()

```
void FARender::RenderScene::AfterDrawText ( )
```

Transitions the render target buffer and executes all of the text drawing commands. Call after calling all the Render↔Text functions.

#### 5.10.2.9 BeforeDrawObjects()

```
void FARender::RenderScene::BeforeDrawObjects ( )
```

Puts all of the commands needed in the command list before drawing the objects of the scene. Call before calling the first drawObjects function.

#### 5.10.2.10 BeforeDrawText()

```
void FARender::RenderScene::BeforeDrawText ( )
```

Puts all of the commands needed in the command list before drawing the text of the scene. Call before calling the first RenderText function.

#### 5.10.2.11 CopyData()

```
void FARender::RenderScene::CopyData (
    UINT index,
    UINT byteSize,
    const void * data,
    UINT64 numOfBytes )
```

Copies the specified data into the constant buffer.

#### 5.10.2.12 CreateCBVHeap()

```
void FARender::RenderScene::CreateCBVHeap (
    UINT numDescriptors,
    UINT shaderRegister )
```

Creates the CBV heap.

#### 5.10.2.13 CreateConstantBuffer()

```
void FARender::RenderScene::CreateConstantBuffer (
    UINT numOfBytes )
```

Creates a constant buffer for each frame.

#### 5.10.2.14 CreateConstantBufferView()

```
void FAREnder::RenderScene::CreateConstantBufferView (
    UINT index,
    UINT numBytes )
```

Creates a constant buffer view for each frame and stores it in the CBV heap.

#### 5.10.2.15 CreateDrawArgument()

```
void FAREnder::RenderScene::CreateDrawArgument (
    unsigned int drawSettingsIndex,
    unsigned int indexCount,
    unsigned int locationOfFirstIndex,
    int indexOfFirstVertex,
    int indexOfConstantData )
```

Creates a DrawArgument structure with the specified values. The created DrawArgument structure is stored in the DrawArguments vector of the specified [DrawSettings](#) structure. If the index to the specified [DrawSettings](#) structure is out of bounds an out\_of\_range exception is thrown.

#### 5.10.2.16 CreateDrawSettings()

```
void FAREnder::RenderScene::CreateDrawSettings ( )
```

Creates a [DrawSettings](#).

#### 5.10.2.17 CreateIndexBuffer()

```
void FAREnder::RenderScene::CreateIndexBuffer ( )
```

Creates an index buffer with the specified name and stores all of the added indices. Also creates a view to the index buffer.

Execute commands and flush the command queue after calling createVertexBuffer() and createIndexBuffer().

#### 5.10.2.18 CreateText()

```
void FAREnder::RenderScene::CreateText (
    FAMath::Vector4D textLocation,
    const std::wstring & textString,
    float textSize,
    const FIColor::Color textColor )
```

Creates a [Text](#) object with the specified properties and stores it. For text location the first two values in the vector is the top-left location of the rectangle and the last two values are the bottom-right location of the rectangle.

#### 5.10.2.19 DisableMSAA()

```
void FARender::RenderScene::DisableMSAA (
    unsigned int width,
    unsigned int height,
    HWND windowHandle )
```

Disables MSAA.

#### 5.10.2.20 DisableText()

```
void FARender::RenderScene::DisableText (
    unsigned int width,
    unsigned int height,
    HWND windowHandle )
```

Disables text.

#### 5.10.2.21 DrawObjects()

```
void FARender::RenderScene::DrawObjects (
    unsigned int drawSettingsIndex )
```

Draws all of the objects that use the same PSO, root signature and primitive. Call in between a beforeDrawObjects function and a afterDrawObjects function.

.

Ex.

beforeDrawObjects()

drawObjects()

drawObjects()

afterDrawObjects()

Throws an out\_of\_range exception if the index of the specified [DrawSettings](#) structure is out of bounds.

#### 5.10.2.22 EnableMSAA()

```
void FARender::RenderScene::EnableMSAA (
    unsigned int width,
    unsigned int height,
    HWND windowHandle )
```

Enables MSAA.

### 5.10.2.23 EnableText()

```
void FARender::RenderScene::EnableText (
    unsigned int width,
    unsigned int height,
    HWND windowHandle )
```

Enables text.

### 5.10.2.24 ExecuteAndFlush()

```
void FARender::RenderScene::ExecuteAndFlush ( )
```

Executes the commands to fill the vertex and index buffer with data and flushes the queue.

### 5.10.2.25 IsMSAAEnabled()

```
bool FARender::RenderScene::IsMSAAEnabled ( ) const
```

Returns true if MSAA is enabled, false otherwise.

### 5.10.2.26 IsTextEnabled()

```
bool FARender::RenderScene::IsTextEnabled ( ) const
```

Returns true if text is enabled, false otherwise.

### 5.10.2.27 NextFrame()

```
void FARender::RenderScene::NextFrame ( )
```

Moves to next frame and waits for the GPU to finish executing the next frame's commands.

### 5.10.2.28 RemoveDrawArgument()

```
void FARender::RenderScene::RemoveDrawArgument (
    unsigned int drawSettingsIndex,
    unsigned int drawArgIndex )
```

Removes the specified DrawArgument structure in the DrawArguments vector of the specified [DrawSettings](#) structure. If the index to the specified [DrawSettings](#) structure or if the index to the specified DrawArguments structure is out of bounds an out\_of\_range exception is thrown.

#### 5.10.2.29 RemoveDrawSettings()

```
void FARender::RenderScene::RemoveDrawSettings (
    unsigned int drawSettingsIndex )
```

Removes the specified [DrawSettings](#) structure. If the index to the specified [DrawSettings](#) structure is out of bounds an `out_of_range` exception is thrown.

#### 5.10.2.30 RemoveText()

```
void FARender::RenderScene::RemoveText (
    unsigned int textIndex )
```

Removes the specified [Text](#) object. If the index to the specified [DrawSettings](#) structure is out of bounds an `out_of_range` exception is thrown.

#### 5.10.2.31 RenderText()

```
void FARender::RenderScene::RenderText (
    unsigned int textIndex )
```

Draws the specified [Text](#) object. Call in between a `BeforeDrawText` function and a `AfterDrawText` function.

.

Ex.

`beforeDrawText()`

`drawText()`

`drawText()`

`afterDrawText()`

Throws an `out_of_range` exception if the specified [Text](#) object does not exist.

#### 5.10.2.32 Resize()

```
void FARender::RenderScene::Resize (
    unsigned int width,
    unsigned int height,
    HWND windowHandle )
```

Resizes the [DeviceResources](#) resources when the window gets resized.

#### 5.10.2.33 SetPrimitive()

```
void FARender::RenderScene::SetPrimitive (
    unsigned int drawSettingsIndex,
    const D3D_PRIMITIVE_TOPOLOGY & primitive )
```

Sets the Primitive in the specified [DrawSettings](#) structure to the specified primitive. If the index to the specified [DrawSettings](#) structure is out of bounds an `out_of_range` exception is thrown.



### 5.10.2.34 SetPSO()

```
void FAREnder::RenderScene::SetPSO (
    unsigned int drawSettingsIndex,
    const Microsoft::WRL::ComPtr< ID3D12PipelineState > & pso )
```

Sets the PSO in the specified [DrawSettings](#) structure to the specified pso. If the index to the specified [DrawSettings](#) structure is out of bounds an out\_of\_range exception is thrown.

### 5.10.2.35 SetRootSignature()

```
void FAREnder::RenderScene::SetRootSignature (
    unsigned int drawSettingsIndex,
    const Microsoft::WRL::ComPtr< ID3D12RootSignature > & rootSignature )
```

Sets the root signature in the specified [DrawSettings](#) structure to the specified root signature. If the index to the specified [DrawSettings](#) structure is out of bounds an out\_of\_range exception is thrown.

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

- [FARenderScene.h](#)

## 5.11 FAREnder::RenderTargetBuffer Class Reference

A wrapper for render target buffer resources. Uses DirectD 12 API.

```
#include "FABuffer.h"
```

### Public Member Functions

- [RenderTargetBuffer](#) (DXGI\_FORMAT format=DXGI\_FORMAT\_R8G8B8A8\_UNORM)  
*Default Constructor.*
- DXGI\_FORMAT [GetRenderTargetFormat](#) () const  
*Returns the format of the render target buffer.*
- Microsoft::WRL::ComPtr< ID3D12Resource > & [GetRenderTargetBuffer](#) ()  
*Returns a reference to the render target buffer.*
- const Microsoft::WRL::ComPtr< ID3D12Resource > & [GetRenderTargetBuffer](#) () const  
*Returns a constant reference to the render target buffer.*
- void [CreateRenderTargetBufferAndView](#) (const Microsoft::WRL::ComPtr< ID3D12Device > &device, const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > &rtvHeap, unsigned int indexWhereToStoreView, unsigned int rtvSize, unsigned int width, unsigned int height, unsigned int sampleCount=1)  
*Creates the render target buffer and view.*
- void [ResetBuffer](#) ()  
*Resest the render target buffer.*
- void [ClearRenderTargetBuffer](#) (const Microsoft::WRL::ComPtr< ID3D12GraphicsCommandList > &commandList, const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > &rtvHeap, unsigned int indexWhereToStoreView, unsigned int rtvSize, const float \*clearValue)  
*Clears the render target buffer with the specified clear value.*

### 5.11.1 Detailed Description

A wrapper for render target buffer resources. Uses DirectD 12 API.

### 5.11.2 Constructor & Destructor Documentation

#### 5.11.2.1 RenderTargetBuffer()

```
FARender::RenderTargetBuffer::RenderTargetBuffer (
    DXGI_FORMAT format = DXGI_FORMAT_R8G8B8A8_UNORM )
```

Default Constructor.

### 5.11.3 Member Function Documentation

#### 5.11.3.1 ClearRenderTargetBuffer()

```
void FARender::RenderTargetBuffer::ClearRenderTargetBuffer (
    const Microsoft::WRL::ComPtr< ID3D12GraphicsCommandList > & commandList,
    const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > & rtvHeap,
    unsigned int indexOfView,
    unsigned int rtvSize,
    const float * clearValue )
```

Clears the render target buffer with the specified clear value.

#### 5.11.3.2 CreateRenderTargetBufferAndView()

```
void FARender::RenderTargetBuffer::CreateRenderTargetBufferAndView (
    const Microsoft::WRL::ComPtr< ID3D12Device > & device,
    const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > & rtvHeap,
    unsigned int indexWhereToStoreView,
    unsigned int rtvSize,
    unsigned int width,
    unsigned int height,
    unsigned int sampleCount = 1 )
```

Creates the render target buffer and view.

### 5.11.3.3 GetRenderTargetBuffer() [1/2]

```
Microsoft::WRL::ComPtr< ID3D12Resource > & FAREnder::RenderTargetBuffer::GetRenderTargetBuffer  
( )
```

Returns a reference to the render target buffer.

### 5.11.3.4 GetRenderTargetBuffer() [2/2]

```
const Microsoft::WRL::ComPtr< ID3D12Resource > & FAREnder::RenderTargetBuffer::GetRenderTargetBuffer  
TargetBuffer ( ) const
```

Returns a constant reference to the render target buffer.

### 5.11.3.5 GetRenderTargetFormat()

```
DXGI_FORMAT FAREnder::RenderTargetBuffer::GetRenderTargetFormat ( ) const
```

Returns the format of the render target buffer.

### 5.11.3.6 ResetBuffer()

```
void FAREnder::RenderTargetBuffer::ResetBuffer ( )
```

Resest the render target buffer.

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

- [FABuffer.h](#)

## 5.12 FAREnder::SwapChain Class Reference

A wrapper for swap chain resources. Uses DirectD 12 API and DXGI API.

```
#include "FASwapChain.h"
```

## Public Member Functions

- [SwapChain](#) (const Microsoft::WRL::ComPtr< IDXGIFactory4 > &dxgiFactory, const Microsoft::WRL::ComPtr< ID3D12CommandQueue > &commandQueue, HWND windowHandle, DXGI\_FORMAT rtFormat=DXGI\_FORMAT\_R8G8B8A8\_UNORM, DXGI\_FORMAT dsFormat=DXGI\_FORMAT\_D24\_UNORM\_S8\_UINT, unsigned int numRenderTargetBuffers=2)  
*Constructor. Creates a swap chain.*
- const [RenderTargetBuffer](#) \* [GetRenderTargetBuffers](#) () const  
*Returns a constant pointer to the render target buffers.*
- const Microsoft::WRL::ComPtr< ID3D12Resource > & [GetCurrentBackBuffer](#) () const  
*Returns a constant reference to the current render target buffer.*
- unsigned int [GetNumRenderTargetBuffers](#) () const  
*Returns the number of swap chain buffers.*
- unsigned int [GetCurrentBackBufferIndex](#) () const  
*Returns the current back buffer index.*
- DXGI\_FORMAT [GetBackBufferFormat](#) () const  
*Returns the format of the swap chain.*
- DXGI\_FORMAT [GetDepthStencilFormat](#) () const  
*Returns the format of the depth stencil buffer.*
- void [ResetBuffers](#) ()  
*The render target buffers no longer reference the swap chain buffers after this function is executed.*
- void [ResizeSwapChain](#) (unsigned width, unsigned height)  
*Resizes the swap chain.*
- void [CreateRenderTargetBuffersAndViews](#) (const Microsoft::WRL::ComPtr< ID3D12Device > &device, const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > &rtvHeap, unsigned int indexWhereToStoreFirstView, unsigned int rtvSize)  
*Creates the render target buffers and views to them.*
- void [CreateDepthStencilBufferAndView](#) (const Microsoft::WRL::ComPtr< ID3D12Device > &device, const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > &dsvHeap, unsigned int index, unsigned int dsvSize, unsigned int width, unsigned int height)  
*Creates the swap chains depth stencil buffer and view to it.*
- void [ClearCurrentBackBuffer](#) (const Microsoft::WRL::ComPtr< ID3D12GraphicsCommandList > &commandList, const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > &rtvHeap, unsigned int indexOfFirstView, unsigned int rtvSize, const float \*backBufferClearValue)  
*Clears the current render target buffer.*
- void [ClearDepthStencilBuffer](#) (const Microsoft::WRL::ComPtr< ID3D12GraphicsCommandList > &commandList, const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > &dsvHeap, unsigned int indexOfView, unsigned int dsvSize, float clearValue)  
*Clears the swap chains depth stencil buffer with the specified clear value.*
- void [Transition](#) (const Microsoft::WRL::ComPtr< ID3D12GraphicsCommandList > &commandList, D3D12\_RESOURCE\_STATES before, D3D12\_RESOURCE\_STATES after)  
*Transitions the current render target buffer from the specified before state to the specified after state.*
- void [Present](#) ()  
*Swaps the front and back buffers.*

### 5.12.1 Detailed Description

A wrapper for swap chain resources. Uses DirectD 12 API and DXGI API.

### 5.12.2 Constructor & Destructor Documentation

### 5.12.2.1 SwapChain()

```
FARender::SwapChain::SwapChain (
    const Microsoft::WRL::ComPtr< IDXGIFactory4 > & dxgiFactory,
    const Microsoft::WRL::ComPtr< ID3D12CommandQueue > & commandQueue,
    HWND windowHandle,
    DXGI_FORMAT rtFormat = DXGI_FORMAT_R8G8B8A8_UNORM,
    DXGI_FORMAT dsFormat = DXGI_FORMAT_D24_UNORM_S8_UINT,
    unsigned int numRenderTargetBuffers = 2 )
```

Constructor. Creates a swap chain.

## 5.12.3 Member Function Documentation

### 5.12.3.1 ClearCurrentBackBuffer()

```
void FAREnder::SwapChain::ClearCurrentBackBuffer (
    const Microsoft::WRL::ComPtr< ID3D12GraphicsCommandList > & commandList,
    const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > & rtvHeap,
    unsigned int indexOfFirstView,
    unsigned int rtvSize,
    const float * backBufferClearValue )
```

Clears the current render target buffer.

### 5.12.3.2 ClearDepthStencilBuffer()

```
void FAREnder::SwapChain::ClearDepthStencilBuffer (
    const Microsoft::WRL::ComPtr< ID3D12GraphicsCommandList > & commandList,
    const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > & dsvHeap,
    unsigned int indexOfView,
    unsigned int dsvSize,
    float clearValue )
```

Clears the swap chains depth stencil buffer with the specified clear value.

### 5.12.3.3 CreateDepthStencilBufferAndView()

```
void FAREnder::SwapChain::CreateDepthStencilBufferAndView (
    const Microsoft::WRL::ComPtr< ID3D12Device > & device,
    const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > & dsvHeap,
    unsigned int index,
    unsigned int dsvSize,
    unsigned int width,
    unsigned int height )
```

Creates the swap chains depth stencil buffer and view to it.

#### 5.12.3.4 CreateRenderTargetBuffersAndViews()

```
void FARender::SwapChain::CreateRenderTargetBuffersAndViews (
    const Microsoft::WRL::ComPtr< ID3D12Device > & device,
    const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > & rtvHeap,
    unsigned int indexOfWhereToStoreFirstView,
    unsigned int rtvSize )
```

Creates the render target buffers and views to them.

#### 5.12.3.5 GetBackBufferFormat()

```
DXGI_FORMAT FARender::SwapChain::GetBackBufferFormat ( ) const
```

Returns the format of the swap chain.

#### 5.12.3.6 GetCurrentBackBuffer()

```
const Microsoft::WRL::ComPtr< ID3D12Resource > & FARender::SwapChain::GetCurrentBackBuffer ( )
const
```

Returns a constant reference to the current render target buffer.

#### 5.12.3.7 GetCurrentBackBufferIndex()

```
unsigned int FARender::SwapChain::GetCurrentBackBufferIndex ( ) const
```

Returns the current back buffer index.

#### 5.12.3.8 GetDepthStencilFormat()

```
DXGI_FORMAT FARender::SwapChain::GetDepthStencilFormat ( ) const
```

Returns the format of the depth stencil buffer.

#### 5.12.3.9 GetNumRenderTargetBuffers()

```
unsigned int FARender::SwapChain::GetNumRenderTargetBuffers ( ) const
```

Returns the number of swap chain buffers.

### 5.12.3.10 GetRenderTargetBuffers()

```
const RenderTargetBuffer * FAREnder::SwapChain::GetRenderTargetBuffers ( ) const
```

Returns a constant pointer to the render target buffers.

### 5.12.3.11 Present()

```
void FAREnder::SwapChain::Present ( )
```

Swaps the front and back buffers.

### 5.12.3.12 ResetBuffers()

```
void FAREnder::SwapChain::ResetBuffers ( )
```

The render target buffers no longer reference the swap chain buffers after this function is executed.

### 5.12.3.13 ResizeSwapChain()

```
void FAREnder::SwapChain::ResizeSwapChain (
    unsigned width,
    unsigned height )
```

Resizes the swap chain.

### 5.12.3.14 Transition()

```
void FAREnder::SwapChain::Transition (
    const Microsoft::WRL::ComPtr< ID3D12GraphicsCommandList > & commandList,
    D3D12_RESOURCE_STATES before,
    D3D12_RESOURCE_STATES after )
```

Transitions the current render target buffer from the specified before state to the specified after state.

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

- FASwapChain.h

## 5.13 FAREnder::Text Class Reference

This class is used to help render text. Stores the location of the text, the text string, text size and the color of the text.

```
#include "FAText.h"
```

### Public Member Functions

- [Text](#) (const FAMath::Vector4D &textLocation, const std::wstring &textString, float textSize, const [FAColor::Color](#) &textColor)  
*Overloaded Constructor. Initializes the format of the text.  
For text location the first two values in the vector is the top-left location of the rectangle and the last two values are the bottom-right location of the rectangle.*
- const FAMath::Vector4D & [GetTextLocation](#) () const  
*Returns a constant reference to the text location.*
- const std::wstring & [GetTextString](#) () const  
*Returns a constant reference to the text string.*
- float [GetTextSize](#) () const  
*Returns the text size.*
- const [FAColor::Color](#) & [GetTextColor](#) () const  
*Returns a constant reference to the text color.*
- void [SetTextSize](#) (float textSize)  
*Changes the text size to the specified size.*
- void [SetTextColor](#) (const [FAColor::Color](#) &textColor)  
*Changes the text color to the specified color.*
- void [SetTextString](#) (const std::wstring &textString)  
*Changes the text string to the specified string.*
- void [SetTextLocation](#) (const FAMath::Vector4D &textLocation)  
*Changes the text location to the specified location.*

### 5.13.1 Detailed Description

This class is used to help render text. Stores the location of the text, the text string, text size and the color of the text.

### 5.13.2 Constructor & Destructor Documentation

#### 5.13.2.1 Text()

```
FARender::Text::Text (
    const FAMath::Vector4D & textLocation,
    const std::wstring & textString,
    float textSize,
    const FAColor::Color & textColor )
```

Overloaded Constructor. Initializes the format of the text.

For text location the first two values in the vector is the top-left location of the rectangle and the last two values are the bottom-right location of the rectangle.



### 5.13.3 Member Function Documentation

#### 5.13.3.1 GetTextColor()

```
const FColor::Color & FARender::Text::GetTextColor ( ) const
```

Returns a constant reference to the text color.

#### 5.13.3.2 GetTextLocation()

```
const FMath::Vector4D & FARender::Text::GetTextLocation ( ) const
```

Returns a constant reference to the text location.

#### 5.13.3.3 GetTextSize()

```
float FARender::Text::GetTextSize ( ) const
```

Returns the text size.

#### 5.13.3.4 GetTextString()

```
const std::wstring & FARender::Text::GetTextString ( ) const
```

Returns a constant reference to the text string.

#### 5.13.3.5 SetTextColor()

```
void FARender::Text::SetTextColor (
    const FColor::Color & textColor )
```

Changes the text color to the specified color.

### 5.13.3.6 SetTextLocation()

```
void FARender::Text::SetTextLocation (
    const FMath::Vector4D & textLocation )
```

Changes the text location to the specified location.

### 5.13.3.7 SetTextSize()

```
void FARender::Text::SetTextSize (
    float textSize )
```

Changes the text size to the specified size.

### 5.13.3.8 SetTextString()

```
void FARender::Text::SetTextString (
    const std::wstring & textString )
```

Changes the text string to the specified string.

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

- [FAText.h](#)

## 5.14 FARender::TextResources Class Reference

A wrapper for resources that are needed to render text using Direct3D 11on12, Direct2D and DirectWrite.

```
#include "FATextResources.h"
```

### Public Member Functions

- [TextResources](#) (const Microsoft::WRL::ComPtr< ID3D12Device > &device, const Microsoft::WRL::ComPtr< ID3D12CommandQueue > &commandQueue, unsigned int numSwapChainBuffers)  
*Constructor. Initializes the text resources.*
- const Microsoft::WRL::ComPtr< ID2D1DeviceContext > & [GetDirect2DDeviceContext](#) () const  
*Returns a constant reference to the direct 2D device context.*
- const Microsoft::WRL::ComPtr< IDWriteFactory > & [GetDirectWriteFactory](#) () const  
*Returns a constant reference to the direct direct write factory.*
- void [ResetBuffers](#) ()  
*Resets the text buffers.*
- void [ResizeBuffers](#) (const [RenderTargetBuffer](#) \*renderTargetBuffers, HWND windowHandle)  
*Resizes the buffers.*
- void [BeforeRenderText](#) (unsigned int currentBackBuffer)  
*Prepares to render text.*
- void [AfterRenderText](#) (unsigned int currentBackBuffer)  
*Executes text commands.*

### 5.14.1 Detailed Description

A wrapper for resources that are needed to render text using Direct3D 11on12, Direct2D and DirectWrite.

### 5.14.2 Constructor & Destructor Documentation

#### 5.14.2.1 TextResources()

```
FARender::TextResources::TextResources (
    const Microsoft::WRL::ComPtr< ID3D12Device > & device,
    const Microsoft::WRL::ComPtr< ID3D12CommandQueue > & commandQueue,
    unsigned int numSwapChainBuffers )
```

Constructor. Initializes the text resources.

### 5.14.3 Member Function Documentation

#### 5.14.3.1 AfterRenderText()

```
void FAREnder::TextResources::AfterRenderText (
    unsigned int currentBackBuffer )
```

Executes text commands.

#### 5.14.3.2 BeforeRenderText()

```
void FAREnder::TextResources::BeforeRenderText (
    unsigned int currentBackBuffer )
```

Prepares to render text.

#### 5.14.3.3 GetDirect2DDeviceContext()

```
const Microsoft::WRL::ComPtr< ID2D1DeviceContext > & FAREnder::TextResources::GetDirect2↔
DDeviceContext ( ) const
```

Returns a constant reference to the direct 2D device context.

#### 5.14.3.4 GetDirectWriteFactory()

```
const Microsoft::WRL::ComPtr< IDWriteFactory > & FAREnder::TextResources::GetDirectWriteFactory ( ) const
```

Returns a constant reference to the direct direct write factory.

#### 5.14.3.5 ResetBuffers()

```
void FAREnder::TextResources::ResetBuffers ( )
```

Resets the text buffers.

#### 5.14.3.6 ResizeBuffers()

```
void FAREnder::TextResources::ResizeBuffers (
    const RenderTargetBuffer * renderTargetBuffers,
    HWND windowHandle )
```

Resizes the buffers.

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

- FATextResources.h

## 5.15 FATime::Time Class Reference

### Public Member Functions

- [Time](#) ()  
*Default Constructor. Gets and stores the seconds per count.*
- void [Tick](#) ()  
*Stores the difference between the current time and the previous time.*
- float [DeltaTime](#) () const  
*Returns the difference between the current time and the previous time.*
- void [Reset](#) ()  
*Resets all time variables.*
- void [Stop](#) ()  
*Stops the timer.*
- void [Start](#) ()  
*Starts the timer.*
- float [TotalTime](#) () const  
*Returns how much time has passed since [Reset\(\)](#) was called. Does not count any pause time.*

## 5.15.1 Constructor & Destructor Documentation

### 5.15.1.1 Time()

```
FATime::Time::Time ( )
```

Default Constructor. Gets and stores the seconds per count.

## 5.15.2 Member Function Documentation

### 5.15.2.1 DeltaTime()

```
float FATime::Time::DeltaTime ( ) const
```

Returns the difference between the current time and the previous time.

### 5.15.2.2 Reset()

```
void FATime::Time::Reset ( )
```

Resets all time variables.

### 5.15.2.3 Start()

```
void FATime::Time::Start ( )
```

Starts the timer.

### 5.15.2.4 Stop()

```
void FATime::Time::Stop ( )
```

Stops the timer.

### 5.15.2.5 Tick()

```
void FTime::Time::Tick ( )
```

Stores the difference between the current time and the previous time.

### 5.15.2.6 TotalTime()

```
float FTime::Time::TotalTime ( ) const
```

Returns how much time has passed since [Reset\(\)](#) was called. Does not count any pause time.

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

- [FTime.h](#)

## 5.16 Time Class Reference

This class is used to get the time between each frame. You can stop start, reset and get the total time.

```
#include "FTime.h"
```

### 5.16.1 Detailed Description

This class is used to get the time between each frame. You can stop start, reset and get the total time.

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

- [FTime.h](#)

## 5.17 FARender::VertexBuffer Class Reference

This class stores vertices in a Direct3D 12 default buffer.

```
#include "FABuffer.h"
```

### Public Member Functions

- **VertexBuffer** (const [VertexBuffer](#) &)=delete
- **VertexBuffer & operator=** (const [VertexBuffer](#) &)=delete
- void [CreateVertexBuffer](#) (const Microsoft::WRL::ComPtr< ID3D12Device > &device, const Microsoft::WRL::ComPtr< ID3D12GraphicsCommandList > &commandList, const void \*data, UINT numBytes)  
*Creates the vertex buffer and stores all of the specified vertices in the vertex buffer.*
- void [CreateVertexBufferView](#) (UINT numBytes, UINT stride)  
*Creates the vertex buffer view and stores it.*
- const D3D12\_VERTEX\_BUFFER\_VIEW & [GetVertexBufferView](#) ()  
*Returns a constant reference to the vertex buffer view.*

### 5.17.1 Detailed Description

This class stores vertices in a Direct3D 12 default buffer.

### 5.17.2 Member Function Documentation

#### 5.17.2.1 CreateVertexBuffer()

```
void FARender::VertexBuffer::CreateVertexBuffer (
    const Microsoft::WRL::ComPtr< ID3D12Device > & device,
    const Microsoft::WRL::ComPtr< ID3D12GraphicsCommandList > & commandList,
    const void * data,
    UINT numBytes )
```

Creates the vertex buffer and stores all of the specified vertices in the vertex buffer.

#### 5.17.2.2 CreateVertexBufferView()

```
void FARender::VertexBuffer::CreateVertexBufferView (
    UINT numBytes,
    UINT stride )
```

Creates the vertex buffer view and stores it.

#### 5.17.2.3 GetVertexBufferView()

```
const D3D12_VERTEX_BUFFER_VIEW & FARender::VertexBuffer::GetVertexBufferView ( )
```

Returns a constant reference to the vertex buffer view.

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

- [FABuffer.h](#)

## 5.18 FAWindow::Window Class Reference

The window class is used to make a [Window](#) using Windows API.

```
#include "FAWindow.h"
```

## Public Member Functions

- [Window](#) (const HINSTANCE &hInstance, const std::wstring &windowClassName, const std::wstring &windowName, WNDPROC winProcFunction, unsigned int width, unsigned int height, void \*additionalData=nullptr)  
*Creates and displays a window. Registers a default window class with the OS with the specified instance, class name and window procedure.*
- [Window](#) (const HINSTANCE &hInstance, const WNDCLASSEX &windowClass, const std::wstring &windowName, unsigned int width, unsigned int height, void \*additionalData=nullptr)  
*Creates and displays a window. Registers the specified window class with the OS.*
- HWND [GetWindowHandle](#) () const  
*Returns the window handle.*
- unsigned int [GetWidth](#) () const  
*Returns the width of the window.*
- unsigned int [GetHeight](#) () const  
*Returns the height of the window.*
- void [SetWidth](#) (unsigned int width)  
*Sets the width of the window to the specified width.*
- void [SetHeight](#) (unsigned int height)  
*Sets the height of the window to the specified height.*

### 5.18.1 Detailed Description

The window class is used to make a [Window](#) using Windows API.

### 5.18.2 Constructor & Destructor Documentation

#### 5.18.2.1 Window() [1/2]

```
FAWindow::Window::Window (
    const HINSTANCE & hInstance,
    const std::wstring & windowClassName,
    const std::wstring & windowName,
    WNDPROC winProcFunction,
    unsigned int width,
    unsigned int height,
    void * additionalData = nullptr )
```

Creates and displays a window. Registers a default window class with the OS with the specified instance, class name and window procedure.



### 5.18.2.2 Window() [2/2]

```
FAWindow::Window::Window (
    const HINSTANCE & hInstance,
    const WNDCLASSEX & windowClass,
    const std::wstring & windowName,
    unsigned int width,
    unsigned int height,
    void * additionalData = nullptr )
```

Creates and displays a window. Registers the specified window class with the OS.

## 5.18.3 Member Function Documentation

### 5.18.3.1 GetHeight()

```
unsigned int FAWindow::Window::GetHeight ( ) const
```

Returns the height of the window.

### 5.18.3.2 GetWidth()

```
unsigned int FAWindow::Window::GetWidth ( ) const
```

Returns the width of the window.

### 5.18.3.3 GetWindowHandle()

```
HWND FAWindow::Window::GetWindowHandle ( ) const
```

Returns the window handle.

### 5.18.3.4 SetHeight()

```
void FAWindow::Window::SetHeight (
    unsigned int height )
```

Sets the height of the window o the specified height.

### 5.18.3.5 SetWidth()

```
void FAWindow::Window::SetWidth (
    unsigned int width )
```

Sets the width of the window to the specified width.

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

- [FAWindow.h](#)



## Chapter 6

# File Documentation

### 6.1 Direct3DLink.h

```
1 #pragma once
2
3 //Link necessary libraries.
4 #pragma comment(lib, "D3D12.lib")
5 #pragma comment(lib, "dxgi.lib")
6 #pragma comment(lib, "dxguid.lib")
7 #pragma comment(lib, "d3dcompiler.lib")
8 #pragma comment(lib, "D3D11.lib")
9 #pragma comment(lib, "D2D1.lib")
10 #pragma comment(lib, "DWrite.lib")
```

### 6.2 FABuffer.h File Reference

File has classes VertexBuffer, IndexBuffer and ConstantBuffer under namespace [FARender](#).

```
#include <wrl.h>
#include <d3d12.h>
```

#### Classes

- class [FARender::RenderTargetBuffer](#)  
*A wrapper for render target buffer resources. Uses DirectD 12 API.*
- class [FARender::DepthStencilBuffer](#)  
*A wrapper for depth stencil buffer resources. Uses DirectD 12 API.*
- class [FARender::VertexBuffer](#)  
*This class stores vertices in a Direct3D 12 default buffer.*
- class [FARender::IndexBuffer](#)  
*This class stores indices in a Direct3D 12 default buffer.*
- class [FARender::ConstantBuffer](#)  
*This class stores constant data in a Direct3D 12 upload buffer.*

#### Namespaces

- namespace [FARender](#)  
*Has classes that are used for rendering objects and text through the Direct3D 12 API.*

## 6.2.1 Detailed Description

File has classes VertexBuffer, IndexBuffer and ConstantBuffer under namespace [FARender](#).

## 6.3 FABuffer.h

[Go to the documentation of this file.](#)

```

1  #pragma once
2
3  #include <wrl.h>
4  #include <d3d12.h>
5
6  namespace FAREnder
7  {
8
9      class RenderTargetBuffer
10     {
11     public:
12         RenderTargetBuffer(DXGI_FORMAT format = DXGI_FORMAT_R8G8B8A8_UNORM);
13
14         DXGI_FORMAT GetRenderTargetFormat() const;
15
16         Microsoft::WRL::ComPtr<ID3D12Resource>& GetRenderTargetBuffer();
17
18         const Microsoft::WRL::ComPtr<ID3D12Resource>& GetRenderTargetBuffer() const;
19
20         void CreateRenderTargetBufferAndView(const Microsoft::WRL::ComPtr<ID3D12Device>& device,
21             const Microsoft::WRL::ComPtr<ID3D12DescriptorHeap>& rtvHeap, unsigned int
22             indexWhereToStoreView, unsigned int rtvSize,
23             unsigned int width, unsigned int height, unsigned int sampleCount = 1);
24
25         void ResetBuffer();
26
27         void ClearRenderTargetBuffer(const Microsoft::WRL::ComPtr<ID3D12GraphicsCommandList>&
28             commandList,
29             const Microsoft::WRL::ComPtr<ID3D12DescriptorHeap>& rtvHeap, unsigned int indexWhereToStoreView,
30             unsigned int rtvSize,
31             const float* clearValue);
32
33     private:
34         Microsoft::WRL::ComPtr<ID3D12Resource> mRenderTargetBuffer;
35         DXGI_FORMAT mRenderTargetFormat;
36
37     };
38
39     class DepthStencilBuffer
40     {
41     public:
42         DepthStencilBuffer(DXGI_FORMAT format = DXGI_FORMAT_D24_UNORM_S8_UINT);
43
44         DXGI_FORMAT GetDepthStencilFormat() const;
45
46         void CreateDepthStencilBufferAndView(const Microsoft::WRL::ComPtr<ID3D12Device>& device,
47             const Microsoft::WRL::ComPtr<ID3D12DescriptorHeap>& dsvHeap, unsigned int
48             indexWhereToStoreView, unsigned int dsvSize,
49             unsigned int width, unsigned int height, unsigned int sampleCount = 1);
50
51         void ResetBuffer();
52
53         void ClearDepthStencilBuffer(const Microsoft::WRL::ComPtr<ID3D12GraphicsCommandList>&
54             commandList,
55             const Microsoft::WRL::ComPtr<ID3D12DescriptorHeap>& dsvHeap, unsigned int indexWhereToStoreView,
56             unsigned int dsvSize,
57             float clearValue);
58
59     private:
60         Microsoft::WRL::ComPtr<ID3D12Resource> mDepthStencilBuffer;
61         DXGI_FORMAT mDepthStencilFormat;
62
63     };
64
65     class VertexBuffer
66     {
67     public:
68         VertexBuffer() = default;
69         VertexBuffer(const VertexBuffer&) = delete;
70         VertexBuffer& operator=(const VertexBuffer&) = delete;
71
72         void CreateVertexBuffer(const Microsoft::WRL::ComPtr<ID3D12Device>& device,

```

```

110         const Microsoft::WRL::ComPtr<ID3D12GraphicsCommandList>& commandList, const void* data, UINT
numBytes);
111
112     void CreateVertexBufferView(UINT numBytes, UINT stride);
113
114     const D3D12_VERTEX_BUFFER_VIEW& GetVertexBufferView();
115
116 private:
117     Microsoft::WRL::ComPtr<ID3D12Resource> mVertexDefaultBuffer;
118     Microsoft::WRL::ComPtr<ID3D12Resource> mVertexUploadBuffer;
119     D3D12_VERTEX_BUFFER_VIEW mVertexBufferView{};
120 };
121
122 class IndexBuffer
123 {
124 public:
125     IndexBuffer() = default;
126     IndexBuffer(const IndexBuffer&) = delete;
127     IndexBuffer& operator=(const IndexBuffer&) = delete;
128
129     const D3D12_INDEX_BUFFER_VIEW& GetIndexBufferView();
130
131     void CreateIndexBuffer(const Microsoft::WRL::ComPtr<ID3D12Device>& device,
const Microsoft::WRL::ComPtr<ID3D12GraphicsCommandList>& commandList, const void* data, UINT
numBytes);
132
133     void CreateIndexBufferView(UINT numBytes, DXGI_FORMAT format);
134
135 private:
136     Microsoft::WRL::ComPtr<ID3D12Resource> mIndexDefaultBuffer;
137     Microsoft::WRL::ComPtr<ID3D12Resource> mIndexUploadBuffer;
138     D3D12_INDEX_BUFFER_VIEW mIndexBufferView{};
139 };
140
141 class ConstantBuffer
142 {
143 public:
144     ConstantBuffer() = default;
145
146     ConstantBuffer(const ConstantBuffer&) = delete;
147     ConstantBuffer& operator=(const ConstantBuffer&) = delete;
148
149     ~ConstantBuffer();
150
151     void CreateConstantBuffer(const Microsoft::WRL::ComPtr<ID3D12Device>& device, const UINT&
numOfBytes);
152
153     void CreateConstantBufferView(const Microsoft::WRL::ComPtr<ID3D12Device>& device,
const Microsoft::WRL::ComPtr<ID3D12DescriptorHeap>& cbvHeap, UINT cbvSize, UINT
cBufferIndex,
154     UINT cbvHeapIndex, UINT numBytes);
155
156     void CopyData(UINT index, UINT byteSize, const void* data, UINT64 numOfBytes);
157
158 private:
159     Microsoft::WRL::ComPtr<ID3D12Resource> mConstantBuffer;
160     BYTE* mMappedData{ nullptr };
161 };
162 }

```

## 6.4 FACamera.h File Reference

File that has namespace [FACamera](#). Withn the namespace is the class [Camera](#).

```

#include "FAMathEngine.h"
#include <Windows.h>

```

### Classes

- class [FACamera::Camera](#)

*Simple first person style camera class that lets the viewer explore the 3D scene.*

*It keeps track of the camera coordinate system relative to the world space so that the view matrix can be constructed.*

*It keeps track of the viewing frustum of the camera so that the projection matrix can be obtained.*

.

## Namespaces

- namespace [FACamera](#)  
*Has [Camera](#) class.*

## Typedefs

- typedef FAMath::Vector2D [vec2](#)
- typedef FAMath::Vector3D [vec3](#)
- typedef FAMath::Vector4D [vec4](#)
- typedef FAMath::Matrix4x4 [mat4](#)

### 6.4.1 Detailed Description

File that has namespace [FACamera](#). Withn the namespace is the class Camera.

### 6.4.2 Typedef Documentation

#### 6.4.2.1 vec2

```
typedef FAMath::Vector2D vec2
```

FACAMERA\_H FILE

## 6.5 FACamera.h

[Go to the documentation of this file.](#)

```
1 #pragma once
2
12 #include "FAMathEngine.h"
13 #include <Windows.h>
14
15 typedef FAMath::Vector2D vec2;
16 typedef FAMath::Vector3D vec3;
17 typedef FAMath::Vector4D vec4;
18 typedef FAMath::Matrix4x4 mat4;
19
23 namespace FACamera
24 {
30     class Camera
31     {
32     public:
44         Camera(vec3 cameraPosition = vec3(0.0f, 0.0f, 0.0f),
45             vec3 x = vec3(1.0f, 0.0f, 0.0f), vec3 y = vec3(0.0f, 1.0f, 0.0f), vec3 z = vec3(0.0f, 0.0f,
1.0f),
46             float znear = 1.0f, float zfar = 100.f, float aspectRatio = 1.0f, float vFov = 45.0f,
47             float cameraVelocity = 10.0f, float angularVelocity = 0.25f);
48
51         const vec3& GetCameraPosition() const;
52
55         const vec3& GetX() const;
56
59         const vec3& GetY() const;
60
63         const vec3& GetZ() const;
64
```

```

67     const mat4& GetViewTransformationMatrix() const;
68
71     float GetCameraVelocity() const;
72
75     float GetAngularVelocity() const;
76
79     void LookAt(vec3 cameraPosition, vec3 target, vec3 up);
80
83     float GetZNear() const;
84
87     float GetZFar() const;
88
91     float GetVerticalFov() const;
92
95     float GetAspectRatio() const;
96
99     void SetCameraPosition(const vec3& position);
100
103     void SetX(const vec3& x);
104
107     void SetY(const vec3& y);
108
111     void SetZ(const vec3& z);
112
115     void SetCameraVelocity(float velocity);
116
119     void SetAngularVelocity(float velocity);
120
123     void SetZNear(float znear);
124
127     void SetZFar(float zfar);
128
131     void SetVerticalFov(float fov);
132
135     void SetAspectRatio(float ar);
136
139     const mat4& GetPerspectiveProjectionMatrix() const;
140
143     const mat4& GetViewPerspectiveProjectionMatrix() const;
144
147     void UpdateViewMatrix();
148
151     void UpdatePerspectiveProjectionMatrix();
152
156     void UpdateViewPerspectiveProjectionMatrix();
157
160     void Left(float dt);
161
164     void Right(float dt);
165
168     void Forward(float dt);
169
172     void Backward(float dt);
173
176     void Up(float dt);
177
180     void Down(float dt);
181
184     void RotateCameraLeftRight(float xDiff);
185
188     void RotateCameraUpDown(float yDiff);
189
195     void KeyboardInput(float dt);
196
199     void MouseInput();
200
201 private:
202     //camera position in world coordinates
203     vec3 mCameraPosition;
204
205     //z-axis of the camera coordinate system
206     vec3 mN;
207
208     //y-axis of the camera coordinate system
209     vec3 mV;
210
211     //x-axis of the camera coordinate system
212     vec3 mU;
213
214     //stores the world to camera transform
215     mat4 mViewMatrix;
216
217     //frustrum properties
218     float mNear;
219     float mFar;
220     float mVerticalFov;
221     float mAspectRatio;

```

```

222         mat4 mPerspectiveProjectionMatrix;
223
224         mat4 mViewPerspectiveProjectionMatrix;
225
226         float mCameraVelocity;
227         float mAngularVelocity;
228
229         vec2 mLastMousePosition;
230     };
231 }

```

## 6.6 FAColor.h File Reference

File has class Color under namespace FAColor.

```
#include "FAMathEngine.h"
```

### Classes

- class [FAColor::Color](#)

*This class stores a RGBA color in a 4D vector using floats. The range of each component is [0.0, 1.0]. The first componet is red, second component is green, third component is blue and the 4th component is alpha.*

### Functions

- Color [FAColor::operator+](#) (const Color &c1, const Color &c2)  
*Returns the result of  $c1 + c2$ . Does component-wise addition. If any of the resultant components are  $> 1.0f$ , they are set to  $1.0f$ .*
- Color [FAColor::operator-](#) (const Color &c1, const Color &c2)  
*Returns the result of  $c1 - c2$ . Does component-wise subtraction. If any of the resultant components are  $< 0.0f$ , they are set to  $0.0f$ .*
- Color [FAColor::operator\\*](#) (const Color &c, float k)  
*Returns the result of  $c * k$ . If  $k < 0.0f$ , no multiplication happens and [Color](#)  $c$  is returned. If any of the resultant components are  $> 1.0f$ , they are set to  $1.0f$ .*
- Color [FAColor::operator\\*](#) (float k, const Color &c)  
*Returns the result of  $k * c$ . If  $k < 0.0f$ , no multiplication happens and [Color](#)  $c$  is returned. If any of the resultant components are  $> 1.0f$ , they are set to  $1.0f$ .*
- Color [FAColor::operator\\*](#) (const Color &c1, const Color &c2)  
*Returns the result of  $c1 * c2$ . If any of the resultant components are  $> 1.0f$ , they are set to  $1.0f$ .*

#### 6.6.1 Detailed Description

File has class Color under namespace FAColor.

#### 6.6.2 Function Documentation



### 6.6.2.1 operator\*() [1/3]

```
Color FColor::operator* (
    const Color & c,
    float k )
```

Returns the result of  $c * k$ . If  $k < 0.0f$ , no multiplication happens and Color  $c$  is returned. If any of the resultant components are  $> 1.0f$ , they are set to  $1.0f$ .

.

### 6.6.2.2 operator\*() [2/3]

```
Color FColor::operator* (
    const Color & c1,
    const Color & c2 )
```

Returns the result of  $c1 * c2$ . If any of the resultant components are  $> 1.0f$ , they are set to  $1.0f$ .

.

### 6.6.2.3 operator\*() [3/3]

```
Color FColor::operator* (
    float k,
    const Color & c )
```

Returns the result of  $k * c$ . If  $k < 0.0f$ , no multiplication happens and Color  $c$  is returned. If any of the resultant components are  $> 1.0f$ , they are set to  $1.0f$ .

.

### 6.6.2.4 operator+()

```
Color FColor::operator+ (
    const Color & c1,
    const Color & c2 )
```

Returns the result of  $c1 + c2$ . Does component-wise addition. If any of the resultant components are  $> 1.0f$ , they are set to  $1.0f$ .

### 6.6.2.5 operator-()

```
Color FColor::operator- (
    const Color & c1,
    const Color & c2 )
```

Returns the result of  $c1 - c2$ . Does component-wise subtraction. If any of the resultant components are  $< 0.0f$ , they are set to  $0.0f$ .

## 6.7 FIColor.h

[Go to the documentation of this file.](#)

```

1  #pragma once
2
3  #include "FAMathEngine.h"
4
5  namespace FIColor
6  {
7      class Color
8      {
9      public:
10
11          Color(float r = 0.0f, float g = 0.0f, float b = 0.0f, float a = 1.0f);
12
13          Color(const FAMath::Vector4D& color);
14
15          const FAMath::Vector4D& GetColor() const;
16
17          float GetRed() const;
18
19          float GetGreen() const;
20
21          float GetBlue() const;
22
23          float GetAlpha() const;
24
25          void SetColor(const FAMath::Vector4D& color);
26
27          void SetRed(float r);
28
29          void SetGreen(float g);
30
31          void SetBlue(float b);
32
33          void SetAlpha(float a);
34
35          Color& operator+=(const Color& c);
36
37          Color& operator-=(const Color& c);
38
39          Color& operator*=(float k);
40
41          Color& operator*=(const Color& c);
42
43      private:
44          FAMath::Vector4D mColor;
45      };
46
47      Color operator+(const Color& c1, const Color& c2);
48
49      Color operator-(const Color& c1, const Color& c2);
50
51      Color operator*(const Color& c, float k);
52
53      Color operator*(float k, const Color& c);
54
55      Color operator*(const Color& c1, const Color& c2);
56  }

```

## 6.8 FADeviceResources.h File Reference

File has class DeviceResources under namespace [FARender](#).

```

#include <wrl.h>
#include <d3d12.h>
#include <dxgil_4.h>
#include "FASwapChain.h"
#include "FAMultiSampling.h"
#include "FATextResources.h"

```

## Classes

- class [FARender::DeviceResources](#)

*A wrapper for resources that are needed to render objects and text using the Direct3D 12 API.*

## Namespaces

- namespace [FARender](#)

*Has classes that are used for rendering objects and text through the Direct3D 12 API.*

### 6.8.1 Detailed Description

File has class DeviceResources under namespace [FARender](#).

## 6.9 FADeviceResources.h

[Go to the documentation of this file.](#)

```

1 #pragma once
2
3 #include <wrl.h>
4 #include <d3d12.h>
5 #include <dxgi1_4.h>
6 #include "FASwapChain.h"
7 #include "FAMultiSampling.h"
8 #include "FATextResources.h"
9
10 namespace FARender
11 {
12     class DeviceResources
13     {
14     public:
15
16         static DeviceResources& GetInstance(unsigned int width, unsigned int height, HWND windowHandle,
17         unsigned int numFrames);
18
19         DeviceResources(const DeviceResources&) = delete;
20         DeviceResources& operator=(const DeviceResources&) = delete;
21
22         ~DeviceResources();
23
24         const Microsoft::WRL::ComPtr<ID3D12Device>& GetDevice() const;
25
26         const Microsoft::WRL::ComPtr<ID3D12GraphicsCommandList>& GetCommandList() const;
27
28         DXGI_FORMAT GetBackBufferFormat() const;
29
30         DXGI_FORMAT GetDepthStencilFormat() const;
31
32         unsigned int GetCBVSize() const;
33
34         unsigned int GetNumFrames() const;
35
36         unsigned int GetCurrentFrame() const;
37
38         const TextResources& GetTextResources() const;
39
40         bool IsMSAAEnabled() const;
41
42         void DisableMSAA(unsigned int width, unsigned int height, HWND windowHandle);
43
44         void EnableMSAA(unsigned int width, unsigned int height, HWND windowHandle);
45
46         bool IsTextEnabled() const;
47
48         void DisableText(unsigned int width, unsigned int height, HWND windowHandle);
49
50         void EnableText(unsigned int width, unsigned int height, HWND windowHandle);
51
52         void UpdateCurrentFrameFenceValue();
53     };
54 }

```

```

99         void FlushCommandQueue();
100
101         void WaitForGPU() const;
102
103         void Signal();
104
105         void Resize(int width, int height, const HWND& handle);
106
107         void RTBufferTransition();
108
109         void BeforeTextDraw();
110
111         void AfterTextDraw();
112
113         void Execute() const;
114
115         void Present();
116
117         /*@brief Calls the necessary functions to let the user draw their objects.
118        */
119         void Draw();
120
121         void NextFrame();
122
123     private:
124
125         DeviceResources(unsigned int width, unsigned int height, HWND windowHandle, unsigned int
126         numFrames);
127
128         unsigned int mNumFrames;
129         unsigned int mCurrentFrameIndex;
130
131         Microsoft::WRL::ComPtr<ID3D12Device> mDirect3DDevice;
132
133         Microsoft::WRL::ComPtr<IDXGIFactory4> mDXGIFactory;
134
135         Microsoft::WRL::ComPtr<ID3D12Fence> mFence;
136         UINT64 mFenceValue;
137         std::vector<UINT64> mCurrentFrameFenceValue;
138
139         Microsoft::WRL::ComPtr<ID3D12CommandQueue> mCommandQueue;
140         std::vector<Microsoft::WRL::ComPtr<ID3D12CommandAllocator> mCommandAllocators;
141         Microsoft::WRL::ComPtr<ID3D12CommandAllocator> mDirectCommandAllocator;
142         Microsoft::WRL::ComPtr<ID3D12GraphicsCommandList> mCommandList;
143
144         UINT mRTVSize;
145         UINT mDSVSize;
146         UINT mCBVSize;
147
148         Microsoft::WRL::ComPtr<ID3D12DescriptorHeap> mRTVHeap;
149         Microsoft::WRL::ComPtr<ID3D12DescriptorHeap> mDSVHeap;
150
151         SwapChain mSwapChain;
152
153         bool mIsMSAAEnabled;
154         MultiSampling mMultiSampling;
155
156         D3D12_VIEWPORT mViewport{};
157         D3D12_RECT mScissor{};
158
159         bool mIsTextEnabled;
160         TextResources mTextResources;
161
162         //Call all of these functions to initialize Direct3D
163         void mEnableDebugLayer();
164         void mCreateDirect3DDevice();
165         void mCreateDXGIFactory();
166         void mCreateFence();
167         void mQueryDescriptorSizes();
168         void mCreateRTVHeap();
169         void mCreateDSVHeap();
170         void mCreateCommandObjects();
171     };
172 }

```

## 6.10 FADirectXException.h

```

1 #pragma once
2
3 #include <wrl.h>
4 #include <dxgidebug.h>
5 #include <comdef.h>
6 #include <string>

```

```

7 #include <sstream>
8 #include <vector>
9
10 inline std::wstring AnsiToWString(const std::string& str)
11 {
12     WCHAR buffer[1024];
13     MultiByteToWideChar(CP_ACP, 0, str.c_str(), -1, buffer, 1024);
14     return std::wstring(buffer);
15 }
16
17 class DirectXException
18 {
19 public:
20     DirectXException(HRESULT hr, const std::wstring& functionName, const std::wstring& fileName, int
        lineNumber);
21
22     std::wstring ErrorMsg() const;
23
24 private:
25     HRESULT errorCode;
26     std::wstring functionName;
27     std::wstring fileName;
28     int lineNumber;
29     Microsoft::WRL::ComPtr<IDXGIInfoQueue> mInfoQueue;
30 };
31
32 //use when calling Direct3D or DXGI function to check if the function failed or not.
33 #ifndef ThrowIfFailed
34 #define ThrowIfFailed(x)
35 {
36     HRESULT hr = (x);
37     std::wstring filename(AnsiToWString(__FILE__));
38     if (FAILED(hr)) { throw DirectXException(hr, L#x, filename, __LINE__); }
39 }
40 #endif

```

## 6.11 FAMultiSampling.h

```

1 #pragma once
2
3 #include <wrl.h>
4 #include "d3dx12.h"
5 #include "FABuffer.h"
6
7 namespace FAREnder
8 {
9     class MultiSampling
10     {
11     public:
12
13         MultiSampling() = default;
14
15         MultiSampling(const Microsoft::WRL::ComPtr<ID3D12Device>& device,
16             DXGI_FORMAT rtFormat, DXGI_FORMAT dsFormat, unsigned int sampleCount);
17
18         const Microsoft::WRL::ComPtr<ID3D12Resource>& GetRenderTargetBuffer();
19
20         DXGI_FORMAT GetRenderTargetFormat();
21
22         DXGI_FORMAT GetDepthStencilFormat();
23
24         void ResetBuffers();
25
26         void CreateRenderTargetBufferAndView(const Microsoft::WRL::ComPtr<ID3D12Device>& device,
27             const Microsoft::WRL::ComPtr<ID3D12DescriptorHeap>& rtvHeap, unsigned int
28             indexWhereToStoreView, unsigned int rtvSize,
29             unsigned int width, unsigned int height);
30
31         void CreateDepthStencilBufferAndView(const Microsoft::WRL::ComPtr<ID3D12Device>& device,
32             const Microsoft::WRL::ComPtr<ID3D12DescriptorHeap>& dsvHeap, unsigned int
33             indexWhereToStoreView, unsigned int dsvSize,
34             unsigned int width, unsigned int height);
35
36         void Transition(const Microsoft::WRL::ComPtr<ID3D12GraphicsCommandList>& commandList,
37             D3D12_RESOURCE_STATES before, D3D12_RESOURCE_STATES after);
38
39         void ClearRenderTargetBuffer(const Microsoft::WRL::ComPtr<ID3D12GraphicsCommandList>&
40             commandList,
41             const Microsoft::WRL::ComPtr<ID3D12DescriptorHeap>& rtvHeap, unsigned int indexWhereToStoreView,
42             unsigned int rtvSize,
43             const float* clearValue);
44
45         void ClearDepthStencilBuffer(const Microsoft::WRL::ComPtr<ID3D12GraphicsCommandList>&
46             commandList,

```

```

63         const Microsoft::WRL::ComPtr<ID3D12DescriptorHeap>& dsvHeap, unsigned int indexOfView,
        unsigned int dsvSize,
64         float clearValue);
65
66     private:
67         RenderTargetBuffer mMSAARenderTargetBuffer;
68         DepthStencilBuffer mMSAADepthStencilBuffer;
69         unsigned int mSampleCount{ 0 };
70
71         /*Microsoft::WRL::ComPtr<ID3D12DescriptorHeap> mMSAARTVDescriptorHeap;
72 Microsoft::WRL::ComPtr<ID3D12DescriptorHeap> mMSAADSVDescriptorHeap;
73 Microsoft::WRL::ComPtr<ID3D12Resource> mMSAARenderTargetBuffer;
74 Microsoft::WRL::ComPtr<ID3D12Resource> mMSAADepthStencilBuffer;*/
75
76     };
77 }

```

## 6.12 FARenderScene.h File Reference

File has class `RenderScene` under namespace `FARender`.

```

#include <d3dcompiler.h>
#include <unordered_map>
#include <string>
#include "FADeviceResources.h"
#include "FABuffer.h"
#include "FACamera.h"
#include "FAText.h"
#include "FAShapesUtility.h"

```

### Classes

- struct `FARender::DrawSettings`  
*Holds a array of objects that use the same PSO, root signature and primitive.*
- class `FARender::RenderScene`  
*This class is used to render a scene using Direct3D 12 API.*

### Namespaces

- namespace `FARender`  
*Has classes that are used for rendering objects and text through the Direct3D 12 API.*

#### 6.12.1 Detailed Description

File has class `RenderScene` under namespace `FARender`.

## 6.13 FARenderScene.h

[Go to the documentation of this file.](#)

```

1 #pragma once
2
3 #include <d3dcompiler.h>
4 #include <unordered_map>
5 #include <string>
6 #include "FADeviceResources.h"
7 #include "FABuffer.h"
8 #include "FACamera.h"
9 #include "FAText.h"
10 #include "FAShapesUtility.h"
11
12 namespace FARender
13 {
14     struct DrawSettings
15     {
16         Microsoft::WRL::ComPtr<ID3D12PipelineState> pipelineState;
17         Microsoft::WRL::ComPtr<ID3D12RootSignature> rootSig;
18         D3D_PRIMITIVE_TOPOLOGY prim;
19         std::vector<FAShapes::DrawArguments> drawArgs;
20     };
21
22     class RenderScene
23     {
24     public:
25         RenderScene(unsigned int width, unsigned int height, HWND windowHandle);
26         RenderScene(const RenderScene&) = delete;
27         RenderScene& operator=(const RenderScene&) = delete;
28
29         /*@brief Returns a constant reference to the device resources object.
30 */
31         const DeviceResources& GetDeviceResources() const;
32
33         /*@brief Returns a reference to the specified DrawArguments structure in the specified
34 DrawSettings structure.
35 * Throws an out_of_range exception if the index to DrawSettings structure or index to the DrawArguments
36 structure
37 * is out of bounds.
38 */
39         FAShapes::DrawArguments& GetDrawArguments(unsigned int drawSettingsIndex, unsigned int
40 drawArgsIndex);
41
42         /*@brief Returns a constant reference to the specified DrawArguments object in the specified
43 DrawSettings structure.
44 * Throws an out_of_range exception if the index to DrawSettings structure or index to the DrawArguments
45 structure
46 * is out of bounds.
47 */
48         const FAShapes::DrawArguments& GetDrawArguments(unsigned int drawSettingsIndex, unsigned int
49 drawArgsIndex) const;
50
51         /*@brief Returns a reference to the this scene's camera;
52 */
53         FACamera::Camera& GetCamera();
54
55         /*@brief Returns a constant reference to the this scene's camera;
56 */
57         const FACamera::Camera& GetCamera() const;
58
59         /*@brief Returns a reference to the specified Text object.
60 * If the index of the Text object is out of bounds an out_of_range exception is thrown.
61 */
62         FARender::Text& GetText(unsigned int textIndex);
63
64         /*@brief Returns a constant reference to the specified Text object.
65 * If the index of the Text object is out of bounds an out_of_range exception is thrown.
66 */
67         const FARender::Text& GetText(unsigned int textIndex) const;
68
69         /*@brief Loads a shader's bytecode and stores it.
70 */
71         void LoadShader(const std::wstring& filename);
72
73         /*@brief Loads a shaders file, compiles it into bytecode and stores the bytecode.
74 */
75         void LoadShaderAndCompile(const std::wstring& filename, const std::string& entryPointName, const
76 std::string& target);
77
78         /*@brief Removes the Shader structue at the specified index.
79 * If the index to the specified Shader structure is out of bounds, an out_of_range exception is thrown.
80 */
81
82
83
84
85

```

```

86 */
87     void RemoveShader(unsigned int index);
88
89     /*@brief Creates an input element description and stores it.
90 */
91     void CreateInputElementDescription(const char* semanticName, unsigned int semanticIndex,
92         DXGI_FORMAT format, unsigned int inputSlot, unsigned int byteOffset,
93         D3D12_INPUT_CLASSIFICATION inputSlotClassification =
94             D3D12_INPUT_CLASSIFICATION_PER_VERTEX_DATA,
95         unsigned int instanceStepRate = 0);
96
97     /*@brief Removes the specified input element description.
98 * If the index to the input element description is out of bounds an out_of_range exceptio is thrown.
99 */
100    void RemoveInputElementDescription(unsigned int index);
101
102    /*@brief Creates a PSO and stores it in the specified DrawSettings structure.
103 * If the indices to the specified DrawSettings structure, Vertex Shader or Pixel Shader
104 * is out of bounds an out_of_range exception is thrown.
105 */
106    void CreatePSO(unsigned int drawSettingsIndex, D3D12_FILL_MODE fillMode, BOOL enableMultisample,
107        unsigned int vsIndex, unsigned int psIndex,
108        const D3D12_PRIMITIVE_TOPOLOGY_TYPE& primitiveType, UINT sampleCount);
109
110    /*@brief Creates a root signature and stores it in the specified DrawSettings structure.
111 * If the index to the specified DrawSettings structure is out of bounds an out_of_range exception is
112 * thrown.
113 */
114    void CreateRootSignature(unsigned int drawSettingsIndex);
115
116    /*@brief Creates a vertex buffer with the specified name and stores all of the added vertices.
117 * Also creates a view to the vertex buffer.\n
118 * Execute commands and the flush command queue after calling createVertexBuffer() and
119 * createIndexBuffer().
120 */
121    void CreateVertexBuffer();
122
123    void CreateIndexBuffer();
124
125    void CreateCBVHeap(UINT numDescriptors, UINT shaderRegister);
126
127    void CreateConstantBuffer(UINT numOfBytes);
128
129    void CreateConstantBufferView(UINT index, UINT numBytes);
130
131    void SetPSO(unsigned int drawSettingsIndex, const Microsoft::WRL::ComPtr<ID3D12PipelineState>&
132        pso);
133
134    void SetRootSignature(unsigned int drawSettingsIndex,
135        const Microsoft::WRL::ComPtr<ID3D12RootSignature>& rootSignature);
136
137    void SetPrimitive(unsigned int drawSettingsIndex, const D3D_PRIMITIVE_TOPOLOGY& primitive);
138
139    void AddDrawArgument(unsigned int drawSettingsIndex, const FAShapes::DrawArguments& drawArg);
140
141    void CreateDrawArgument(unsigned int drawSettingsIndex,
142        unsigned int indexCount, unsigned int locationOfFirstIndex, int indexOfFirstVertex, int
143        indexOfConstantData);
144
145    void RemoveDrawArgument(unsigned int drawSettingsIndex, unsigned int drawArgIndex);
146
147    void CreateDrawSettings();
148
149    void RemoveDrawSettings(unsigned int drawSettingsIndex);
150
151    void CreateText(FAMath::Vector4D textLocation, const std::wstring& textString,
152        float textSize, const FIColor::Color textColor);
153
154    void RemoveText(unsigned int textIndex);
155
156    void AddVertices(const std::vector<FAShapes::Vertex>& vertices);
157
158    void AddVertices(const FAShapes::Vertex* vertices, unsigned int numVertices);
159
160    void AddIndices(const std::vector<unsigned int>& indices);
161
162    void AddIndices(const unsigned int* indices, unsigned int numIndices);
163
164    void BeforeDrawObjects();
165
166    void DrawObjects(unsigned int drawSettingsIndex);
167
168    void AfterDrawObjects();
169
170    void BeforeDrawText();
171
172    void RenderText(unsigned int textIndex);

```



```

248
252     void AfterDrawText();
253
257     void AfterDraw();
258
261     void ExecuteAndFlush();
262
265     void NextFrame();
266
269     void Resize(unsigned int width, unsigned int height, HWND windowHandle);
270
273     void CopyData(UINT index, UINT byteSize, const void* data, UINT64 numBytes);
274
277     bool IsMSAAEnabled() const;
278
281     void DisableMSAA(unsigned int width, unsigned int height, HWND windowHandle);
282
285     void EnableMSAA(unsigned int width, unsigned int height, HWND windowHandle);
286
289     bool IsTextEnabled() const;
290
293     void DisableText(unsigned int width, unsigned int height, HWND windowHandle);
294
297     void EnableText(unsigned int width, unsigned int height, HWND windowHandle);
298
299 private:
300
301     static const unsigned int NUM_OF_FRAMES{ 3 };
302
303     //The device resources object that all RenderScene objects share.
304     DeviceResources& mDeviceResources;
305
306     //Stores all of the shaders for this scene.
307     std::vector<Microsoft::WRL::ComPtr<ID3DBlob>> mShaders;
308
309     //Stores input element descriptions for the shaders.
310     std::vector<D3D12_INPUT_ELEMENT_DESC> mInputElementDescriptions;
311
312     //Stores draw settings that the scene uses.
313     std::vector<DrawSettings> mSceneObjects;
314
315     //Each scene gets a CBV heap.
316     Microsoft::WRL::ComPtr<ID3D12DescriptorHeap> mCBVHeap;
317     D3D12_DESCRIPTOR_RANGE mCBVHeapDescription{};
318     D3D12_ROOT_PARAMETER mCBVHeapRootParameter;
319
320     //Stores all of the constant buffers this scene uses. We can't update a constant buffer until
the GPU
321     //is done executing all the commands that reference it, so each frame needs its own constant
buffer.
322     ConstantBuffer mConstantBuffer[NUM_OF_FRAMES];
323
324     //The vertices and indices for the scene.
325     std::vector<FAShapes::Vertex> mVertexList;
326     std::vector<unsigned int> mIndexList;
327
328     //The vertex and index buffer for the scene.
329     VertexBuffer mVertexBuffer;
330     IndexBuffer mIndexBuffer;
331
332     //All of the text that is rendered with the scene.
333     std::vector<Text> mTexts;
334
335     //The camera for the scene.
336     FACamera::Camera mCamera;
337 }
338 }

```

## 6.14 FASwapChain.h

```

1 #pragma once
2
3 #include <wrl.h>
4 #include "d3dx12.h"
5 #include <dxgi1_4.h>
6 #include <vector>
7 #include "FABuffer.h"
8
9 namespace FASwapChain
10 {
11     class SwapChain
12     {
13     public:

```

```

17
18     SwapChain() = default;
19
20
21
22
23     SwapChain(const Microsoft::WRL::ComPtr<IDXGIFactory4>& dxgiFactory,
24               const Microsoft::WRL::ComPtr<ID3D12CommandQueue>& commandQueue, HWND windowHandle,
25               DXGI_FORMAT rtFormat = DXGI_FORMAT_R8G8B8A8_UNORM, DXGI_FORMAT dsFormat =
DXGI_FORMAT_D24_UNORM_S8_UINT,
26               unsigned int numRenderTargetBuffers = 2);
27
28
29
30     const RenderTargetBuffer* GetRenderTargetBuffers() const;
31
32
33     const Microsoft::WRL::ComPtr<ID3D12Resource>& GetCurrentBackBuffer() const;
34
35
36     unsigned int GetNumRenderTargetBuffers() const;
37
38     unsigned int GetCurrentBackBufferIndex() const;
39
40
41     DXGI_FORMAT GetBackBufferFormat() const;
42
43     DXGI_FORMAT GetDepthStencilFormat() const;
44
45
46     void ResetBuffers();
47
48     void ResizeSwapChain(unsigned width, unsigned height);
49
50     void CreateRenderTargetBuffersAndViews(const Microsoft::WRL::ComPtr<ID3D12Device>& device,
51               const Microsoft::WRL::ComPtr<ID3D12DescriptorHeap>& rtvHeap, unsigned int
indexOfWhereToStoreFirstView,
52               unsigned int rtvSize);
53
54     void CreateDepthStencilBufferAndView(const Microsoft::WRL::ComPtr<ID3D12Device>& device,
55               const Microsoft::WRL::ComPtr<ID3D12DescriptorHeap>& dsvHeap, unsigned int index, unsigned int
dsvSize,
56               unsigned int width, unsigned int height);
57
58     void ClearCurrentBackBuffer(const Microsoft::WRL::ComPtr<ID3D12GraphicsCommandList>& commandList,
59               const Microsoft::WRL::ComPtr<ID3D12DescriptorHeap>& rtvHeap, unsigned int indexOfFirstView,
60               unsigned int rtvSize,
61               const float* backBufferClearValue);
62
63     void ClearDepthStencilBuffer(const Microsoft::WRL::ComPtr<ID3D12GraphicsCommandList>&
commandList,
64               const Microsoft::WRL::ComPtr<ID3D12DescriptorHeap>& dsvHeap, unsigned int indexOfView,
65               unsigned int dsvSize,
66               float clearValue);
67
68     void Transition(const Microsoft::WRL::ComPtr<ID3D12GraphicsCommandList>& commandList,
69               D3D12_RESOURCE_STATES before, D3D12_RESOURCE_STATES after);
70
71     void Present();
72
73 private:
74     unsigned int mNumRenderTargetBuffers = 0;
75     unsigned int mCurrentBackBufferIndex = 0;
76
77     Microsoft::WRL::ComPtr<IDXGISwapChain1> mSwapChain;
78     std::vector<RenderTargetBuffer> mRenderTargetBuffers;
79
80     DepthStencilBuffer mDepthStencilBuffer;
81 };
82 }

```

## 6.15 FAText.h File Reference

File has class `Text` under namespace `FARender`.

```

#include <string>
#include "FAColor.h"

```

### Classes

- class `FARender::Text`

*This class is used to help render text. Stores the location of the text, the text string, text size and the color of the text.*

## Namespaces

- namespace [FARender](#)

*Has classes that are used for rendering objects and text through the Direct3D 12 API.*

### 6.15.1 Detailed Description

File has class Text under namespace [FARender](#).

## 6.16 FAText.h

[Go to the documentation of this file.](#)

```

1 #pragma once
2
3 #include <string>
4 #include "FAColor.h"
5
6 namespace FASystem
7 {
8     namespace FASystem
9     {
10         namespace FASystem
11         {
12             namespace FASystem
13             {
14                 namespace FASystem
15                 {
16                     class Text
17                     {
18     public:
19
20         Text() = default;
21
22         Text(const FAMath::Vector4D& textLocation, const std::wstring& textString, float textSize, const
FAColor::Color& textColor);
23
24         const FAMath::Vector4D& GetTextLocation() const;
25
26         const std::wstring& GetTextString() const;
27
28         float GetTextSize() const;
29
30         const FAColor::Color& GetTextColor() const;
31
32         void SetTextSize(float textSize);
33
34         void SetTextColor(const FAColor::Color& textColor);
35
36         void SetTextString(const std::wstring& textString);
37
38         void SetTextLocation(const FAMath::Vector4D& textLocation);
39
40     private:
41
42         FAMath::Vector4D mTextLocation;
43         std::wstring mText;
44         float mTextSize{ 0.0f };
45         FAColor::Color mTextColor;
46     };
47 }
48 }
49 }
50 }
51 }
52 }
53 }
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }

```

## 6.17 FATextResources.h

```

1 #pragma once
2
3 #include <wrl.h>
4 #include <d3d11.h>
5 #include <d3d11on12.h>
6 #include <d2d1_3.h>
7 #include <dwrite.h>
8 #include <vector>
9 #include "FABuffer.h"
10
11 namespace FASystem
12 {
13     namespace FASystem
14     {
15         namespace FASystem
16         {
17             namespace FASystem
18             {
19                 namespace FASystem
20                 {
21                     class TextResources
22                     {
23     public:
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```

19     TextResources() = default;
20
21     TextResources(const Microsoft::WRL::ComPtr<ID3D12Device>& device,
22         const Microsoft::WRL::ComPtr<ID3D12CommandQueue>& commandQueue, unsigned int
23         numSwapChainBuffers);
24
25     const Microsoft::WRL::ComPtr<ID2D1DeviceContext>& GetDirect2DDeviceContext() const;
26
27     const Microsoft::WRL::ComPtr<IDWriteFactory>& GetDirectWriteFactory() const;
28
29     void ResetBuffers();
30
31     void ResizeBuffers(const RenderTargetBuffer* renderTargetBuffers, HWND windowHandle);
32
33     void BeforeRenderText(unsigned int currentBackBuffer);
34
35     void AfterRenderText(unsigned int currentBackBuffer);
36
37 private:
38     Microsoft::WRL::ComPtr<ID3D11Device> mDevice11;
39     Microsoft::WRL::ComPtr<ID3D11DeviceContext> mDevice11Context;
40     Microsoft::WRL::ComPtr<ID3D11On12Device> mDevice11on12;
41
42     Microsoft::WRL::ComPtr<ID2D1Device2> mDirect2DDevice;
43     Microsoft::WRL::ComPtr<ID2D1Factory3> mDirect2DFactory;
44     Microsoft::WRL::ComPtr<ID2D1DeviceContext> mDirect2DDeviceContext;
45
46     Microsoft::WRL::ComPtr<IDWriteFactory> mDirectWriteFactory;
47
48     std::vector<Microsoft::WRL::ComPtr<ID3D11Resource>> mWrappedBuffers;
49     std::vector<Microsoft::WRL::ComPtr<ID2D1Bitmap1>> mDirect2DBuffers;
50     std::vector<Microsoft::WRL::ComPtr<IDXGISurface>> mSurfaces;
51 };
52 }

```

## 6.18 FATime.h File Reference

File that has namespace FATime. Withn the namespace is the class [Time](#).

```
#include <Windows.h>
```

### Classes

- class [FATime::Time](#)

#### 6.18.1 Detailed Description

File that has namespace FATime. Withn the namespace is the class [Time](#).

## 6.19 FATime.h

[Go to the documentation of this file.](#)

```

1  #pragma once
2
3  #include <Windows.h>
4
5  namespace FATime
6  {
7      class Time
8      {
9      public:
10         Time();
11
12         void Tick();

```

```

25
28     float DeltaTime() const;
29
32     void Reset();
33
36     void Stop();
37
40     void Start();
41
44     float TotalTime() const;
45
46     private:
47         __int64 mCurrTime; //holds current time stamp ti
48         __int64 mPrevTime; //holds previous time stamp ti-1
49         __int64 mStopTime; //holds the time we stopped the game/animation
50         __int64 mPausedTime; //holds how long the game/animation was paused for
51         __int64 mBaseTime; //holds the time we started / resetted
52
53         double mSecondsPerCount;
54         double mDeltaTime; //time elapsed btw frames change in t = ti - ti-1
55
56         bool mStopped; //flag to indicate if the game/animation is paused or not
57     };
58 }
59 }

```

## 6.20 FAWindow.h File Reference

File that has namespace [FAWindow](#). Withn the namespace is the class Window.

```

#include <Windows.h>
#include <string>
#include <stdexcept>

```

### Classes

- class [FAWindow::Window](#)  
*The window class is used to make a [Window](#) using Windows API.*

### Namespaces

- namespace [FAWindow](#)  
*Has [Window](#) class.*

#### 6.20.1 Detailed Description

File that has namespace [FAWindow](#). Withn the namespace is the class Window.

## 6.21 FAWindow.h

[Go to the documentation of this file.](#)

```
1 #pragma once
2
3
4
5
6
7 #include <Windows.h>
8 #include <string>
9 #include <stdexcept>
10
11
12
13
14 namespace FAWindow
15 {
16
17     class Window
18     {
19     public:
20         //Window();
21
22         Window(const HINSTANCE& hInstance, const std::wstring& windowClassName, const std::wstring&
23 windowName,
24 WNDPROC winProcFunction, unsigned int width, unsigned int height, void* additionalData =
25 nullptr);
26
27         Window(const HINSTANCE& hInstance, const WNDCLASSEX& windowClass, const std::wstring& windowName,
28 unsigned int width, unsigned int height, void* additionalData = nullptr);
29
30         HWND GetWindowHandle() const;
31
32         unsigned int GetWidth() const ;
33
34         unsigned int GetHeight() const;
35
36         void SetWidth(unsigned int width);
37
38         void SetHeight(unsigned int height);
39
40     private:
41         HWND mWindowHandle;
42
43         WNDCLASSEX mWindowClass;
44         std::wstring mWindowClassName;
45
46         unsigned int mWidth;
47         unsigned int mHeight;
48     };
49 }
```

# Index

- ~ConstantBuffer
  - FARender::ConstantBuffer, [23](#)
- ~DeviceResources
  - FARender::DeviceResources, [27](#)
- AddDrawArgument
  - FARender::RenderScene, [40](#)
- AddIndices
  - FARender::RenderScene, [40](#)
- AddVertices
  - FARender::RenderScene, [41](#)
- AfterDraw
  - FARender::RenderScene, [41](#)
- AfterDrawObjects
  - FARender::RenderScene, [41](#)
- AfterDrawText
  - FARender::RenderScene, [41](#)
- AfterRenderText
  - FARender::TextResources, [57](#)
- AfterTextDraw
  - FARender::DeviceResources, [28](#)
- Backward
  - FACamera::Camera, [11](#)
- BeforeDrawObjects
  - FARender::RenderScene, [41](#)
- BeforeDrawText
  - FARender::RenderScene, [42](#)
- BeforeRenderText
  - FARender::TextResources, [57](#)
- BeforeTextDraw
  - FARender::DeviceResources, [28](#)
- Camera
  - FACamera::Camera, [11](#)
- ClearCurrentBackBuffer
  - FARender::SwapChain, [51](#)
- ClearDepthStencilBuffer
  - FARender::DepthStencilBuffer, [25](#)
  - FARender::MultiSampling, [36](#)
  - FARender::SwapChain, [51](#)
- ClearRenderTargetBuffer
  - FARender::MultiSampling, [36](#)
  - FARender::RenderTargetBuffer, [48](#)
- Color
  - FAColor::Color, [19](#)
- CopyData
  - FARender::ConstantBuffer, [23](#)
  - FARender::RenderScene, [42](#)
- CreateCBVHeap
  - FARender::RenderScene, [42](#)
- CreateConstantBuffer
  - FARender::ConstantBuffer, [23](#)
  - FARender::RenderScene, [42](#)
- CreateConstantBufferView
  - FARender::ConstantBuffer, [24](#)
  - FARender::RenderScene, [42](#)
- CreateDepthStencilBufferAndView
  - FARender::DepthStencilBuffer, [25](#)
  - FARender::MultiSampling, [36](#)
  - FARender::SwapChain, [51](#)
- CreateDrawArgument
  - FARender::RenderScene, [43](#)
- CreateDrawSettings
  - FARender::RenderScene, [43](#)
- CreateIndexBuffer
  - FARender::IndexBuffer, [34](#)
  - FARender::RenderScene, [43](#)
- CreateIndexBufferView
  - FARender::IndexBuffer, [34](#)
- CreateRenderTargetBufferAndView
  - FARender::MultiSampling, [36](#)
  - FARender::RenderTargetBuffer, [48](#)
- CreateRenderTargetBuffersAndViews
  - FARender::SwapChain, [51](#)
- CreateText
  - FARender::RenderScene, [43](#)
- CreateVertexBuffer
  - FARender::VertexBuffer, [61](#)
- CreateVertexBufferView
  - FARender::VertexBuffer, [61](#)
- DeltaTime
  - FATime::Time, [59](#)
- DepthStencilBuffer
  - FARender::DepthStencilBuffer, [25](#)
- DirectXException, [33](#)
- DisableMSAA
  - FARender::DeviceResources, [28](#)
  - FARender::RenderScene, [43](#)
- DisableText
  - FARender::DeviceResources, [28](#)
  - FARender::RenderScene, [44](#)
- Down
  - FACamera::Camera, [12](#)
- DrawObjects
  - FARender::RenderScene, [44](#)
- EnableMSAA
  - FARender::DeviceResources, [28](#)

- FARender::RenderScene, 44
- EnableText
  - FARender::DeviceResources, 29
  - FARender::RenderScene, 44
- Execute
  - FARender::DeviceResources, 29
- ExecuteAndFlush
  - FARender::RenderScene, 45
- FABuffer.h, 65
- FACamera, 7
- FACamera.h, 67
  - vec2, 68
- FACamera::Camera, 9
  - Backward, 11
  - Camera, 11
  - Down, 12
  - Foward, 12
  - GetAngularVelocity, 12
  - GetAspectRatio, 12
  - GetCameraPosition, 12
  - GetCameraVelocity, 12
  - GetPerspectiveProjectionMatrix, 13
  - GetVerticalFov, 13
  - GetViewPerspectiveProjectionMatrix, 13
  - GetViewTransformationMatrix, 13
  - GetX, 13
  - GetY, 13
  - GetZ, 14
  - GetZFar, 14
  - GetZNear, 14
  - KeyboardInput, 14
  - Left, 14
  - LookAt, 14
  - MouseInput, 15
  - Right, 15
  - RotateCameraLeftRight, 15
  - RotateCameraUpDown, 15
  - SetAngularVelocity, 15
  - SetAspectRatio, 16
  - SetCameraPosition, 16
  - SetCameraVelocity, 16
  - SetVerticalFov, 16
  - SetX, 16
  - SetY, 16
  - SetZ, 17
  - SetZFar, 17
  - SetZNear, 17
  - Up, 17
  - UpdatePerspectiveProjectionMatrix, 17
  - UpdateViewMatrix, 17
  - UpdateViewPerspectiveProjectionMatrix, 18
- FAColor.h, 70
  - operator\*, 70, 71
  - operator+, 71
  - operator-, 71
- FAColor::Color, 18
  - Color, 19
  - GetAlpha, 20
  - GetBlue, 20
  - GetColor, 20
  - GetGreen, 20
  - GetRed, 20
  - operator\*=, 20, 21
  - operator+=, 21
  - operator-=, 21
  - SetAlpha, 21
  - SetBlue, 21
  - SetColor, 22
  - SetGreen, 22
  - SetRed, 22
- FADeviceResources.h, 72
- FARender, 7
- FARender::ConstantBuffer, 22
  - ~ConstantBuffer, 23
  - CopyData, 23
  - CreateConstantBuffer, 23
  - CreateConstantBufferView, 24
- FARender::DepthStencilBuffer, 24
  - ClearDepthStencilBuffer, 25
  - CreateDepthStencilBufferAndView, 25
  - DepthStencilBuffer, 25
  - GetDepthStencilFormat, 25
  - ResetBuffer, 26
- FARender::DeviceResources, 26
  - ~DeviceResources, 27
  - AfterTextDraw, 28
  - BeforeTextDraw, 28
  - DisableMSAA, 28
  - DisableText, 28
  - EnableMSAA, 28
  - EnableText, 29
  - Execute, 29
  - FlushCommandQueue, 29
  - GetBackBufferFormat, 29
  - GetCBVSize, 29
  - GetCommandList, 30
  - GetCurrentFrame, 30
  - GetDepthStencilFormat, 30
  - GetDevice, 30
  - GetInstance, 30
  - GetNumFrames, 30
  - GetTextResources, 31
  - IsMSAAEnabled, 31
  - IsTextEnabled, 31
  - NextFrame, 31
  - Present, 31
  - Resize, 31
  - RTBufferTransition, 32
  - Signal, 32
  - UpdateCurrentFrameFenceValue, 32
  - WaitForGPU, 32
- FARender::DrawSettings, 33
- FARender::IndexBuffer, 33
  - CreateIndexBuffer, 34
  - CreateIndexBufferView, 34
  - GetIndexBufferView, 34



- FARender::MultiSampling, 35
  - ClearDepthStencilBuffer, 36
  - ClearRenderTargetBuffer, 36
  - CreateDepthStencilBufferAndView, 36
  - CreateRenderTargetBufferAndView, 36
  - GetRenderTargetBuffer, 37
  - MultiSampling, 35
  - ResetBuffers, 37
  - Transition, 37
- FARender::RenderScene, 37
  - AddDrawArgument, 40
  - AddIndices, 40
  - AddVertices, 41
  - AfterDraw, 41
  - AfterDrawObjects, 41
  - AfterDrawText, 41
  - BeforeDrawObjects, 41
  - BeforeDrawText, 42
  - CopyData, 42
  - CreateCBVHeap, 42
  - CreateConstantBuffer, 42
  - CreateConstantBufferView, 42
  - CreateDrawArgument, 43
  - CreateDrawSettings, 43
  - CreateIndexBuffer, 43
  - CreateText, 43
  - DisableMSAA, 43
  - DisableText, 44
  - DrawObjects, 44
  - EnableMSAA, 44
  - EnableText, 44
  - ExecuteAndFlush, 45
  - IsMSAAEnabled, 45
  - IsTextEnabled, 45
  - NextFrame, 45
  - RemoveDrawArgument, 45
  - RemoveDrawSettings, 45
  - RemoveText, 46
  - RenderText, 46
  - Resize, 46
  - SetPrimitive, 46
  - SetPSO, 46
  - SetRootSignature, 47
- FARender::RenderTargetBuffer, 47
  - ClearRenderTargetBuffer, 48
  - CreateRenderTargetBufferAndView, 48
  - GetRenderTargetBuffer, 48, 49
  - GetRenderTargetFormat, 49
  - RenderTargetBuffer, 48
  - ResetBuffer, 49
- FARender::SwapChain, 49
  - ClearCurrentBackBuffer, 51
  - ClearDepthStencilBuffer, 51
  - CreateDepthStencilBufferAndView, 51
  - CreateRenderTargetBuffersAndViews, 51
  - GetBackBufferFormat, 52
  - GetCurrentBackBuffer, 52
  - GetCurrentBackBufferIndex, 52
  - GetDepthStencilFormat, 52
  - GetNumRenderTargetBuffers, 52
  - GetRenderTargetBuffers, 52
  - Present, 53
  - ResetBuffers, 53
  - ResizeSwapChain, 53
  - SwapChain, 50
  - Transition, 53
- FARender::Text, 54
  - GetTextColor, 55
  - GetTextLocation, 55
  - GetTextSize, 55
  - GetTextString, 55
  - SetTextColor, 55
  - SetTextLocation, 55
  - SetTextSize, 56
  - SetTextString, 56
  - Text, 54
- FARender::TextResources, 56
  - AfterRenderText, 57
  - BeforeRenderText, 57
  - GetDirect2DDeviceContext, 57
  - GetDirectWriteFactory, 57
  - ResetBuffers, 58
  - ResizeBuffers, 58
  - TextResources, 57
- FARender::VertexBuffer, 60
  - CreateVertexBuffer, 61
  - CreateVertexBufferView, 61
  - GetVertexBufferView, 61
- FARenderScene.h, 76
- FAText.h, 80
- FATime.h, 82
- FATime::Time, 58
  - DeltaTime, 59
  - Reset, 59
  - Start, 59
  - Stop, 59
  - Tick, 59
  - Time, 59
  - TotalTime, 60
- FAWindow, 8
- FAWindow.h, 83
- FAWindow::Window, 61
  - GetHeight, 63
  - GetWidth, 63
  - GetWindowHandle, 63
  - SetHeight, 63
  - SetWidth, 63
  - Window, 62
- FlushCommandQueue
  - FARender::DeviceResources, 29
- Foward
  - FACamera::Camera, 12
- GetAlpha
  - FAColor::Color, 20
- GetAngularVelocity
  - FACamera::Camera, 12

- GetAspectRatio
  - FACamera::Camera, [12](#)
- GetBackBufferFormat
  - FARender::DeviceResources, [29](#)
  - FARender::SwapChain, [52](#)
- GetBlue
  - FAColor::Color, [20](#)
- GetCameraPosition
  - FACamera::Camera, [12](#)
- GetCameraVelocity
  - FACamera::Camera, [12](#)
- GetCBVSize
  - FARender::DeviceResources, [29](#)
- GetColor
  - FAColor::Color, [20](#)
- GetCommandList
  - FARender::DeviceResources, [30](#)
- GetCurrentBackBuffer
  - FARender::SwapChain, [52](#)
- GetCurrentBackBufferIndex
  - FARender::SwapChain, [52](#)
- GetCurrentFrame
  - FARender::DeviceResources, [30](#)
- GetDepthStencilFormat
  - FARender::DepthStencilBuffer, [25](#)
  - FARender::DeviceResources, [30](#)
  - FARender::SwapChain, [52](#)
- GetDevice
  - FARender::DeviceResources, [30](#)
- GetDirect2DDeviceContext
  - FARender::TextResources, [57](#)
- GetDirectWriteFactory
  - FARender::TextResources, [57](#)
- GetGreen
  - FAColor::Color, [20](#)
- GetHeight
  - FAWindow::Window, [63](#)
- GetIndexBufferView
  - FARender::IndexBuffer, [34](#)
- GetInstance
  - FARender::DeviceResources, [30](#)
- GetNumFrames
  - FARender::DeviceResources, [30](#)
- GetNumRenderTargetBuffers
  - FARender::SwapChain, [52](#)
- GetPerspectiveProjectionMatrix
  - FACamera::Camera, [13](#)
- GetRed
  - FAColor::Color, [20](#)
- GetRenderTargetBuffer
  - FARender::MultiSampling, [37](#)
  - FARender::RenderTargetBuffer, [48](#), [49](#)
- GetRenderTargetBuffers
  - FARender::SwapChain, [52](#)
- GetRenderTargetFormat
  - FARender::RenderTargetBuffer, [49](#)
- GetTextColor
  - FARender::Text, [55](#)
- GetTextLocation
  - FARender::Text, [55](#)
- GetTextResources
  - FARender::DeviceResources, [31](#)
- GetTextSize
  - FARender::Text, [55](#)
- GetTextString
  - FARender::Text, [55](#)
- GetVertexBufferView
  - FARender::VertexBuffer, [61](#)
- GetVerticalFov
  - FACamera::Camera, [13](#)
- GetViewPerspectiveProjectionMatrix
  - FACamera::Camera, [13](#)
- GetViewTransformationMatrix
  - FACamera::Camera, [13](#)
- GetWidth
  - FAWindow::Window, [63](#)
- GetWindowHandle
  - FAWindow::Window, [63](#)
- GetX
  - FACamera::Camera, [13](#)
- GetY
  - FACamera::Camera, [13](#)
- GetZ
  - FACamera::Camera, [14](#)
- GetZFar
  - FACamera::Camera, [14](#)
- GetZNear
  - FACamera::Camera, [14](#)
- IsMSAAEnabled
  - FARender::DeviceResources, [31](#)
  - FARender::RenderScene, [45](#)
- IsTextEnabled
  - FARender::DeviceResources, [31](#)
  - FARender::RenderScene, [45](#)
- KeyboardInput
  - FACamera::Camera, [14](#)
- Left
  - FACamera::Camera, [14](#)
- LookAt
  - FACamera::Camera, [14](#)
- MouseInput
  - FACamera::Camera, [15](#)
- MultiSampling
  - FARender::MultiSampling, [35](#)
- NextFrame
  - FARender::DeviceResources, [31](#)
  - FARender::RenderScene, [45](#)
- operator\*
  - FAColor.h, [70](#), [71](#)
- operator\*=
  - FAColor::Color, [20](#), [21](#)
- operator+

- FAColor.h, [71](#)
- operator+=
  - FAColor::Color, [21](#)
- operator-
  - FAColor.h, [71](#)
- operator-=
  - FAColor::Color, [21](#)
- Present
  - FARender::DeviceResources, [31](#)
  - FARender::SwapChain, [53](#)
- RemoveDrawArgument
  - FARender::RenderScene, [45](#)
- RemoveDrawSettings
  - FARender::RenderScene, [45](#)
- RemoveText
  - FARender::RenderScene, [46](#)
- RenderTargetBuffer
  - FARender::RenderTargetBuffer, [48](#)
- RenderText
  - FARender::RenderScene, [46](#)
- Reset
  - FATime::Time, [59](#)
- ResetBuffer
  - FARender::DepthStencilBuffer, [26](#)
  - FARender::RenderTargetBuffer, [49](#)
- ResetBuffers
  - FARender::MultiSampling, [37](#)
  - FARender::SwapChain, [53](#)
  - FARender::TextResources, [58](#)
- Resize
  - FARender::DeviceResources, [31](#)
  - FARender::RenderScene, [46](#)
- ResizeBuffers
  - FARender::TextResources, [58](#)
- ResizeSwapChain
  - FARender::SwapChain, [53](#)
- Right
  - FACamera::Camera, [15](#)
- RotateCameraLeftRight
  - FACamera::Camera, [15](#)
- RotateCameraUpDown
  - FACamera::Camera, [15](#)
- RTBufferTransition
  - FARender::DeviceResources, [32](#)
- SetAlpha
  - FAColor::Color, [21](#)
- SetAngularVelocity
  - FACamera::Camera, [15](#)
- SetAspectRatio
  - FACamera::Camera, [16](#)
- SetBlue
  - FAColor::Color, [21](#)
- SetCameraPosition
  - FACamera::Camera, [16](#)
- SetCameraVelocity
  - FACamera::Camera, [16](#)
- SetColor
  - FAColor::Color, [22](#)
- SetGreen
  - FAColor::Color, [22](#)
- SetHeight
  - FAWindow::Window, [63](#)
- SetPrimitive
  - FARender::RenderScene, [46](#)
- SetPSO
  - FARender::RenderScene, [46](#)
- SetRed
  - FAColor::Color, [22](#)
- SetRootSignature
  - FARender::RenderScene, [47](#)
- SetTextColor
  - FARender::Text, [55](#)
- SetTextLocation
  - FARender::Text, [55](#)
- SetTextSize
  - FARender::Text, [56](#)
- SetTextString
  - FARender::Text, [56](#)
- SetVerticalFov
  - FACamera::Camera, [16](#)
- SetWidth
  - FAWindow::Window, [63](#)
- SetX
  - FACamera::Camera, [16](#)
- SetY
  - FACamera::Camera, [16](#)
- SetZ
  - FACamera::Camera, [17](#)
- SetZFar
  - FACamera::Camera, [17](#)
- SetZNear
  - FACamera::Camera, [17](#)
- Signal
  - FARender::DeviceResources, [32](#)
- Start
  - FATime::Time, [59](#)
- Stop
  - FATime::Time, [59](#)
- SwapChain
  - FARender::SwapChain, [50](#)
- Text
  - FARender::Text, [54](#)
- TextResources
  - FARender::TextResources, [57](#)
- Tick
  - FATime::Time, [59](#)
- Time, [60](#)
  - FATime::Time, [59](#)
- TotalTime
  - FATime::Time, [60](#)
- Transition
  - FARender::MultiSampling, [37](#)
  - FARender::SwapChain, [53](#)

## Up

- FACamera::Camera, [17](#)

## UpdateCurrentFrameFenceValue

- FARender::DeviceResources, [32](#)

## UpdatePerspectiveProjectionMatrix

- FACamera::Camera, [17](#)

## UpdateViewMatrix

- FACamera::Camera, [17](#)

## UpdateViewPerspectiveProjectionMatrix

- FACamera::Camera, [18](#)

## vec2

- FACamera.h, [68](#)

## WaitForGPU

- FARender::DeviceResources, [32](#)

## Window

- FAWindow::Window, [62](#)