Farouq Adepetu's Rendering Engine

Generated by Doxygen 1.9.4

1 Namespace Index 1

1.1 Namespace List . . . 1

2 Class Index 3

2.1 Class List . . . 3

3 File Index 5

3.1 File List . . . 5

4 Namespace Documentation 7

4.1 FACamera Namespace Reference . . . 7

4.1.1 Detailed Description . . . 7

4.2 FARender Namespace Reference . . . 7

4.2.1 Detailed Description . . . 8

4.2.2 Function Documentation . . . 8

4.2.2.1 nextFrame() . . . 8

4.3 FAWindow Namespace Reference . . . 8

4.3.1 Detailed Description . . . 8

5 Class Documentation 9

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 FAColor::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 FARender::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
5.3.3.4 GetConstantBuffer() [1/2] . . . 24
5.3.3.5 GetConstantBuffer() [2/2] . . . 24
5.4 FARender::DeviceResources Class Reference . . . 24
5.4.1 Detailed Description . . . 26
5.4.2 Constructor & Destructor Documentation . . . 27
5.4.2.1 ~DeviceResources() . . . 27
5.4.3 Member Function Documentation . . . 27
5.4.3.1 AfterTextDraw() . . . 27
5.4.3.2 BeforeTextDraw() . . . 27
5.4.3.3 DisableMSAA() . . . 27
5.4.3.4 EnableMSAA() . . . 27
5.4.3.5 Execute() . . . 28
5.4.3.6 FlushCommandQueue() . . . 28
5.4.3.7 GetBackBufferFormat() . . . 28
5.4.3.8 GetCommandAllocator() . . . 28
5.4.3.9 GetCommandList() . . . 28
5.4.3.10 GetCommandQueue() . . . 28
5.4.3.11 GetCurrentBackBuffer() . . . 29
5.4.3.12 GetCurrentFenceValue() . . . 29
5.4.3.13 GetDepthStencilBuffer() . . . 29
5.4.3.14 GetDepthStencilFormat() . . . 29
5.4.3.15 GetDevice() . . . 29
5.4.3.16 GetDevice2DContext() . . . 29
5.4.3.17 GetDirectWriteFactory() . . . 30
5.4.3.18 GetDSVDescriptorHeap() . . . 30
5.4.3.19 GetDSVDescriptorSize() . . . 30
5.4.3.20 GetNumOfSwapChainBuffers() . . . 30
5.4.3.21 GetRTVDescriptorHeap() . . . 30
5.4.3.22 GetRTVDescriptorSize() . . . 30
5.4.3.23 GetSampleCount() [1/2] . . . 31
5.4.3.24 GetSampleCount() [2/2] . . . 31
5.4.3.25 GetScissor() . . . 31
5.4.3.26 GetSwapChain() . . . 31
5.4.3.27 GetSwapChainBuffers() . . . 31
5.4.3.28 GetViewport() . . . 31
5.4.3.29 InitializeDirect3D() . . . 32
5.4.3.30 IsMSAAEnabled() . . . 32
5.4.3.31 Present() . . . 32

5.4.3.32 ResetCommandAllocator() . . . 32
5.4.3.33 ResetCommandList() . . . 32
5.4.3.34 Resize() . . . 33
5.4.3.35 RTBufferTransition() . . . 33
5.4.3.36 Signal() . . . 33
5.4.3.37 UpdateCurrentFrameFenceValue() . . . 33
5.4.3.38 WaitForGPU() . . . 33
5.5 DirectXException Class Reference . . . 34
5.6 FARender::DrawSettings Struct Reference . . . 34
5.6.1 Detailed Description . . . 34
5.7 FARender::IndexBuffer Class Reference . . . 34
5.7.1 Detailed Description . . . 35
5.7.2 Member Function Documentation . . . 35
5.7.2.1 CreateIndexBuffer() . . . 35
5.7.2.2 CreateIndexBufferView() . . . 35
5.7.2.3 GetIndexBufferView() . . . 35
5.8 FARender::RenderScene Class Reference . . . 36
5.8.1 Detailed Description . . . 38
5.8.2 Member Function Documentation . . . 38
5.8.2.1 AddDrawArgument() [1/2] . . . 38
5.8.2.2 AddDrawArgument() [2/2] . . . 38
5.8.2.3 AfterDraw() . . . 39
5.8.2.4 AfterDrawObjects() . . . 39
5.8.2.5 AfterDrawText() . . . 39
5.8.2.6 BeforeDrawObjects() . . . 39
5.8.2.7 BeforeDrawText() . . . 39
5.8.2.8 ChangeTextColor() . . . 40
5.8.2.9 ChangeTextLocation() . . . 40
5.8.2.10 ChangeTextSize() . . . 40
5.8.2.11 ChangeTextString() . . . 40
5.8.2.12 CreateCBVHeap() . . . 40
5.8.2.13 CreateConstantBuffer() . . . 41
5.8.2.14 CreateConstantBufferView() . . . 41
5.8.2.15 CreateDrawSettings() . . . 41
5.8.2.16 CreateIndexBuffer() . . . 41
5.8.2.17 CreateText() . . . 41
5.8.2.18 DrawObjects() . . . 42
5.8.2.19 ExecuteAndFlush() . . . 42
5.8.2.20 GetCBVHeap() . . . 42
5.8.2.21 GetCBVHeapRootParameter() . . . 42
5.8.2.22 GetCBVSize() . . . 42
5.8.2.23 RemoveDrawArgument() . . . 43

5.8.2.24 RemoveDrawSettings() . . . 43
5.8.2.25 RemoveText() . . . 43
5.8.2.26 RenderText() . . . 43
5.8.2.27 SetPrimitive() . . . 43
5.8.2.28 SetPSO() . . . 44
5.8.2.29 SetRootSignature() . . . 44
5.9 FARender::Text Class Reference . . . 44
5.9.1 Detailed Description . . . 45
5.9.2 Constructor & Destructor Documentation . . . 45
5.9.2.1 Text() [1/2] . . . 45
5.9.2.2 Text() [2/2] . . . 45
5.9.3 Member Function Documentation . . . 45
5.9.3.1 GetBrush() . . . 46
5.9.3.2 GetFormat() . . . 46
5.9.3.3 GetTextColor() . . . 46
5.9.3.4 GetTextLocation() . . . 46
5.9.3.5 GetTextSize() . . . 46
5.9.3.6 GetTextString() . . . 46
5.9.3.7 Initialize() . . . 47
5.9.3.8 SetTextColor() . . . 47
5.9.3.9 SetTextLocation() . . . 47
5.9.3.10 SetTextSize() . . . 47
5.9.3.11 SetTextString() . . . 47
5.10 FATime::Time Class Reference . . . 48
5.10.1 Constructor & Destructor Documentation . . . 48
5.10.1.1 Time() . . . 48
5.10.2 Member Function Documentation . . . 48
5.10.2.1 DeltaTime() . . . 48
5.10.2.2 Reset() . . . 48
5.10.2.3 Start() . . . 49
5.10.2.4 Stop() . . . 49
5.10.2.5 Tick() . . . 49
5.10.2.6 TotalTime() . . . 49
5.11 Time Class Reference . . . 49
5.11.1 Detailed Description . . . 49
5.12 FARender::VertexBuffer Class Reference . . . 50
5.12.1 Detailed Description . . . 50
5.12.2 Member Function Documentation . . . 50
5.12.2.1 CreateVertexBuffer() . . . 50
5.12.2.2 CreateVertexBufferView() . . . 50
5.12.2.3 GetVertexBufferView() . . . 51
5.13 FAWindow::Window Class Reference . . . 51

5.13.1 Detailed Description . . . 51
5.13.2 Constructor & Destructor Documentation . . . 51
5.13.2.1 Window() [1/2] . . . 52
5.13.2.2 Window() [2/2] . . . 52
5.13.3 Member Function Documentation . . . 52
5.13.3.1 GetHeight() . . . 52
5.13.3.2 GetWidth() . . . 52
5.13.3.3 GetWindowHandle() . . . 53
5.13.3.4 SetHeight() . . . 53
5.13.3.5 SetWidth() . . . 53

6 File Documentation **55**

6.1 Direct3DLink.h . . . 55
6.2 FABuffer.h File Reference . . . 55
6.2.1 Detailed Description . . . 56
6.3 FABuffer.h . . . 56
6.4 FACamera.h File Reference . . . 57
6.4.1 Detailed Description . . . 57
6.4.2 Typedef Documentation . . . 57
6.4.2.1 vec2 . . . 57
6.5 FACamera.h . . . 58
6.6 FAColor.h File Reference . . . 59
6.6.1 Detailed Description . . . 60
6.6.2 Function Documentation . . . 60
6.6.2.1 operator∗() [1/3] . . . 60
6.6.2.2 operator∗() [2/3] . . . 60
6.6.2.3 operator∗() [3/3] . . . 60
6.6.2.4 operator+() . . . 61
6.6.2.5 operator-() . . . 61
6.7 FAColor.h . . . 61
6.8 FADeviceResources.h File Reference . . . 62
6.8.1 Detailed Description . . . 62
6.9 FADeviceResources.h . . . 62
6.10 FADirectXException.h . . . 65
6.11 FARenderingUtility.h File Reference . . . 65
6.11.1 Detailed Description . . . 66
6.12 FARenderingUtility.h . . . 66
6.13 FARenderScene.h File Reference . . . 66
6.13.1 Detailed Description . . . 67
6.14 FARenderScene.h . . . 67
6.15 FAText.h File Reference . . . 70
6.15.1 Detailed Description . . . 70

6.16 FAText.h 70
6.17 FATime.h File Reference 71
6.17.1 Detailed Description 71
6.18 FATime.h 71
6.19 FAWindow.h File Reference 72
6.19.1 Detailed Description 72
6.20 FAWindow.h 73

Index **75**

Chapter 1

Namespace Index

1.1 Namespace List

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

FACamera
: Has Camera class 7

FARender
: The namespace has utility functions and structs, VertexBuffer, IndexBuffer, ConstantBuffer, DeviceResources, RenderScene and Text classes 7

FAWindow
: Has Window class 8

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 buffers . . . 22

FARender::DeviceResources
: A wrapper for a Direct3D 12 device, swapchain, depth buffer, MSAA buffers and command objects . . . 24

DirectXException . . . 34

FARender::DrawSettings
: Holds a array of objects that use the same PSO, root signature and primitive . . . 34

FARender::IndexBuffer
: This class stores indices in a Direct3D 12 default buffer . . . 34

FARender::RenderScene
: This class is used to render a scene using Direct3D 12 API . . . 36

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 . . . 44

FATime::Time . . . 48

Time
: This class is used to get the time between each frame. You can stop start, reset and get the total time . . . 49

FARender::VertexBuffer
: This class stores vertices in a Direct3D 12 default buffer . . . 50

FAWindow::Window
: The window class is used to make a Window using Windows API . . . 51

Chapter 3

File Index

3.1 File List

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

Direct3DLink.h . . . . . ??
FABuffer.h
File has classes VertexBuffer, IndexBuffer and ConstantBuffer under namespace FARender . . 55
FACamera.h
File that has namespace FACamera. Withn the namespace is the class Camera 57
FAColor.h
File has class Color under namespace FAColor 59
FADeviceResources.h
File has class DeviceResources under namespace FARender 62
FADirectXException.h . . . . . ??
FARenderingUtility.h
File has static variables numFrames and current frame, function nextFrame() and struct Draw↩
Arguments under the namespace FARender 65
FARenderScene.h
File has class RenderScene under namespace FARender 66
FAText.h
File has class Text under namespace FARender 70
FATime.h
File that has namespace FATime. Withn the namespace is the class Time 71
FAWindow.h
File that has namespace FAWindow. Withn the namespace is the class Window 72

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

The namespace has utility functions and structs, VertexBuffer, IndexBuffer, ConstantBuffer, DeviceResources, RenderScene and Text classes.

Classes

- class ConstantBuffer

  This class stores constant data in a Direct3D 12 upload buffers.
- class DeviceResources

  A wrapper for a Direct3D 12 device, swapchain, depth buffer, MSAA buffers and command objects.
- 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 RenderScene

  This class is used to render a scene using Direct3D 12 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 VertexBuffer

  This class stores vertices in a Direct3D 12 default buffer.

Functions

- void nextFrame ()

  Update our current frame value to go to the next frame.

4.2.1 Detailed Description

The namespace has utility functions and structs, VertexBuffer, IndexBuffer, ConstantBuffer, DeviceResources, RenderScene and Text classes.

4.2.2 Function Documentation

4.2.2.1 nextFrame()

```
void FARender::nextFrame ( )
```

Update our current frame value to go to the next frame.

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

  Returnsa 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 frustrum.
- 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 velcoity)

  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 Foward (float dt)

  Moves the camera foward 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 foward/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 frustrum.

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 frustrum 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
```

Returnsa 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 frustrum.

5.1.3.16 GetZNear()

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

Returns the near value of the frustrum.

5.1.3.17 KeyboardInput()

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

Polls keyboard input and moves the camera. Moves the camera foward/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 addtion. 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]

```
FAColor::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]

```
FAColor::Color::Color (
            const FAMath::Vector4D & color )
```

Overloaded Constructor. Initializes the color to the specified color.

5.2.3 Member Function Documentation

5.2.3.1 GetAlpha()

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

Returns the value of the alpha component.

5.2.3.2 GetBlue()

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

Returns the value of the green component.

5.2.3.3 GetColor()

```
const FAMath::Vector4D & FAColor::Color::GetColor ( ) const
```

Returns the color.

5.2.3.4 GetGreen()

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

Returns the value of the blue component.

5.2.3.5 GetRed()

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

Returns the value of the red component.

5.2.3.6 operator*=() [1/2]

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

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

5.2.3.9 operator-=()

```
Color & FAColor::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 FAColor::Color::SetAlpha (
            float a )
```

Sets the alpha component to the specified float value.

5.2.3.11 SetBlue()

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

Sets the blue component to the specified float value.

5.2.3.12 SetColor()

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

Sets the color to the specified color.

5.2.3.13 SetGreen()

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

Sets the green component to the specified float value.

5.2.3.14 SetRed()

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

Sets the red component to the specified float value.

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

- FAColor.h

5.3 FARender::ConstantBuffer Class Reference

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

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

Public Member Functions

- **ConstantBuffer** (const ConstantBuffer &)=delete
- ConstantBuffer & **operator=** (const ConstantBuffer &)=delete
- ~ConstantBuffer ()

  Unmaps the pointer to the constant buffer.
- Microsoft::WRL::ComPtr< ID3D12Resource > & GetConstantBuffer ()

  Returns a reference to the constant buffer resource.
- const Microsoft::WRL::ComPtr< ID3D12Resource > & GetConstantBuffer () const

  Returns a constant reference to the constant buffer resource.
- 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, const UINT64 &numOfBytes)

  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 buffers.

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,
            const UINT64 & numOfBytes )
```

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.

5.3.3.4 GetConstantBuffer() [1/2]

```
Microsoft::WRL::ComPtr< ID3D12Resource > & FARender::ConstantBuffer::GetConstantBuffer ( )
```

Returns a reference to the constant buffer resource.

5.3.3.5 GetConstantBuffer() [2/2]

```
const Microsoft::WRL::ComPtr< ID3D12Resource > & FARender::ConstantBuffer::GetConstantBuffer (
) const
```

Returns a constant reference to the constant buffer resource.

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

- FABuffer.h

5.4 FARender::DeviceResources Class Reference

A wrapper for a Direct3D 12 device, swapchain, depth buffer, MSAA buffers and command objects.

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

Public Member Functions

- **DeviceResources** (unsigned int width, unsigned int height, HWND windowHandle)
- **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 objcet.
- const Microsoft::WRL::ComPtr< ID3D12CommandQueue > & GetCommandQueue () const

  Returns a constant reference to the ID3D12CommandQueue objcet.
- const Microsoft::WRL::ComPtr< ID3D12CommandAllocator > & GetCommandAllocator () const

  Returns a constant reference to the current ID3D12CommandAllocator objcet.
- const Microsoft::WRL::ComPtr< ID3D12GraphicsCommandList > & GetCommandList () const

  Returns a constant reference to the ID3D12GraphicsCommandList objcet.
- const DXGI_FORMAT & GetBackBufferFormat () const

  Returns a constant reference to the back buffer format.
- const UINT GetNumOfSwapChainBuffers () const

  Returns a constant reference to the number of swap chains.
- const Microsoft::WRL::ComPtr< IDXGISwapChain1 > & GetSwapChain () const

  Returns a constant reference to the IDXGISwapChain1 object.
- const UINT & GetRTVDescriptorSize () const

  Returns a constant reference to the render target view descriptor size.
- const UINT & GetDSVDescriptorSize () const

  Returns a constant reference to the depth/stencil view descriptor size.
- const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > & GetRTVDescriptorHeap () const

  Returns a constant reference to the render target descriptor heap.
- const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > & GetDSVDescriptorHeap () const

  Returns a constant reference to the depth/stencil descriptor heap.
- const UINT & GetCurrentBackBuffer () const

  Returns a constant reference to the current back buffer.
- const Microsoft::WRL::ComPtr< ID3D12Resource > * GetSwapChainBuffers () const

  Returns a pointer to the swap chain buffers.
  There are two swap chain buffers.
  To access each buffer do swapChainBuffers()[i], where i is the index of the buffer you want to access.
- const Microsoft::WRL::ComPtr< ID3D12Resource > & GetDepthStencilBuffer () const

  Returns a constant reference to the depth stencil buffer.
- const DXGI_FORMAT & GetDepthStencilFormat () const

  Returns a constant reference to the depth stencil format.
- const D3D12_VIEWPORT & GetViewport () const

  Returns a constant reference to the D3D12_VIEWPORT objcet.
- const D3D12_RECT & GetScissor () const

  Returns a constant reference to the D3D12_RECT scissor objcet.
- bool IsMSAAEnabled ()

  Returns true if MSAA is enabled, false otherwise.
- void DisableMSAA ()

  Disables MSAA.
- void EnableMSAA ()

  Enables MSAA.
- UINT & GetSampleCount ()

  Returns a reference to the sample count.
- const UINT & GetSampleCount () const

Returns a constant reference to the sample count.

- const UINT64 & GetCurrentFenceValue () const

  Returns a constant reference to the current fence value.

- const Microsoft::WRL::ComPtr< ID2D1DeviceContext > & GetDevice2DContext () 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 UpdateCurrentFrameFenceValue ()

  Updates the current frames fence value.

- void InitializeDirect3D (unsigned int width, unsigned int height, HWND handle)

  Initializes Direct3D. Enables the debug layer if in debug mode.
  Creates a Direct3D 12 device.
  Creates a DXGI factory object.
  Creates a fence.
  Queries descriptor sizes.
  Creates command objects.
  Creates a swap chain.
  Creates a render target view and a depth/stencil view heap. Creates the initial render target buffers, depth stencil buffer, MSAA buffers and text buffers.

- 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 ResetCommandList ()

  Resets the command list to open it with a current frame command allocator.

- void **ResetDirectCommandList** ()
- void ResetCommandAllocator ()

  Resets command allocator to allow reuse of the memory.

- void RTBufferTransition (bool renderText)

  Transistions 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** ()

5.4.1 Detailed Description

A wrapper for a Direct3D 12 device, swapchain, depth buffer, MSAA buffers and command objects.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 ~DeviceResources()

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

Flushes the command queue.

5.4.3 Member Function Documentation

5.4.3.1 AfterTextDraw()

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

Executes the text commands.

5.4.3.2 BeforeTextDraw()

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

Prepares to render text.

5.4.3.3 DisableMSAA()

```
void FARender::DeviceResources::DisableMSAA ( )
```

Disables MSAA.

5.4.3.4 EnableMSAA()

```
void FARender::DeviceResources::EnableMSAA ( )
```

Enables MSAA.

5.4.3.5 Execute()

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

Executes the command list.

5.4.3.6 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.

5.4.3.7 GetBackBufferFormat()

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

Returns a constant reference to the back buffer format.

5.4.3.8 GetCommandAllocator()

```
const Microsoft::WRL::ComPtr< ID3D12CommandAllocator > & FARender::DeviceResources::Get↩
CommandAllocator ( ) const
```

Returns a constant reference to the current ID3D12CommandAllocator objcet.

5.4.3.9 GetCommandList()

```
const Microsoft::WRL::ComPtr< ID3D12GraphicsCommandList > & FARender::DeviceResources::Get↩
CommandList ( ) const
```

Returns a constant reference to the ID3D12GraphicsCommandList objcet.

5.4.3.10 GetCommandQueue()

```
const Microsoft::WRL::ComPtr< ID3D12CommandQueue > & FARender::DeviceResources::GetCommand↩
Queue ( ) const
```

Returns a constant reference to the ID3D12CommandQueue objcet.

5.4.3.11 GetCurrentBackBuffer()

```
const UINT & FARender::DeviceResources::GetCurrentBackBuffer ( ) const
```

Returns a constant reference to the current back buffer.

5.4.3.12 GetCurrentFenceValue()

```
const UINT64 & FARender::DeviceResources::GetCurrentFenceValue ( ) const
```

Returns a constant reference to the current fence value.

5.4.3.13 GetDepthStencilBuffer()

```
const Microsoft::WRL::ComPtr< ID3D12Resource > & FARender::DeviceResources::GetDepthStencil↩
Buffer ( ) const
```

Returns a constant reference to the depth stencil buffer.

5.4.3.14 GetDepthStencilFormat()

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

Returns a constant reference to the depth stencil format.

5.4.3.15 GetDevice()

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

Returns a constant reference to the ID3D12Device objcet.

5.4.3.16 GetDevice2DContext()

```
const Microsoft::WRL::ComPtr< ID2D1DeviceContext > & FARender::DeviceResources::GetDevice2↩
DContext ( ) const
```

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

5.4.3.17 GetDirectWriteFactory()

```
const Microsoft::WRL::ComPtr< IDWriteFactory > & FARender::DeviceResources::GetDirectWrite↩
Factory ( ) const
```

Returns a constant reference to the direct direct write factory.

5.4.3.18 GetDSVDescriptorHeap()

```
const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > & FARender::DeviceResources::GetDSVDescriptor↩
Heap ( ) const
```

Returns a constant reference to the depth/stencil descriptor heap.

5.4.3.19 GetDSVDescriptorSize()

```
const UINT & FARender::DeviceResources::GetDSVDescriptorSize ( ) const
```

Returns a constant reference to the depth/stencil view descriptor size.

5.4.3.20 GetNumOfSwapChainBuffers()

```
const UINT FARender::DeviceResources::GetNumOfSwapChainBuffers ( ) const
```

Returns a constant reference to the number of swap chains.

5.4.3.21 GetRTVDescriptorHeap()

```
const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > & FARender::DeviceResources::GetRTVDescriptor↩
Heap ( ) const
```

Returns a constant reference to the render target descriptor heap.

5.4.3.22 GetRTVDescriptorSize()

```
const UINT & FARender::DeviceResources::GetRTVDescriptorSize ( ) const
```

Returns a constant reference to the render target view descriptor size.

5.4.3.23 GetSampleCount() [1/2]

```
UINT & FARender::DeviceResources::GetSampleCount ( )
```

Returns a reference to the sample count.

5.4.3.24 GetSampleCount() [2/2]

```
const UINT & FARender::DeviceResources::GetSampleCount ( ) const
```

Returns a constant reference to the sample count.

5.4.3.25 GetScissor()

```
const D3D12_RECT & FARender::DeviceResources::GetScissor ( ) const
```

Returns a constant reference to the D3D12_RECT scissor objcet.

5.4.3.26 GetSwapChain()

```
const Microsoft::WRL::ComPtr< IDXGISwapChain1 > & FARender::DeviceResources::GetSwapChain ( )
const
```

Returns a constant reference to the IDXGISwapChain1 object.

5.4.3.27 GetSwapChainBuffers()

```
const Microsoft::WRL::ComPtr< ID3D12Resource > * FARender::DeviceResources::GetSwapChain↩
Buffers ( ) const
```

Returns a pointer to the swap chain buffers.
There are two swap chain buffers.
To access each buffer do swapChainBuffers()[i], where i is the index of the buffer you want to access.

5.4.3.28 GetViewport()

```
const D3D12_VIEWPORT & FARender::DeviceResources::GetViewport ( ) const
```

Returns a constant reference to the D3D12_VIEWPORT objcet.

5.4.3.29 InitializeDirect3D()

```
void FARender::DeviceResources::InitializeDirect3D (
            unsigned int width,
            unsigned int height,
            HWND handle )
```

Initializes Direct3D. Enables the debug layer if in debug mode.
Creates a Direct3D 12 device.
Creates a DXGI factory object.
Creates a fence.
Queries descriptor sizes.
Creates command objects.
Creates a swap chain.
Creates a render target view and a depth/stencil view heap. Creates the initial render target buffers, depth stencil buffer, MSAA buffers and text buffers.

5.4.3.30 IsMSAAEnabled()

```
bool FARender::DeviceResources::IsMSAAEnabled ( )
```

Returns true if MSAA is enabled, false otherwise.

5.4.3.31 Present()

```
void FARender::DeviceResources::Present ( )
```

Swaps the front and back buffers.

5.4.3.32 ResetCommandAllocator()

```
void FARender::DeviceResources::ResetCommandAllocator ( )
```

Resets command allocator to allow reuse of the memory.

5.4.3.33 ResetCommandList()

```
void FARender::DeviceResources::ResetCommandList ( )
```

Resets the command list to open it with a current frame command allocator.

5.4.3.34 Resize()

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

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

5.4.3.35 RTBufferTransition()

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

Transistions the render target buffer.

5.4.3.36 Signal()

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

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

5.4.3.37 UpdateCurrentFrameFenceValue()

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

Updates the current frames fence value.

5.4.3.38 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

5.5 DirectXException Class Reference

Public Member Functions

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

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

- FADirectXException.h

5.6 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** = D3D_PRIMITIVE_TOPOLOGY_TRIANGLELIST
- std::vector< FAShapes::DrawArguments > **drawArgs**

5.6.1 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

5.7 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.

5.7.1 Detailed Description

This class stores indices in a Direct3D 12 default buffer.

5.7.2 Member Function Documentation

5.7.2.1 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.

5.7.2.2 CreateIndexBufferView()

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

Creates the vertex buffer view and stores it.

5.7.2.3 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

5.8 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 handle)
- **RenderScene** (const RenderScene &)=delete
- RenderScene & **operator=** (const RenderScene &)=delete
- DeviceResources & **GetDeviceResources** ()
- const DeviceResources & **GetDeviceResources** () const
- const Microsoft::WRL::ComPtr< ID3DBlob > & **GetShader** (const std::wstring &name) const
- const std::vector< D3D12_INPUT_ELEMENT_DESC > & **GetInputElementLayout** (const std::wstring &name) const
- const D3D12_RASTERIZER_DESC & **GetRasterizationState** (const std::wstring &name) const
- const Microsoft::WRL::ComPtr< ID3D12PipelineState > & **GetPSO** (const std::wstring &drawSettingsName) const
- const Microsoft::WRL::ComPtr< ID3D12RootSignature > & **GetRootSignature** (const std::wstring &drawSettingsName) const
- const D3D_PRIMITIVE_TOPOLOGY & **GetPrimitive** (const std::wstring &drawSettingsName) const
- FAShapes::DrawArguments & **GetDrawArguments** (const std::wstring &drawSettingsName, unsigned int index)
- const FAShapes::DrawArguments & **GetDrawArguments** (const std::wstring &drawSettingsName, unsigned int index) const
- ConstantBuffer & **GetConstantBuffer** ()
- const ConstantBuffer & **GetConstantBuffer** () const
- const UINT & GetCBVSize () const

  Returns a constant reference to the CBV/SRV/UAV descriptor size.
- const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > & GetCBVHeap () const

  Returns a constant reference to the CBV descriptor heap.
- const D3D12_ROOT_PARAMETER & GetCBVHeapRootParameter () const

  Returns a constant reference to the CBV's heap root parameter.
- void **LoadShader** (const std::wstring &filename, const std::wstring &name)
- void **RemoveShader** (const std::wstring &shaderName)
- void **StoreInputElementDescriptions** (const std::wstring &name, const std::vector< D3D12_INPUT_ELEMENT_DESC > &inputElementLayout)
- void **StoreInputElementDescriptions** (const std::wstring &name, const D3D12_INPUT_ELEMENT_DESC *inputElementLayout, UINT numElements)
- void **RemoveInputElementDescription** (const std::wstring &name)
- void **CreateRasterizationState** (D3D12_FILL_MODE fillMode, BOOL enableMultisample, const std::wstring &name)
- void **RemoveRasterizationState** (const std::wstring &name)
- void **CreatePSO** (const std::wstring &drawSettingsName, const std::wstring &rStateName, const std::wstring &vsName, const std::wstring &psName, const std::wstring &inputLayoutName, const D3D12_PRIMITIVE_TOPOLOGY_TYPE &primitiveType, UINT sampleCount)
- void **CreateRootSignature** (const std::wstring &drawSettingsName, const D3D12_ROOT_PARAMETER *rootParameters, UINT numParameters)
- void **CreateVertexBuffer** (const void *data, UINT numBytes, UINT stride)
- void CreateIndexBuffer (const void *data, UINT numBytes, DXGI_FORMAT format)

  Creates an index buffer with the specified name and stores all of given data in the index buffer. 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 (const std::wstring &drawSettingsName, const Microsoft::WRL::ComPtr< ID3D12PipelineState > &pso)

  Sets the PSO in the specified DrawSettings structure to the specified pso. If the specifed DrawSettings structure does not exist an out_of_range exception is thrown.
- void SetRootSignature (const std::wstring &drawSettingsName, const Microsoft::WRL::ComPtr< ID3D12RootSignature > &rootSignature)

  Sets the root signature in the specified DrawSettings structure to the specified root signature. If the specifed DrawSettings structure does not exist an out_of_range exception is thrown.
- void SetPrimitive (const std::wstring &drawSettingsName, const D3D_PRIMITIVE_TOPOLOGY &primitive)

  Sets the Primitive in the specified DrawSettings structure to the specified primitive. If the specifed DrawSettings structure does not exist an out_of_range exception is thrown.
- void AddDrawArgument (const std::wstring &drawSettingsName, const FAShapes::DrawArguments &drawArg)

  Adds the specified draw argument structure to the DrawArguments vector of the specified DrawSettings structure. If the specifed DrawSettings structure does not exist an out_of_range exception is thrown.
- void AddDrawArgument (const std::wstring &drawSettingsName, unsigned int indexCount, unsigned int locationOfFirstIndex, int indexOfFirstVertex, int indexOfConstantData)

  Adds the specified draw arguments to the DrawArguments vector of the specified DrawSettings structure. If the specifed DrawSettings structure does not exist an out_of_range exception is thrown.
- void RemoveDrawArgument (const std::wstring &drawSettingsName, unsigned int index)

  Removes the draw argument in the specified DrawSettings structure at the specified index. If the DrawSettings does not exist or if the index is out of bounds an out_of_range exception is thrown.
- void CreateDrawSettings (const std::wstring &drawSettingsName)

  Creates a DrawSettings structure with the specified name.
- void RemoveDrawSettings (const std::wstring &drawSettingsName)

  Removes the specified DrawSettings structure. If the DrawSettings structure does not exist an out_of_range exception is thrown.
- void CreateText (const std::wstring &textName, FAMath::Vector4D textLocation, const std::wstring &textString, float textSize, const FAColor::Color textColor)

  Creates a Text object with the specified properties and stores it with the specified name. 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 (const std::wstring &textName)

  Removes the specified text object with the specified name. If the Text object does not exist an out_of_range exception is thrown.
- void ChangeTextLocation (const std::wstring &textName, FAMath::Vector4D textLocation)

  Changes the text location of the specified Text object. If the Text object does not exist an out_of_range exception is thrown.
  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 ChangeTextString (const std::wstring &textName, const std::wstring &textString)

  Changes the text string of the specified Text object. If the Text object does not exist an out_of_range exception is thrown.
- void ChangeTextSize (const std::wstring &textName, float textSize)

  Changes the text size of the specified Text object. If the Text object does not exist an out_of_range exception is thrown.
- void ChangeTextColor (const std::wstring &textName, const FAColor::Color textColor)

  Changes the text color of the specified Text object. If the Text object does not exist an out_of_range exception is thrown.

- 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 (const std::wstring &drawSettingsName)

  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 (bool renderText)

  Transitions the render target buffer to the correct state and excutes all the beforeDrawObjects and drawObjects commands. Pass in true if you are going to render text, false otherwise. Call after calling all the drawObjects functions.
- void BeforeDrawText ()

  Puts all of the commands needed in the command list before drawing the text of the scene. Call before calling the first drawText function.
- void RenderText (const std::wstring &textName)

  Draws the specified Text object. Call in between a beforeDrawObjects function and a afterDrawObjects function.

  .
- void AfterDrawText ()

  Transitions the render target buffer and executes all of the text drawing commands.
- 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.

5.8.1 Detailed Description

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

5.8.2 Member Function Documentation

5.8.2.1 AddDrawArgument() [1/2]

```
void FARender::RenderScene::AddDrawArgument (
            const std::wstring & drawSettingsName,
            const FAShapes::DrawArguments & drawArg )
```

Adds the specified draw argument structure to the DrawArguments vector of the specified DrawSettings structure. If the specifed DrawSettings structure does not exist an out_of_range exception is thrown.

5.8.2.2 AddDrawArgument() [2/2]

```
void FARender::RenderScene::AddDrawArgument (
            const std::wstring & drawSettingsName,
            unsigned int indexCount,
            unsigned int locationOfFirstIndex,
            int indexOfFirstVertex,
            int indexOfConstantData )
```

Adds the specified draw arguments to the DrawArguments vector of the specified DrawSettings structure. If the specifed DrawSettings structure does not exist an out_of_range exception is thrown.

5.8.2.3 AfterDraw()

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

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

5.8.2.4 AfterDrawObjects()

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

Transitions the render target buffer to the correct state and excutes all the beforeDrawObjects and drawObjects commands. Pass in true if you are going to render text, false otherwise. Call after calling all the drawObjects functions.

5.8.2.5 AfterDrawText()

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

Transitions the render target buffer and executes all of the text drawing commands.

5.8.2.6 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.

5.8.2.7 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 drawText function.

5.8.2.8 ChangeTextColor()

```
void FARender::RenderScene::ChangeTextColor (
            const std::wstring & textName,
            const FAColor::Color textColor )
```

Changes the text color of the specified Text object. If the Text object does not exist an out_of_range exception is thrown.

5.8.2.9 ChangeTextLocation()

```
void FARender::RenderScene::ChangeTextLocation (
            const std::wstring & textName,
            FAMath::Vector4D textLocation )
```

Changes the text location of the specified Text object. If the Text object does not exist an out_of_range exception is thrown.
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.

5.8.2.10 ChangeTextSize()

```
void FARender::RenderScene::ChangeTextSize (
            const std::wstring & textName,
            float textSize )
```

Changes the text size of the specified Text object. If the Text object does not exist an out_of_range exception is thrown.

5.8.2.11 ChangeTextString()

```
void FARender::RenderScene::ChangeTextString (
            const std::wstring & textName,
            const std::wstring & textString )
```

Changes the text string of the specified Text object. If the Text object does not exist an out_of_range exception is thrown.

5.8.2.12 CreateCBVHeap()

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

Creates the CBV heap.

5.8.2.13 CreateConstantBuffer()

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

Creates a constant buffer for each frame.

5.8.2.14 CreateConstantBufferView()

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

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

5.8.2.15 CreateDrawSettings()

```
void FARender::RenderScene::CreateDrawSettings (
            const std::wstring & drawSettingsName )
```

Creates a DrawSettings structure with the specified name.

5.8.2.16 CreateIndexBuffer()

```
void FARender::RenderScene::CreateIndexBuffer (
            const void * data,
            UINT numBytes,
            DXGI_FORMAT format )
```

Creates an index buffer with the specified name and stores all of given data in the index buffer. Also creates a view to the index buffer.
Execute commands and flush the command queue after calling createVertexBuffer() and createIndexBuffer().

5.8.2.17 CreateText()

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

Creates a Text object with the specified properties and stores it with the specified name. 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.

5.8.2.18 DrawObjects()

```
void FARender::RenderScene::DrawObjects (
            const std::wstring & drawSettingsName )
```

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 specified DrawSettings structure does not exist.

5.8.2.19 ExecuteAndFlush()

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

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

5.8.2.20 GetCBVHeap()

```
const Microsoft::WRL::ComPtr< ID3D12DescriptorHeap > & FARender::RenderScene::GetCBVHeap ( )
const
```

Returns a constant reference to the CBV descriptor heap.

5.8.2.21 GetCBVHeapRootParameter()

```
const D3D12_ROOT_PARAMETER & FARender::RenderScene::GetCBVHeapRootParameter ( ) const
```

Returns a constant reference to the CBV's heap root parameter.

5.8.2.22 GetCBVSize()

```
const UINT & FARender::RenderScene::GetCBVSize ( ) const
```

Returns a constant reference to the CBV/SRV/UAV descriptor size.

5.8.2.23 RemoveDrawArgument()

```
void FARender::RenderScene::RemoveDrawArgument (
            const std::wstring & drawSettingsName,
            unsigned int index )
```

Removes the draw argument in the specified DrawSettings structure at the specified index. If the DrawSettings does not exist or if the index is out of bounds an out_of_range exception is thrown.

5.8.2.24 RemoveDrawSettings()

```
void FARender::RenderScene::RemoveDrawSettings (
            const std::wstring & drawSettingsName )
```

Removes the specified DrawSettings structure. If the DrawSettings structure does not exist an out_of_range exception is thrown.

5.8.2.25 RemoveText()

```
void FARender::RenderScene::RemoveText (
            const std::wstring & textName )
```

Removes the specified text object with the specified name. If the Text object does not exist an out_of_range exception is thrown.

5.8.2.26 RenderText()

```
void FARender::RenderScene::RenderText (
            const std::wstring & textName )
```

Draws the specified Text object. Call in between a beforeDrawObjects function and a afterDrawObjects function.
.

Ex.
beforeDrawText()
drawText()
drawText()
afterDrawText()
Throws an out_of_range exception if the specified Text object does not exist.

5.8.2.27 SetPrimitive()

```
void FARender::RenderScene::SetPrimitive (
            const std::wstring & drawSettingsName,
            const D3D_PRIMITIVE_TOPOLOGY & primitive )
```

Sets the Primitive in the specified DrawSettings structure to the specified primitive. If the specifed DrawSettings structure does not exist an out_of_range exception is thrown.

5.8.2.28 SetPSO()

```
void FARender::RenderScene::SetPSO (
            const std::wstring & drawSettingsName,
            const Microsoft::WRL::ComPtr< ID3D12PipelineState > & pso )
```

Sets the PSO in the specified DrawSettings structure to the specified pso. If the specifed DrawSettings structure does not exist an out_of_range exception is thrown.

5.8.2.29 SetRootSignature()

```
void FARender::RenderScene::SetRootSignature (
            const std::wstring & drawSettingsName,
            const Microsoft::WRL::ComPtr< ID3D12RootSignature > & rootSignature )
```

Sets the root signature in the specified DrawSettings structure to the specified root signature. If the specifed DrawSettings structure does not exist an out_of_range exception is thrown.

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

- FARenderScene.h

5.9 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 ()

  Default Constructor.
- Text (const DeviceResources &deviceResources, 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.
- void Initialize (const DeviceResources &deviceResources, const FAMath::Vector4D &textLocation, const std::wstring &textString, float textSize, const FAColor::Color &textColor)

  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 Microsoft::WRL::ComPtr< ID2D1SolidColorBrush > & GetBrush () const

  Returns a constant reference to the color brush.
- const Microsoft::WRL::ComPtr< IDWriteTextFormat > & GetFormat () const

  Returns a constant reference to the format of the text.
- const FAColor::Color & GetTextColor () const

  Returns a constant reference to the text color.
- void SetTextSize (const DeviceResources &deviceResources, 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.

5.9.1 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.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 Text() [1/2]

```
FARender::Text::Text ( )
```

Default Constructor.

5.9.2.2 Text() [2/2]

```
FARender::Text::Text (
            const DeviceResources & deviceResources,
            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.

5.9.3 Member Function Documentation

5.9.3.1 GetBrush()

```
const Microsoft::WRL::ComPtr< ID2D1SolidColorBrush > & FARender::Text::GetBrush ( ) const
```

Returns a constant reference to the color brush.

5.9.3.2 GetFormat()

```
const Microsoft::WRL::ComPtr< IDWriteTextFormat > & FARender::Text::GetFormat ( ) const
```

Returns a constant reference to the format of the text.

5.9.3.3 GetTextColor()

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

Returns a constant reference to the text color.

5.9.3.4 GetTextLocation()

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

Returns a constant reference to the text location.

5.9.3.5 GetTextSize()

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

Returns the text size.

5.9.3.6 GetTextString()

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

Returns a constant reference to the text string.

5.9.3.7 Initialize()

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

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.

5.9.3.8 SetTextColor()

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

Changes the text color to the specified color.

5.9.3.9 SetTextLocation()

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

Changes the text location to the specified location.

5.9.3.10 SetTextSize()

```
void FARender::Text::SetTextSize (
            const DeviceResources & deviceResources,
            float textSize )
```

Changes the text size to the specified size.

5.9.3.11 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

5.10 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.

5.10.1 Constructor & Destructor Documentation

5.10.1.1 Time()

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

Default Constructor. Gets and stores the seconds per count.

5.10.2 Member Function Documentation

5.10.2.1 DeltaTime()

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

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

5.10.2.2 Reset()

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

Resets all time variables.

5.10.2.3 Start()

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

Starts the timer.

5.10.2.4 Stop()

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

Stops the timer.

5.10.2.5 Tick()

```
void FATime::Time::Tick ( )
```

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

5.10.2.6 TotalTime()

```
float FATime::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:

- FATime.h

5.11 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 "FATime.h"
```

5.11.1 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:

- FATime.h

5.12 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.

5.12.1 Detailed Description

This class stores vertices in a Direct3D 12 default buffer.

5.12.2 Member Function Documentation

5.12.2.1 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.

5.12.2.2 CreateVertexBufferView()

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

Creates the vertex buffer view and stores it.

5.12.2.3 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

5.13 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 *additional↩ Data=nullptr)

  Creates and displays a window. Registers a default window class with the OS with the specified instance, class name and window procdure.
- Window (const HINSTANCE &hInstance, const WNDCLASSEX &windowClass, const std::wstring &window↩ Name, 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 o the specified height.

5.13.1 Detailed Description

The window class is used to make a Window using Windows API.

5.13.2 Constructor & Destructor Documentation

5.13.2.1 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 procdure.

5.13.2.2 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.

5.13.3 Member Function Documentation

5.13.3.1 GetHeight()

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

Returns the height of the window.

5.13.3.2 GetWidth()

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

Returns the width of the window.

5.13.3.3 GetWindowHandle()

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

Returns the window handle.

5.13.3.4 SetHeight()

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

Sets the height of the window o the specified height.

5.13.3.5 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::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 buffers.

Namespaces

- namespace FARender

  The namespace has utility functions and structs, VertexBuffer, IndexBuffer, ConstantBuffer, DeviceResources, RenderScene and Text classes.

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
7 #include <wrl.h>
8 #include <d3d12.h>
9
10 namespace FARender
11 {
16     class VertexBuffer
17     {
18     public:
19         VertexBuffer() = default;
20         VertexBuffer(const VertexBuffer&) = delete;
21         VertexBuffer& operator=(const VertexBuffer&) = delete;
22
25         void CreateVertexBuffer(const Microsoft::WRL::ComPtr<ID3D12Device>& device,
26             const Microsoft::WRL::ComPtr<ID3D12GraphicsCommandList>& commandList, const void* data, UINT
    numBytes);
27
30         void CreateVertexBufferView(UINT numBytes, UINT stride);
31
34         const D3D12_VERTEX_BUFFER_VIEW& GetVertexBufferView();
35
36     private:
37         Microsoft::WRL::ComPtr<ID3D12Resource> mVertexDefaultBuffer;
38         Microsoft::WRL::ComPtr<ID3D12Resource> mVertexUploadBuffer;
39         D3D12_VERTEX_BUFFER_VIEW mVertexBufferView{};
40     };
41
46     class IndexBuffer
47     {
48     public:
49         IndexBuffer() = default;
50         IndexBuffer(const IndexBuffer&) = delete;
51         IndexBuffer& operator=(const IndexBuffer&) = delete;
52
55         const D3D12_INDEX_BUFFER_VIEW& GetIndexBufferView();
56
59         void CreateIndexBuffer(const Microsoft::WRL::ComPtr<ID3D12Device>& device,
60             const Microsoft::WRL::ComPtr<ID3D12GraphicsCommandList>& commandList, const void* data, UINT
    numBytes);
61
64         void CreateIndexBufferView(UINT numBytes, DXGI_FORMAT format);
65
66     private:
67         Microsoft::WRL::ComPtr<ID3D12Resource> mIndexDefaultBuffer;
68         Microsoft::WRL::ComPtr<ID3D12Resource> mIndexUploadBuffer;
69         D3D12_INDEX_BUFFER_VIEW mIndexBufferView;
70     };
71
76     class ConstantBuffer
77     {
78     public:
79         ConstantBuffer() = default;
80
81         ConstantBuffer(const ConstantBuffer&) = delete;
82         ConstantBuffer& operator=(const ConstantBuffer&) = delete;
83
86         ~ConstantBuffer();
87
90         Microsoft::WRL::ComPtr<ID3D12Resource>& GetConstantBuffer();
91
94         const Microsoft::WRL::ComPtr<ID3D12Resource>& GetConstantBuffer() const;
95
99         void CreateConstantBuffer(const Microsoft::WRL::ComPtr<ID3D12Device>& device, const UINT&
    numOfBytes);
100
103         void CreateConstantBufferView(const Microsoft::WRL::ComPtr<ID3D12Device>& device,
104             const Microsoft::WRL::ComPtr<ID3D12DescriptorHeap>& cbvHeap, UINT cbvSize, UINT
    cBufferIndex,
105             UINT cbvHeapIndex, UINT numBytes);
106
110         void CopyData(UINT index, UINT byteSize, const void* data, const UINT64& numOfBytes);
```

```
111
112     private:
113         Microsoft::WRL::ComPtr<ID3D12Resource> mConstantBuffer;
114         BYTE* mMappedData{ nullptr };
115     };
116 }
```

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 velcoity);
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 Foward(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 addtion. 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 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.
.

6.6.2.2 operator∗() [2/3]

```
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.
.

6.6.2.3 operator∗() [3/3]

```
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.
.

6.6.2.4 operator+()

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

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

6.6.2.5 operator-()

```
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.

6.7 FAColor.h

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "FAMathEngine.h"
4
9 namespace FAColor
10 {
16     class Color
17     {
18     public:
19
23         Color(float r = 0.0f, float g = 0.0f, float b = 0.0f, float a = 1.0f);
24
28         Color(const FAMath::Vector4D& color);
29
32         const FAMath::Vector4D& GetColor() const;
33
36         float GetRed() const;
37
40         float GetGreen() const;
41
44         float GetBlue() const;
45
48         float GetAlpha() const;
49
52         void SetColor(const FAMath::Vector4D& color);
53
56         void SetRed(float r);
57
60         void SetGreen(float g);
61
64         void SetBlue(float b);
65
68         void SetAlpha(float a);
69
73         Color& operator+=(const Color& c);
74
78         Color& operator-=(const Color& c);
79
84         Color& operator*=(float k);
85
90         Color& operator*=(const Color& c);
91
92     private:
93         FAMath::Vector4D mColor;
94     };
95
```

```
99     Color operator+(const Color& c1, const Color& c2);
100
104     Color operator-(const Color& c1, const Color& c2);
105
110     Color operator*(const Color& c, float k);
111
116     Color operator*(float k, const Color& c);
117
121     Color operator*(const Color& c1, const Color& c2);
122 }
```

6.8 FADeviceResources.h File Reference

File has class DeviceResources under namespace FARender.

```
#include <wrl.h>
#include <d3d12.h>
#include <dxgi1_4.h>
#include <d3d11.h>
#include <d3d11on12.h>
#include <d2d1_3.h>
#include <dwrite.h>
#include <vector>
#include "FARenderingUtility.h"
```

Classes

- class FARender::DeviceResources

  A wrapper for a Direct3D 12 device, swapchain, depth buffer, MSAA buffers and command objects.

Namespaces

- namespace FARender

  The namespace has utility functions and structs, VertexBuffer, IndexBuffer, ConstantBuffer, DeviceResources, RenderScene and Text classes.

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
7 #include <wrl.h>
8 #include <d3d12.h>
9 #include <dxgi1_4.h>
10 #include <d3d11.h>
11 #include <d3d11on12.h>
12 #include <d2d1_3.h>
13 #include <dwrite.h>
14 #include <vector>
15 #include "FARenderingUtility.h"
```

```
16
17 namespace FARender
18 {
22     class DeviceResources
23     {
24     public:
25         DeviceResources() = default;
26
27         DeviceResources(unsigned int width, unsigned int height, HWND windowHandle);
28
29         DeviceResources(const DeviceResources&) = delete;
30         DeviceResources& operator=(const DeviceResources&) = delete;
31
34         ~DeviceResources();
35
38         const Microsoft::WRL::ComPtr<ID3D12Device>& GetDevice() const;
39
42         const Microsoft::WRL::ComPtr<ID3D12CommandQueue>& GetCommandQueue() const;
43
46         const Microsoft::WRL::ComPtr<ID3D12CommandAllocator>& GetCommandAllocator() const;
47
50         const Microsoft::WRL::ComPtr<ID3D12GraphicsCommandList>& GetCommandList() const;
51
54         const DXGI_FORMAT& GetBackBufferFormat() const;
55
58         const UINT GetNumOfSwapChainBuffers() const;
59
62         const Microsoft::WRL::ComPtr<IDXGISwapChain1>& GetSwapChain() const;
63
66         const UINT& GetRTVDescriptorSize() const;
67
70         const UINT& GetDSVDescriptorSize() const;
71
74         const Microsoft::WRL::ComPtr<ID3D12DescriptorHeap>& GetRTVDescriptorHeap() const;
75
78         const Microsoft::WRL::ComPtr<ID3D12DescriptorHeap>& GetDSVDescriptorHeap() const;
79
82         const UINT& GetCurrentBackBuffer() const;
83
88         const Microsoft::WRL::ComPtr<ID3D12Resource>* GetSwapChainBuffers() const;
89
92         const Microsoft::WRL::ComPtr<ID3D12Resource>& GetDepthStencilBuffer() const;
93
96         const DXGI_FORMAT& GetDepthStencilFormat() const;
97
100         const D3D12_VIEWPORT& GetViewport() const;
101
104         const D3D12_RECT& GetScissor() const;
105
108         bool IsMSAAEnabled();
109
112         void DisableMSAA();
113
116         void EnableMSAA();
117
120         UINT& GetSampleCount();
121
124         const UINT& GetSampleCount() const;
125
128         const UINT64& GetCurrentFenceValue() const;
129
132         const Microsoft::WRL::ComPtr<ID2D1DeviceContext>& GetDevice2DContext() const;
133
136         const Microsoft::WRL::ComPtr<IDWriteFactory>& GetDirectWriteFactory() const;
137
140         void UpdateCurrentFrameFenceValue();
141
153         void InitializeDirect3D(unsigned int width, unsigned int height, HWND handle);
154
159         void FlushCommandQueue();
160
164         void WaitForGPU() const;
165
168         void Signal();
169
173         void Resize(int width, int height, const HWND& handle);
174
177         void ResetCommandList();
178
179         /*@brief Resets the command list to open it with the direct command allocator.
180 */
181         void ResetDirectCommandList();
182
185         void ResetCommandAllocator();
186
189         void RTBufferTransition(bool renderText);
190
```

```
193         void BeforeTextDraw();
194
197         void AfterTextDraw();
198
201         void Execute() const;
202
205         void Present();
206
207         /*@brief Calls the necessary functions to let the user draw their objects.
208 */
209         void Draw();
210
211     private:
212         Microsoft::WRL::ComPtr<ID3D12Device> mDirect3DDevice;
213
214         Microsoft::WRL::ComPtr<IDXGIFactory4> mDXGIFactory;
215
216         Microsoft::WRL::ComPtr<ID3D12Fence> mFence;
217         UINT64 mFenceValue{ 0 };
218         UINT64 mCurrentFrameFenceValue[numFrames];
219
220         Microsoft::WRL::ComPtr<ID3D12CommandQueue> mCommandQueue;
221         Microsoft::WRL::ComPtr<ID3D12CommandAllocator> mCommandAllocator[numFrames];
222         Microsoft::WRL::ComPtr<ID3D12CommandAllocator> mDirectCommandAllocator;
223         Microsoft::WRL::ComPtr<ID3D12GraphicsCommandList> mCommandList;
224
225         DXGI_FORMAT mBackBufferFormat{ DXGI_FORMAT_R8G8B8A8_UNORM };
226         static const UINT mNumOfSwapChainBuffers{ 2 };
227         UINT mCurrentBackBuffer{ 0 };
228         Microsoft::WRL::ComPtr<IDXGISwapChain1> mSwapChain;
229         Microsoft::WRL::ComPtr<ID3D12Resource> mSwapChainBuffers[mNumOfSwapChainBuffers];
230
231         Microsoft::WRL::ComPtr<ID3D12Resource> mDepthStencilBuffer;
232         DXGI_FORMAT mDepthStencilFormat{ DXGI_FORMAT_D24_UNORM_S8_UINT };
233
234         UINT mRTVSize;
235         UINT mDSVSize;
236         Microsoft::WRL::ComPtr<ID3D12DescriptorHeap> mRTVHeap;
237         Microsoft::WRL::ComPtr<ID3D12DescriptorHeap> mDSVHeap;
238
239         D3D12_VIEWPORT mViewport;
240         D3D12_RECT mScissor;
241
242         bool mMSAA4xSupported = false;
243         bool mIsMSAAEnabled = false;
244         UINT mSampleCount{ 4 };
245         Microsoft::WRL::ComPtr<ID3D12DescriptorHeap> mMSAARTVDescriptorHeap;
246         Microsoft::WRL::ComPtr<ID3D12DescriptorHeap> mMSAADSVDescriptorHeap;
247         Microsoft::WRL::ComPtr<ID3D12Resource> mMSAARenderTargetBuffer;
248         Microsoft::WRL::ComPtr<ID3D12Resource> mMSAADepthStencilBuffer;
249
250         Microsoft::WRL::ComPtr<ID3D11Device> mDevice11;
251         Microsoft::WRL::ComPtr<ID3D11DeviceContext> mDevice11Context;
252         Microsoft::WRL::ComPtr<ID3D11On12Device> mDevice11on12;
253
254         Microsoft::WRL::ComPtr<ID2D1Device2> mDirect2DDevice;
255         Microsoft::WRL::ComPtr<ID2D1Factory3> mDirect2DFactory;
256         Microsoft::WRL::ComPtr<ID2D1DeviceContext> mDirect2DDeviceContext;
257
258         Microsoft::WRL::ComPtr<IDWriteFactory> mDirectWriteFactory;
259
260         std::vector<Microsoft::WRL::ComPtr<ID3D11Resource>> mWrappedBuffers;
261         std::vector<Microsoft::WRL::ComPtr<ID2D1Bitmap1>> mDirect2DBuffers;
262         std::vector<Microsoft::WRL::ComPtr<IDXGISurface>> mSurfaces;
263
264         //Call all of these functions to initialize Direct3D
265         void EnableDebugLayer();
266         void CreateDirect3DDevice();
267         void CreateDXGIFactory();
268         void CreateFence();
269         void QueryDescriptorSizes();
270         void CreateCommandObjects();
271         void CreateSwapChain(HWND handle);
272         void CreateRTVHeap();
273         void CreateDSVHeap();
274
275         //if MSAA is supported, creates a MSAA RTV and DSV heap.
276         void CheckMSAASupport();
277         void CreateMSAARTVHeap();
278         void CreateMSAADSVHeap();
279
280         //Creates and initializes everything needed to render text.
281         void InitializeText();
282
283         //These functions are for creating swap chain buffers, depth/stencil buffer, render target views
    and depth/stencil view.
284         //They are called in the resize function.
```

```
285         void CreateRenderTargetBufferAndView();
286         void CreateDepthStencilBufferAndView(int width, int height);
287
288         //These functions are for creating a MSAA render target buffer, MSAA depth/stencil buffer,
289         //MSAA render target view, and a MSAA depth/stencil view.
290         //They are called in the resize function.
291         void CreateMSAARenderTargetBufferAndView(int width, int height);
292         void CreateMSAADepthStencilBufferAndView(int width, int height);
293
294         /* Resets the text buffers.
295 * Gets called in the resize function.
296 */
297         void ResetTextBuffers();
298
299         /*Resizes the necessary text buffers.
300 * Gets called in the resize function.
301 */
302         void TextResize(const HWND& handle);
303     };
304 }
```

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 FARenderingUtility.h File Reference

File has static variables numFrames and current frame, function nextFrame() and struct DrawArguments under the namespace FARender.

Namespaces

- namespace FARender

  The namespace has utility functions and structs, VertexBuffer, IndexBuffer, ConstantBuffer, DeviceResources, RenderScene and Text classes.

Functions

- void FARender::nextFrame ()

  Update our current frame value to go to the next frame.

6.11.1 Detailed Description

File has static variables numFrames and current frame, function nextFrame() and struct DrawArguments under the namespace FARender.

6.12 FARenderingUtility.h

Go to the documentation of this file.

```
1 #pragma once
2
12 namespace FARender
13 {
14     static const unsigned int numFrames{ 3 };
15     static unsigned int currentFrame{ 0 };
16
19     void nextFrame();
20 }
```

6.13 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 "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

  The namespace has utility functions and structs, VertexBuffer, IndexBuffer, ConstantBuffer, DeviceResources, RenderScene and Text classes.

6.13.1 Detailed Description

File has class RenderScene under namespace FARender.

6.14 FARenderScene.h

Go to the documentation of this file.

```
1 #pragma once
2
7 #include <d3dcompiler.h>
8 #include <unordered_map>
9 #include <string>
10 #include "FADeviceResources.h"
11 #include "FABuffer.h"
12 #include "FAText.h"
13 #include "FAShapesUtility.h"
14
15 namespace FARender
16 {
20     struct DrawSettings
21     {
22         Microsoft::WRL::ComPtr<ID3D12PipelineState> pipelineState;
23         Microsoft::WRL::ComPtr<ID3D12RootSignature> rootSig;
24         D3D_PRIMITIVE_TOPOLOGY prim = D3D_PRIMITIVE_TOPOLOGY_TRIANGLELIST;
25         std::vector<FAShapes::DrawArguments> drawArgs;
26     };
27
28
32     class RenderScene
33     {
34     public:
35
36         RenderScene(unsigned int width, unsigned int height, HWND handle);
37
38         RenderScene(const RenderScene&) = delete;
39         RenderScene& operator=(const RenderScene&) = delete;
40
41         /*@brief Returns a reference to the device resources object.
42 */
43         DeviceResources& GetDeviceResources();
44
45         /*@brief Returns a constant reference to the device resources object.
46 */
47         const DeviceResources& GetDeviceResources() const;
48
49         /*@brief Returns a constant reference to the shader with the specified name.
50 * Throws an out_of_range exception if the shader does not exist.
51 */
52         const Microsoft::WRL::ComPtr<ID3DBlob>& GetShader(const std::wstring& name) const;
53
54         /*@brief Returns a constant reference to the specified array of input element layout
    descriptions.
55 * Throws an out_of_range exception if the array of input element layout descriptions does not exist.
56 */
57         const std::vector<D3D12_INPUT_ELEMENT_DESC>& GetInputElementLayout(const std::wstring& name)
    const;
58
59         /*@brief Returns a constant reference to the specified rasterization description.
60 * Throws an out_of_range exception if the rasterization description does not exist.
61 */
62         const D3D12_RASTERIZER_DESC& GetRasterizationState(const std::wstring& name) const;
63
64         /*@brief Returns a constant reference to the PSO in the specified DrawSettings.
65 * Throws an out_of_range exception if the DrawSettings does not exist.
66 */
67         const Microsoft::WRL::ComPtr<ID3D12PipelineState>& GetPSO(const std::wstring& drawSettingsName)
    const;
68
69         /*@brief Returns a constant reference to the root signature in the specified DrawSettings
    structure.
70 * Throws an out_of_range exception if the DrawSettings does not exist.
71 */
72         const Microsoft::WRL::ComPtr<ID3D12RootSignature>& GetRootSignature(const std::wstring&
    drawSettingsName) const;
73
74         /*@brief Returns a constant reference to the primitive in the specified DrawSettings structure.
75 * Throws an out_of_range exception if the DrawSettings does not exist.
76 */
77         const D3D_PRIMITIVE_TOPOLOGY& GetPrimitive(const std::wstring& drawSettingsName) const;
```

```
78
79          /*@brief Returns a reference to the specified DrawArguments object in the specified DrawSettings
    structure.
80 * Throws an out_of_range exception if the DrawSettings does not exist or if the index is out of range.
81 */
82          FAShapes::DrawArguments& GetDrawArguments(const std::wstring& drawSettingsName, unsigned int
    index);
83
84          /*@brief Returns a constant reference to the specified DrawArguments object in the specified
    DrawSettings structure.
85 * Throws an out_of_range exception if the DrawSettings does not exist or if the index is out of range.
86 */
87          const FAShapes::DrawArguments& GetDrawArguments(const std::wstring& drawSettingsName, unsigned
    int index) const;
88
89          /*@brief Returns a reference to the constant buffer with the specified name.
90 */
91          ConstantBuffer& GetConstantBuffer();
92
93          /*@brief Returns a constant reference to the constant buffer with the specified name.
94 */
95          const ConstantBuffer& GetConstantBuffer() const;
96
99          const UINT& GetCBVSize() const;
100
103          const Microsoft::WRL::ComPtr<ID3D12DescriptorHeap>& GetCBVHeap() const;
104
107          const D3D12_ROOT_PARAMETER& GetCBVHeapRootParameter() const;
108
109          /*@brief Loads a shader's bytecode and stores it with the specified name.
110 */
111          void LoadShader(const std::wstring& filename, const std::wstring& name);
112
113          /*@brief Removes the specified shader.
114 * If the specified shader does not exist an out_of_range exception is thrown.
115 */
116          void RemoveShader(const std::wstring& shaderName);
117
118          /*@brief Stores an array of input element descriptions with the specified name.
119 */
120          void StoreInputElementDescriptions(const std::wstring& name, const
    std::vector<D3D12_INPUT_ELEMENT_DESC>& inputElementLayout);
121
122          /*@brief Stores an array of input element descriptions with the specified name.
123 */
124          void StoreInputElementDescriptions(const std::wstring& name, const D3D12_INPUT_ELEMENT_DESC*
    inputElementLayout,
125              UINT numElements);
126
127          /*@brief Removes the specified input element description.
128 * If the specified input element description does not exist an out_of_range exception is thrown.
129 */
130          void RemoveInputElementDescription(const std::wstring& name);
131
132          /*@brief Creates a rasterization state description and stores it with the specified name.
133 */
134          void CreateRasterizationState(D3D12_FILL_MODE fillMode, BOOL enableMultisample, const
    std::wstring& name);
135
136          /*@brief Removes the specified rasterization state.
137 * If the specified rasterization state does not exist an out_of_range exception is thrown.
138 */
139          void RemoveRasterizationState(const std::wstring& name);
140
141          /*@brief Creates a PSO and stores it in the specified DrawSettings structure.
142 * If the specifed DrawSettings structure, Rasterization State, Vertex Shader, Pixel Shader or Input
    Layout
143 * does not exist an out_of_range exception is thrown.
144 */
145          void CreatePSO(const std::wstring& drawSettingsName, const std::wstring& rStateName,
146              const std::wstring& vsName, const std::wstring& psName, const std::wstring& inputLayoutName,
147              const D3D12_PRIMITIVE_TOPOLOGY_TYPE& primitiveType, UINT sampleCount);
148
149          /*@brief Creates a root signature and stores it with the specified name.
150 * If the specifed DrawSettings structure does not exist an out_of_range exception is thrown.
151 */
152          void CreateRootSignature(const std::wstring& drawSettingsName,
153              const D3D12_ROOT_PARAMETER* rootParameters, UINT numParameters);
154
155          /*@brief Creates a vertex buffer with the specified name and stores all of given data in the
    vertex buffer.
156 * Also creates a view to the vertex buffer.\n
157 * Execute commands and the flush command queue after calling createVertexBuffer() and
    createIndexBuffer().
158 */
159          void CreateVertexBuffer(const void* data, UINT numBytes, UINT stride);
160
```

```
165         void CreateIndexBuffer(const void* data, UINT numBytes, DXGI_FORMAT format);
166
169         void CreateCBVHeap(UINT numDescriptors, UINT shaderRegister);
170
173         void CreateConstantBuffer(UINT numOfBytes);
174
177         void CreateConstantBufferView(UINT index, UINT numBytes);
178
182         void SetPSO(const std::wstring& drawSettingsName, const
    Microsoft::WRL::ComPtr<ID3D12PipelineState>& pso);
183
187         void SetRootSignature(const std::wstring& drawSettingsName, const
    Microsoft::WRL::ComPtr<ID3D12RootSignature>& rootSignature);
188
192         void SetPrimitive(const std::wstring& drawSettingsName, const D3D_PRIMITIVE_TOPOLOGY&
    primitive);
193
197         void AddDrawArgument(const std::wstring& drawSettingsName, const FAShapes::DrawArguments&
    drawArg);
198
202         void AddDrawArgument(const std::wstring& drawSettingsName,
203             unsigned int indexCount, unsigned int locationOfFirstIndex, int indexOfFirstVertex, int
    indexOfConstantData);
204
208         void RemoveDrawArgument(const std::wstring& drawSettingsName, unsigned int index);
209
212         void CreateDrawSettings(const std::wstring& drawSettingsName);
213
217         void RemoveDrawSettings(const std::wstring& drawSettingsName);
218
223         void CreateText(const std::wstring& textName, FAMath::Vector4D textLocation, const std::wstring&
    textString,
224             float textSize, const FAColor::Color textColor);
225
229         void RemoveText(const std::wstring& textName);
230
236         void ChangeTextLocation(const std::wstring& textName, FAMath::Vector4D textLocation);
237
241         void ChangeTextString(const std::wstring& textName, const std::wstring& textString);
242
246         void ChangeTextSize(const std::wstring& textName, float textSize);
247
251         void ChangeTextColor(const std::wstring& textName, const FAColor::Color textColor);
252
256         void BeforeDrawObjects();
257
269         void DrawObjects(const std::wstring& drawSettingsName);
270
275         void AfterDrawObjects(bool renderText);
276
280         void BeforeDrawText();
281
293         void RenderText(const std::wstring& textName);
294
297         void AfterDrawText();
298
302         void AfterDraw();
303
306         void ExecuteAndFlush();
307
308     private:
309         //The device resources object that all RenderScene objects share.
310         static DeviceResources mDeviceResources;
311
312         //Stores all of the shaders and input element descriptions for this scene.
313         std::unordered_map<std::wstring, Microsoft::WRL::ComPtr<ID3DBlob>> mShaders;
314         std::unordered_map < std::wstring, std::vector<D3D12_INPUT_ELEMENT_DESC>>
    mInputElementDescriptions;
315
316         //Stores all of the rasterization states.
317         std::unordered_map <std::wstring, D3D12_RASTERIZER_DESC> mRasterizationStates;
318
319         //Stores all of the possible draw settings that the scene uses.
320         std::unordered_map <std::wstring, DrawSettings> mSceneObjects;
321
322         //Each scene gets one CBV heap.
323         Microsoft::WRL::ComPtr<ID3D12DescriptorHeap> mCBVHeap;
324         UINT mCBVSize;
325         D3D12_DESCRIPTOR_RANGE mCBVHeapDescription{};
326         D3D12_ROOT_PARAMETER mCBVHeapRootParameter;
327
328         //Stores all of the constant buffers this scene uses.  We can't update a constant buffer until
    the GPU
329         //is done executing all the commands that reference it, so each frame needs its own constant
    buffer.
330         ConstantBuffer mConstantBuffer[numFrames];
331
```

```
332         //The vertex and index buffer for this scene
333         VertexBuffer mVertexBuffer;
334         IndexBuffer mIndexBuffer;
335 
336         //All of the text that is rendered with the scene.
337         //Stores all of the possible draw settings that the scene uses.
338         std::unordered_map <std::wstring, Text> mSceneText;
339 
340     };
341 }
```

6.15 FAText.h File Reference

File has class Text under namespace FARender.

```
#include <string>
#include "FADeviceResources.h"
#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

  The namespace has utility functions and structs, VertexBuffer, IndexBuffer, ConstantBuffer, DeviceResources, RenderScene and Text classes.

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 
7 #include <string>
8 #include "FADeviceResources.h"
9 #include "FAColor.h"
10 
11 namespace FARender
12 {
17     class Text
18     {
19     public:
20 
23         Text();
24 
30         Text(const DeviceResources& deviceResources,
31             const FAMath::Vector4D& textLocation, const std::wstring& textString, float textSize, const
      FAColor::Color& textColor);
32 
37         void Initialize(const DeviceResources& deviceResources,
```

```
38            const FAMath::Vector4D& textLocation, const std::wstring& textString, float textSize, const
      FAColor::Color& textColor);
39
42          const FAMath::Vector4D& GetTextLocation() const;
43
46          const std::wstring& GetTextString() const;
47
50          float GetTextSize() const;
51
54          const Microsoft::WRL::ComPtr<ID2D1SolidColorBrush>& GetBrush() const;
55
58          const Microsoft::WRL::ComPtr<IDWriteTextFormat>& GetFormat() const;
59
62          const FAColor::Color& GetTextColor() const;
63
66          void SetTextSize(const DeviceResources& deviceResources, float textSize);
67
70          void SetTextColor(const FAColor::Color& textColor);
71
74          void SetTextString(const std::wstring& textString);
75
78          void SetTextLocation(const FAMath::Vector4D& textLocation);
79
80      private:
81
82          FAMath::Vector4D mTextLocation;
83          std::wstring mText;
84          float mTextSize;
85          FAColor::Color mTextColor;
86
87          Microsoft::WRL::ComPtr<ID2D1SolidColorBrush> mDirect2DBrush;
88          Microsoft::WRL::ComPtr<IDWriteTextFormat> mDirectWriteFormat;
89      };
90  }
```

6.17 FATime.h File Reference

File that has namespace FATime. Withn the namespace is the class Time.

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

Classes

- class FATime::Time

6.17.1 Detailed Description

File that has namespace FATime. Withn the namespace is the class Time.

6.18 FATime.h

Go to the documentation of this file.

```
1 #pragma once
2
7 #include <Windows.h>
8
12 namespace FATime
13 {
14      class Time
15      {
16      public:
20          Time();
21
24          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  }
```

6.19 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.

6.19.1 Detailed Description

File that has namespace FAWindow. Withn the namespace is the class Window.

6.20 FAWindow.h

Go to the documentation of this file.

```
1 #pragma once
2
7 #include <Windows.h>
8 #include <string>
9 #include <stdexcept>
10
14 namespace FAWindow
15 {
19     class Window
20     {
21     public:
22         //Window();
23
27         Window(const HINSTANCE& hInstance, const std::wstring& windowClassName, const std::wstring&
      windowName,
28             WNDPROC winProcFunction, unsigned int width, unsigned int height, void* additionalData =
      nullptr);
29
33         Window(const HINSTANCE& hInstance, const WNDCLASSEX& windowClass, const std::wstring& windowName,
34             unsigned int width, unsigned int height, void* additionalData = nullptr);
35
38         HWND GetWindowHandle() const;
39
42         unsigned int GetWidth() const ;
43
46         unsigned int GetHeight() const;
47
50         void SetWidth(unsigned int width);
51
54         void SetHeight(unsigned int height);
55
56     private:
57         HWND mWindowHandle;
58
59         WNDCLASSEX mWindowClass;
60         std::wstring mWindowClassName;
61
62         unsigned int mWidth;
63         unsigned int mHeight;
64     };
65 }
```

Index

~ConstantBuffer
    FARender::ConstantBuffer, 23
~DeviceResources
    FARender::DeviceResources, 27

AddDrawArgument
    FARender::RenderScene, 38
AfterDraw
    FARender::RenderScene, 38
AfterDrawObjects
    FARender::RenderScene, 39
AfterDrawText
    FARender::RenderScene, 39
AfterTextDraw
    FARender::DeviceResources, 27

Backward
    FACamera::Camera, 11
BeforeDrawObjects
    FARender::RenderScene, 39
BeforeDrawText
    FARender::RenderScene, 39
BeforeTextDraw
    FARender::DeviceResources, 27

Camera
    FACamera::Camera, 11
ChangeTextColor
    FARender::RenderScene, 39
ChangeTextLocation
    FARender::RenderScene, 40
ChangeTextSize
    FARender::RenderScene, 40
ChangeTextString
    FARender::RenderScene, 40
Color
    FAColor::Color, 19
CopyData
    FARender::ConstantBuffer, 23
CreateCBVHeap
    FARender::RenderScene, 40
CreateConstantBuffer
    FARender::ConstantBuffer, 23
    FARender::RenderScene, 40
CreateConstantBufferView
    FARender::ConstantBuffer, 24
    FARender::RenderScene, 41
CreateDrawSettings
    FARender::RenderScene, 41
CreateIndexBuffer
    FARender::IndexBuffer, 35
    FARender::RenderScene, 41
CreateIndexBufferView
    FARender::IndexBuffer, 35
CreateText
    FARender::RenderScene, 41
CreateVertexBuffer
    FARender::VertexBuffer, 50
CreateVertexBufferView
    FARender::VertexBuffer, 50

DeltaTime
    FATime::Time, 48
DirectXException, 34
DisableMSAA
    FARender::DeviceResources, 27
Down
    FACamera::Camera, 12
DrawObjects
    FARender::RenderScene, 41

EnableMSAA
    FARender::DeviceResources, 27
Execute
    FARender::DeviceResources, 27
ExecuteAndFlush
    FARender::RenderScene, 42

FABuffer.h, 55
FACamera, 7
FACamera.h, 57
    vec2, 57
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, 59
operator∗, 60
operator+, 60
operator-, 61
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, 62
FARender, 7
nextFrame, 8
FARender::ConstantBuffer, 22
∼ConstantBuffer, 23
CopyData, 23
CreateConstantBuffer, 23
CreateConstantBufferView, 24
GetConstantBuffer, 24
FARender::DeviceResources, 24
∼DeviceResources, 27
AfterTextDraw, 27
BeforeTextDraw, 27
DisableMSAA, 27
EnableMSAA, 27
Execute, 27
FlushCommandQueue, 28
GetBackBufferFormat, 28
GetCommandAllocator, 28
GetCommandList, 28
GetCommandQueue, 28
GetCurrentBackBuffer, 28
GetCurrentFenceValue, 29
GetDepthStencilBuffer, 29
GetDepthStencilFormat, 29
GetDevice, 29
GetDevice2DContext, 29
GetDirectWriteFactory, 29
GetDSVDescriptorHeap, 30
GetDSVDescriptorSize, 30
GetNumOfSwapChainBuffers, 30
GetRTVDescriptorHeap, 30
GetRTVDescriptorSize, 30
GetSampleCount, 30, 31
GetScissor, 31
GetSwapChain, 31
GetSwapChainBuffers, 31
GetViewport, 31
InitializeDirect3D, 31
IsMSAAEnabled, 32
Present, 32
ResetCommandAllocator, 32
ResetCommandList, 32
Resize, 32
RTBufferTransition, 33
Signal, 33
UpdateCurrentFrameFenceValue, 33
WaitForGPU, 33
FARender::DrawSettings, 34
FARender::IndexBuffer, 34
CreateIndexBuffer, 35
CreateIndexBufferView, 35
GetIndexBufferView, 35
FARender::RenderScene, 36
AddDrawArgument, 38
AfterDraw, 38
AfterDrawObjects, 39
AfterDrawText, 39
BeforeDrawObjects, 39
BeforeDrawText, 39
ChangeTextColor, 39
ChangeTextLocation, 40
ChangeTextSize, 40
ChangeTextString, 40
CreateCBVHeap, 40
CreateConstantBuffer, 40
CreateConstantBufferView, 41
CreateDrawSettings, 41
CreateIndexBuffer, 41
CreateText, 41
DrawObjects, 41
ExecuteAndFlush, 42
GetCBVHeap, 42
GetCBVHeapRootParameter, 42
GetCBVSize, 42
RemoveDrawArgument, 42

RemoveDrawSettings, 43
RemoveText, 43
RenderText, 43
SetPrimitive, 43
SetPSO, 43
SetRootSignature, 44
FARender::Text, 44
GetBrush, 45
GetFormat, 46
GetTextColor, 46
GetTextLocation, 46
GetTextSize, 46
GetTextString, 46
Initialize, 46
SetTextColor, 47
SetTextLocation, 47
SetTextSize, 47
SetTextString, 47
Text, 45
FARender::VertexBuffer, 50
CreateVertexBuffer, 50
CreateVertexBufferView, 50
GetVertexBufferView, 50
FARenderingUtility.h, 65
FARenderScene.h, 66
FAText.h, 70
FATime.h, 71
FATime::Time, 48
DeltaTime, 48
Reset, 48
Start, 48
Stop, 49
Tick, 49
Time, 48
TotalTime, 49
FAWindow, 8
FAWindow.h, 72
FAWindow::Window, 51
GetHeight, 52
GetWidth, 52
GetWindowHandle, 52
SetHeight, 53
SetWidth, 53
Window, 51, 52
FlushCommandQueue
FARender::DeviceResources, 28
Foward
FACamera::Camera, 12

GetAlpha
FAColor::Color, 20
GetAngularVelocity
FACamera::Camera, 12
GetAspectRatio
FACamera::Camera, 12
GetBackBufferFormat
FARender::DeviceResources, 28
GetBlue
FAColor::Color, 20
GetBrush
FARender::Text, 45
GetCameraPosition
FACamera::Camera, 12
GetCameraVelocity
FACamera::Camera, 12
GetCBVHeap
FARender::RenderScene, 42
GetCBVHeapRootParameter
FARender::RenderScene, 42
GetCBVSize
FARender::RenderScene, 42
GetColor
FAColor::Color, 20
GetCommandAllocator
FARender::DeviceResources, 28
GetCommandList
FARender::DeviceResources, 28
GetCommandQueue
FARender::DeviceResources, 28
GetConstantBuffer
FARender::ConstantBuffer, 24
GetCurrentBackBuffer
FARender::DeviceResources, 28
GetCurrentFenceValue
FARender::DeviceResources, 29
GetDepthStencilBuffer
FARender::DeviceResources, 29
GetDepthStencilFormat
FARender::DeviceResources, 29
GetDevice
FARender::DeviceResources, 29
GetDevice2DContext
FARender::DeviceResources, 29
GetDirectWriteFactory
FARender::DeviceResources, 29
GetDSVDescriptorHeap
FARender::DeviceResources, 30
GetDSVDescriptorSize
FARender::DeviceResources, 30
GetFormat
FARender::Text, 46
GetGreen
FAColor::Color, 20
GetHeight
FAWindow::Window, 52
GetIndexBufferView
FARender::IndexBuffer, 35
GetNumOfSwapChainBuffers
FARender::DeviceResources, 30
GetPerspectiveProjectionMatrix
FACamera::Camera, 13
GetRed
FAColor::Color, 20
GetRTVDescriptorHeap
FARender::DeviceResources, 30
GetRTVDescriptorSize
FARender::DeviceResources, 30

GetSampleCount
FARender::DeviceResources, 30, 31
GetScissor
FARender::DeviceResources, 31
GetSwapChain
FARender::DeviceResources, 31
GetSwapChainBuffers
FARender::DeviceResources, 31
GetTextColor
FARender::Text, 46
GetTextLocation
FARender::Text, 46
GetTextSize
FARender::Text, 46
GetTextString
FARender::Text, 46
GetVertexBufferView
FARender::VertexBuffer, 50
GetVerticalFov
FACamera::Camera, 13
GetViewPerspectiveProjectionMatrix
FACamera::Camera, 13
GetViewport
FARender::DeviceResources, 31
GetViewTransformationMatrix
FACamera::Camera, 13
GetWidth
FAWindow::Window, 52
GetWindowHandle
FAWindow::Window, 52
GetX
FACamera::Camera, 13
GetY
FACamera::Camera, 13
GetZ
FACamera::Camera, 14
GetZFar
FACamera::Camera, 14
GetZNear
FACamera::Camera, 14

Initialize
FARender::Text, 46
InitializeDirect3D
FARender::DeviceResources, 31
IsMSAAEnabled
FARender::DeviceResources, 32

KeyboardInput
FACamera::Camera, 14

Left
FACamera::Camera, 14
LookAt
FACamera::Camera, 14

MouseInput
FACamera::Camera, 15

nextFrame
FARender, 8

operator∗
FAColor.h, 60
operator∗=
FAColor::Color, 20, 21
operator+
FAColor.h, 60
operator+=
FAColor::Color, 21
operator-
FAColor.h, 61
operator-=
FAColor::Color, 21

Present
FARender::DeviceResources, 32

RemoveDrawArgument
FARender::RenderScene, 42
RemoveDrawSettings
FARender::RenderScene, 43
RemoveText
FARender::RenderScene, 43
RenderText
FARender::RenderScene, 43
Reset
FATime::Time, 48
ResetCommandAllocator
FARender::DeviceResources, 32
ResetCommandList
FARender::DeviceResources, 32
Resize
FARender::DeviceResources, 32
Right
FACamera::Camera, 15
RotateCameraLeftRight
FACamera::Camera, 15
RotateCameraUpDown
FACamera::Camera, 15
RTBufferTransition
FARender::DeviceResources, 33

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, 53
SetPrimitive
FARender::RenderScene, 43
SetPSO
FARender::RenderScene, 43
SetRed
FAColor::Color, 22
SetRootSignature
FARender::RenderScene, 44
SetTextColor
FARender::Text, 47
SetTextLocation
FARender::Text, 47
SetTextSize
FARender::Text, 47
SetTextString
FARender::Text, 47
SetVerticalFov
FACamera::Camera, 16
SetWidth
FAWindow::Window, 53
SetX
FACamera::Camera, 16
SetY
FACamera::Camera, 16
SetZ
FACamera::Camera, 17
SetZFar
FACamera::Camera, 17
SetZNear
FACamera::Camera, 17
Signal
FARender::DeviceResources, 33
Start
FATime::Time, 48
Stop
FATime::Time, 49

Text
FARender::Text, 45
Tick
FATime::Time, 49
Time, 49
FATime::Time, 48
TotalTime
FATime::Time, 49

Up
FACamera::Camera, 17
UpdateCurrentFrameFenceValue
FARender::DeviceResources, 33
UpdatePerspectiveProjectionMatrix
FACamera::Camera, 17
UpdateViewMatrix
FACamera::Camera, 17
UpdateViewPerspectiveProjectionMatrix
FACamera::Camera, 18

vec2
FACamera.h, 57

WaitForGPU
FARender::DeviceResources, 33
Window
FAWindow::Window, 51, 52