

Blade Engine

Generated by Doxygen 1.8.13

Contents

1	Namespace Index	1
1.1	Namespace List	1
2	Hierarchical Index	3
2.1	Class Hierarchy	3
3	Class Index	7
3.1	Class List	7
4	Namespace Documentation	11
4.1	Blade Namespace Reference	11
4.1.1	Detailed Description	16
5	Class Documentation	17
5.1	Blade::AbstractViewport Class Reference	17
5.1.1	Detailed Description	17
5.1.2	Constructor & Destructor Documentation	18
5.1.2.1	AbstractViewport()	18
5.1.3	Member Function Documentation	18
5.1.3.1	GetRect()	18
5.1.3.2	SetRect()	18
5.2	Blade::Animation Class Reference	19
5.3	Blade::Application Class Reference	19
5.4	Blade::AudioManager Class Reference	20
5.5	Blade::AudioSample Class Reference	21

5.6	Blade::AudioSource Class Reference	21
5.7	Blade::AudioStream Class Reference	22
5.8	Blade::AudioStreamBuffer Struct Reference	22
5.9	Blade::BehaviourComponent Class Reference	23
5.9.1	Constructor & Destructor Documentation	23
5.9.1.1	BehaviourComponent()	23
5.9.2	Member Function Documentation	24
5.9.2.1	Update()	24
5.10	Blade::BehaviourSystem Class Reference	24
5.10.1	Detailed Description	25
5.10.2	Member Function Documentation	25
5.10.2.1	Initialize()	25
5.10.2.2	Process()	25
5.10.2.3	RegisterComponent()	25
5.10.2.4	UnregisterComponent()	26
5.11	Blade::BoundingSphere Class Reference	26
5.12	Blade::Camera Class Reference	27
5.13	Blade::CameraComponent Class Reference	27
5.13.1	Detailed Description	28
5.13.2	Constructor & Destructor Documentation	28
5.13.2.1	CameraComponent() [1/3]	28
5.13.2.2	CameraComponent() [2/3]	29
5.13.2.3	CameraComponent() [3/3]	29
5.13.2.4	~CameraComponent()	30
5.13.3	Member Function Documentation	30
5.13.3.1	GetClippingPlanes()	30
5.13.3.2	GetFarPlane()	30
5.13.3.3	GetFov()	30
5.13.3.4	GetNearPlane()	31
5.13.3.5	GetProjectionMatrix()	31

5.13.3.6	GetViewMatrix()	31
5.13.3.7	GetViewport()	31
5.13.3.8	SetClippingPlanes() [1/2]	31
5.13.3.9	SetClippingPlanes() [2/2]	32
5.13.3.10	SetFarPlane()	32
5.13.3.11	SetFov()	32
5.13.3.12	SetNearPlane()	33
5.13.3.13	SetViewMatrix()	33
5.13.3.14	SetViewport()	33
5.14	Blade::CameraDesc Struct Reference	33
5.15	Blade::CameraSystem Class Reference	34
5.15.1	Detailed Description	34
5.15.2	Member Function Documentation	35
5.15.2.1	GetActiveCamera()	35
5.15.2.2	GetActiveCameraProjectionMatrtix()	35
5.15.2.3	GetActiveCameraViewMatrix()	35
5.15.2.4	GetActiveCameraViewport()	35
5.15.2.5	GetCamera()	35
5.15.2.6	RegisterComponent()	36
5.15.2.7	SetActiveCamera()	36
5.15.2.8	UnregisterComponent()	36
5.16	Blade::Collider Class Reference	37
5.17	Blade::ColliderComponent Class Reference	37
5.18	Blade::Command Class Reference	38
5.19	Blade::Component Class Reference	38
5.19.1	Detailed Description	39
5.19.2	Constructor & Destructor Documentation	39
5.19.2.1	Component()	39
5.19.3	Member Function Documentation	39
5.19.3.1	GetId()	40

5.19.3.2	GetParent()	40
5.19.3.3	GetType()	40
5.19.3.4	OnMessage()	40
5.20	Blade::ConfigEntry Class Reference	41
5.21	Blade::ConfigFile Class Reference	41
5.22	Blade::ConnectionInfo Struct Reference	42
5.23	Blade::ContactManifold Class Reference	42
5.24	Blade::D3D11BlendState Class Reference	42
5.25	Blade::D3D11Context Class Reference	43
5.26	Blade::D3D11DepthStencilState Class Reference	43
5.27	Blade::D3D11IBO Class Reference	44
5.28	Blade::D3D11RasterizerState Class Reference	44
5.29	Blade::D3D11RenderTarget Class Reference	45
5.30	Blade::D3D11Shader Class Reference	45
5.31	Blade::D3D11ShaderProgram Class Reference	46
5.32	Blade::D3D11Texture Class Reference	46
5.33	Blade::D3D11VBO Class Reference	47
5.34	Blade::D3D11Viewport Class Reference	47
5.34.1	Detailed Description	47
5.34.2	Constructor & Destructor Documentation	48
5.34.2.1	D3D11Viewport()	48
5.35	Blade::D3D11Window Class Reference	48
5.36	Blade::MathUtils::Derivative Struct Reference	49
5.37	Blade::DirectionalLight Class Reference	49
5.38	Blade::DirectionalLightComponent Class Reference	49
5.39	Blade::DirectionalLightDesc Struct Reference	50
5.39.1	Detailed Description	50
5.40	Blade::EmitterComponent Class Reference	50
5.40.1	Member Function Documentation	51
5.40.1.1	Update()	51

5.41 Blade::EmitterDescriptor Struct Reference	52
5.42 Blade::EngineContext Class Reference	53
5.43 Blade::Entity Class Reference	53
5.44 Blade::GAPIContext Class Reference	54
5.45 Blade::IBO Class Reference	55
5.46 Blade::InputComponent Class Reference	55
5.46.1 Member Function Documentation	56
5.46.1.1 Update()	56
5.47 Blade::InputDevice Class Reference	56
5.48 Blade::InputManager Class Reference	58
5.48.1 Member Function Documentation	58
5.48.1.1 ActiveDeviceExists()	58
5.48.1.2 AssignDeviceToPlayer()	59
5.48.1.3 DevicePoolQueryType()	59
5.48.1.4 EnumerateDevices()	60
5.48.1.5 GetActiveDevice()	60
5.48.1.6 GetAnalogStickVector()	60
5.48.1.7 Initialize()	61
5.48.1.8 PooledDeviceExists()	61
5.48.1.9 QueryAllKeyStates()	61
5.48.1.10 QueryDeviceAllStates()	62
5.48.1.11 QueryDeviceState()	62
5.48.1.12 QueryKeyState()	62
5.48.1.13 QueryMouseButtonState()	63
5.48.1.14 QueryMouseMovement()	63
5.48.1.15 QueryMouseMovementNormalized()	63
5.48.1.16 QueryMousePosition()	64
5.48.1.17 UnassignDevice()	64
5.48.1.18 Update()	64
5.49 Blade::InputState Struct Reference	64

5.49.1 Detailed Description	65
5.50 Blade::JoypadInputComponent Class Reference	65
5.50.1 Member Function Documentation	66
5.50.1.1 Update()	66
5.51 Blade::KeyboardInput Class Reference	67
5.51.1 Detailed Description	67
5.51.2 Member Function Documentation	67
5.51.2.1 QueryAllKeyStates()	67
5.51.2.2 QueryKeyState()	67
5.52 Blade::KeyboardInputComponent Class Reference	68
5.52.1 Member Function Documentation	69
5.52.1.1 Update()	69
5.53 Blade::Keyframe< T > Struct Template Reference	69
5.54 Blade::LightComponent Class Reference	70
5.54.1 Detailed Description	70
5.55 Blade::LightSystem Class Reference	70
5.55.1 Detailed Description	71
5.55.2 Member Function Documentation	71
5.55.2.1 GetDirectionalLightDescriptions()	71
5.55.2.2 GetPointLightDescriptions()	71
5.55.2.3 GetSpotlightDescriptions()	72
5.55.2.4 Initialize()	72
5.55.2.5 Process()	72
5.55.2.6 RegisterComponent()	72
5.55.2.7 UnregisterComponent()	73
5.56 Blade::ManifoldEntry Struct Reference	73
5.57 Blade::Material Struct Reference	73
5.58 Blade::Mesh Class Reference	74
5.59 Blade::Message< T > Class Template Reference	75
5.60 Blade::NCF Class Reference	75

5.61 Blade::NetworkManager Class Reference	75
5.62 Blade::NetworkMessage Class Reference	76
5.63 Blade::Observer Class Reference	76
5.64 Blade::ObserverSubject Class Reference	76
5.65 Blade::OggVorbisStream Class Reference	77
5.66 Blade::Particle Struct Reference	77
5.67 Blade::ParticleSystem Class Reference	78
5.67.1 Member Function Documentation	78
5.67.1.1 Initialize()	78
5.67.1.2 Process()	78
5.68 Blade::Pipeline< T, Tdata > Class Template Reference	79
5.68.1 Detailed Description	79
5.68.2 Member Function Documentation	79
5.68.2.1 AddStage()	79
5.68.2.2 Execute()	80
5.69 Blade::PipelineData< T > Class Template Reference	80
5.69.1 Detailed Description	80
5.69.2 Constructor & Destructor Documentation	81
5.69.2.1 PipelineData()	81
5.69.3 Member Function Documentation	81
5.69.3.1 Get()	81
5.70 Blade::PipelineStage< T, Tdata > Class Template Reference	81
5.70.1 Detailed Description	82
5.70.2 Constructor & Destructor Documentation	82
5.70.2.1 PipelineStage()	82
5.70.3 Member Function Documentation	82
5.70.3.1 Execute()	83
5.70.3.2 Initialize()	83
5.71 Blade::PlaneCollider Class Reference	83
5.71.1 Detailed Description	84

5.72 Blade::PointLight Class Reference	84
5.73 Blade::PointLightComponent Class Reference	85
5.74 Blade::PointLightDesc Struct Reference	85
5.74.1 Detailed Description	85
5.75 Blade::RefCountedContainer< T > Class Template Reference	86
5.76 Blade::RenderComponent Class Reference	86
5.76.1 Detailed Description	87
5.76.2 Constructor & Destructor Documentation	87
5.76.2.1 RenderComponent()	87
5.76.3 Member Function Documentation	87
5.76.3.1 GetMaterial()	87
5.76.3.2 GetMesh()	87
5.76.3.3 SetMaterial()	87
5.76.3.4 SetMesh()	88
5.77 Blade::RenderState Class Reference	88
5.78 Blade::RenderStateManager Class Reference	89
5.79 Blade::RenderSystem Class Reference	89
5.79.1 Detailed Description	89
5.79.2 Member Function Documentation	90
5.79.2.1 Initialize()	90
5.79.2.2 Process()	90
5.79.2.3 RegisterComponent()	90
5.79.2.4 SetRenderPassPipeline()	91
5.79.2.5 UnregisterComponent()	91
5.80 Blade::RenderTarget Class Reference	91
5.81 Blade::Resource Class Reference	92
5.82 Blade::ResourceManager Class Reference	92
5.83 Blade::MathUtils::RungeKutta4Integrator Class Reference	92
5.84 Blade::SamplePlaylist Struct Reference	93
5.85 Blade::Scene Class Reference	93

5.86 Blade::SceneManager Class Reference	94
5.87 Blade::Serializable Class Reference	94
5.88 Blade::Shader Class Reference	94
5.89 Blade::ShaderProgram Class Reference	95
5.90 Blade::ShaderProgramDesc Struct Reference	95
5.91 Blade::ShaderProgramManager Class Reference	96
5.92 Blade::SimulationComponent Class Reference	96
5.93 Blade::SimulationComponentState Struct Reference	97
5.94 Blade::SimulationSystem Class Reference	97
5.94.1 Detailed Description	98
5.94.2 Member Function Documentation	98
5.94.2.1 Initialize()	98
5.94.2.2 Process()	98
5.95 Blade::Socket Class Reference	99
5.96 Blade::Spotlight Class Reference	99
5.97 Blade::SpotlightComponent Class Reference	100
5.98 Blade::SpotlightDesc Struct Reference	100
5.98.1 Detailed Description	101
5.99 Blade::MathUtils::State Struct Reference	101
5.100 Blade::StreamPlaylist Struct Reference	101
5.101 Blade::System Class Reference	101
5.101.1 Detailed Description	102
5.101.2 Member Function Documentation	102
5.101.2.1 Initialize()	102
5.101.2.2 Process()	102
5.102 Blade::Texture Class Reference	103
5.103 Blade::ThreadPool Class Reference	103
5.103.1 Member Function Documentation	103
5.103.1.1 Initialize()	104
5.104 Blade::ThumbStick Struct Reference	104
5.104.1 Detailed Description	104
5.105 Blade::Timer Class Reference	104
5.106 Blade::VBO Class Reference	105
5.107 Blade::Vertex Struct Reference	105
5.108 Blade::Win32Window Class Reference	106
5.109 Blade::Window Class Reference	106
5.110 Blade::WindowFunctionCallbacks Struct Reference	107
5.111 Blade::WindowingService Class Reference	108
5.112 Blade::XInputDevice Class Reference	108

Chapter 1

Namespace Index

1.1 Namespace List

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

Blade	Key values for keyboard input	11
-----------------------	---	--------------------

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Blade::AbstractViewport	17
Blade::D3D11Viewport	47
Blade::Animation	19
Blade::Application	19
Blade::AudioManager	20
Blade::AudioSource	21
Blade::AudioStream	22
Blade::OggVorbisStream	77
Blade::AudioStreamBuffer	22
Blade::CameraDesc	33
Blade::Collider	37
Blade::BoundingSphere	26
Blade::PlaneCollider	83
Blade::Command	38
Blade::ConfigEntry	41
Blade::ConfigFile	41
Blade::ConnectionInfo	42
Blade::ContactManifold	42
Blade::MathUtils::Derivative	49
Blade::DirectionalLightDesc	50
Blade::EngineContext	53
Blade::GAPIContext	54
Blade::D3D11Context	43
Blade::IBO	55
Blade::D3D11IBO	44
Blade::InputDevice	56
Blade::XInputDevice	108
Blade::InputManager	58
Blade::InputState	64
Blade::KeyboardInput	67
Blade::Keyframe< T >	69
Blade::ManifoldEntry	73
Blade::Material	73

Blade::Message< T >	75
Blade::Message< int >	75
Blade::NetworkMessage	76
Blade::NCF	75
Blade::NetworkManager	75
Blade::Observer	76
Blade::Component	38
Blade::BehaviourComponent	23
Blade::EmitterComponent	50
Blade::InputComponent	55
Blade::JoypadInputComponent	65
Blade::KeyboardInputComponent	68
Blade::CameraComponent	27
Blade::ColliderComponent	37
Blade::LightComponent	70
Blade::DirectionalLightComponent	49
Blade::PointLightComponent	85
Blade::SpotlightComponent	100
Blade::RenderComponent	86
Blade::SimulationComponent	96
Blade::ObserverSubject	76
Blade::Entity	53
Blade::Camera	27
Blade::DirectionalLight	49
Blade::PointLight	84
Blade::Spotlight	99
Blade::Particle	77
Blade::Pipeline< T, Tdata >	79
Blade::PipelineData< T >	80
Blade::PipelineStage< T, Tdata >	81
Blade::PointLightDesc	85
Blade::RefCountedContainer< T >	86
Blade::RenderState	88
Blade::D3D11BlendState	42
Blade::D3D11DepthStencilState	43
Blade::D3D11RasterizerState	44
Blade::RenderStateManager	89
Blade::RenderTarget	91
Blade::D3D11RenderTarget	45
Blade::Resource	92
Blade::AudioSample	21
Blade::EmitterDescriptor	52
Blade::Entity	53
Blade::Mesh	74
Blade::Shader	94
Blade::D3D11Shader	45
Blade::Texture	103
Blade::D3D11Texture	46
Blade::ResourceManager	92
Blade::MathUtils::RungeKutta4Integrator	92
Blade::SamplePlaylist	93
Blade::Scene	93
Blade::SceneManager	94
Blade::Serializable	94
Blade::NetworkMessage	76
Blade::ShaderProgram	95

Blade::D3D11ShaderProgram	46
Blade::ShaderProgramDesc	95
Blade::ShaderProgramManager	96
Blade::SimulationComponentState	97
Blade::Socket	99
Blade::SpotlightDesc	100
Blade::MathUtils::State	101
Blade::StreamPlaylist	101
Blade::System	101
Blade::BehaviourSystem	24
Blade::CameraSystem	34
Blade::LightSystem	70
Blade::ParticleSystem	78
Blade::RenderSystem	89
Blade::SimulationSystem	97
Blade::ThreadPool	103
Blade::ThumbStick	104
Blade::Timer	104
Blade::VBO	105
Blade::D3D11VBO	47
Blade::Vertex	105
Blade::Window	106
Blade::Win32Window	106
Blade::D3D11Window	48
Blade::WindowFunctionCallbacks	107
Blade::WindowingService	108

Chapter 3

Class Index

3.1 Class List

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

Blade::AbstractViewport	
Describes an implementation agnostic Viewport	17
Blade::Animation	19
Blade::Application	19
Blade::AudioManager	20
Blade::AudioSample	21
Blade::AudioSource	21
Blade::AudioStream	22
Blade::AudioStreamBuffer	22
Blade::BehaviourComponent	23
Blade::BehaviourSystem	
A System responsible to process and manage the BehaviourComponents by calling the Update method on every component	24
Blade::BoundingSphere	26
Blade::Camera	27
Blade::CameraComponent	
Represents a CameraComponent . This component contains all the information needed for the view and projection transformations. Managed by the CameraSystem	27
Blade::CameraDesc	33
Blade::CameraSystem	
A System responsible to process and manage the CameraComponents by swapping the current active camera and providing the current active camera's matrices	34
Blade::Collider	37
Blade::ColliderComponent	37
Blade::Command	38
Blade::Component	
Base Component class of the engine. All the components of the engine derive from this class. Component inherits from the Observer class so it can register and receive specific messages .	38
Blade::ConfigEntry	41
Blade::ConfigFile	41
Blade::ConnectionInfo	42
Blade::ContactManifold	42
Blade::D3D11BlendState	42
Blade::D3D11Context	43
Blade::D3D11DepthStencilState	43

Blade::D3D11IBO	44
Blade::D3D11RasterizerState	44
Blade::D3D11RenderTarget	45
Blade::D3D11Shader	45
Blade::D3D11ShaderProgram	46
Blade::D3D11Texture	46
Blade::D3D11VBO	47
Blade::D3D11Viewport	
D3D11 implementation of the AbstractViewport	47
Blade::D3D11Window	48
Blade::MathUtils::Derivative	49
Blade::DirectionalLight	49
Blade::DirectionalLightComponent	49
Blade::DirectionalLightDesc	
A struct describing a directional light	50
Blade::EmitterComponent	50
Blade::EmitterDescriptor	52
Blade::EngineContext	53
Blade::Entity	53
Blade::GAPIContext	54
Blade::IBO	55
Blade::InputComponent	55
Blade::InputDevice	56
Blade::InputManager	58
Blade::InputState	
InputState describes the current state of a device	64
Blade::JoypadInputComponent	65
Blade::KeyboardInput	
Keyboard abstraction of the engine	67
Blade::KeyboardInputComponent	68
Blade::Keyframe< T >	69
Blade::LightComponent	
Abstract class that describes a LightComponent . Provides the base functionality of a LightComponent . It contains the component's type and an index to the entry of the correct light description cache in the LightSystem . Managed by the LightSystem	70
Blade::LightSystem	
A System responsible for managing LightComponents. This system updates the positions of all the lights in the scene every frame. It is also responsible for caching the light descriptions of each light upon registration of a LightComponent	70
Blade::ManifoldEntry	73
Blade::Material	73
Blade::Mesh	74
Blade::Message< T >	75
Blade::NCF	75
Blade::NetworkManager	75
Blade::NetworkMessage	76
Blade::Observer	76
Blade::ObserverSubject	76
Blade::OggVorbisStream	77
Blade::Particle	77
Blade::ParticleSystem	78
Blade::Pipeline< T, Tdata >	
Abstract class that describes a pipeline that processes the specified object data type	79
Blade::PipelineData< T >	
An abstract data container for the data returned by a PipelineStage	80
Blade::PipelineStage< T, Tdata >	
This class describes an abstract stage of a pipeline that processes the specified type of data and returns the specified type of data	81

Blade::PlaneCollider	
Bounding Plane class is a collider	83
Blade::PointLight	84
Blade::PointLightComponent	85
Blade::PointLightDesc	
A struct describing a point light	85
Blade::RefCountedContainer< T >	86
Blade::RenderComponent	
Represents a RenderComponent . The RenderComponent makes an entity renderable. This component is processed by the RenderSystem	86
Blade::RenderState	88
Blade::RenderStateManager	89
Blade::RenderSystem	
A System responsible for processing the RenderComponents by passing them through a specified pipeline	89
Blade::RenderTarget	91
Blade::Resource	92
Blade::ResourceManager	92
Blade::MathUtils::RungeKutta4Integrator	92
Blade::SamplePlaylist	93
Blade::Scene	93
Blade::SceneManager	94
Blade::Serializable	94
Blade::Shader	94
Blade::ShaderProgram	95
Blade::ShaderProgramDesc	95
Blade::ShaderProgramManager	96
Blade::SimulationComponent	96
Blade::SimulationComponentState	97
Blade::SimulationSystem	
The simulation system of the engine	97
Blade::Socket	99
Blade::Spotlight	99
Blade::SpotlightComponent	100
Blade::SpotlightDesc	
A struct describing a spotlight	100
Blade::MathUtils::State	101
Blade::StreamPlaylist	101
Blade::System	
An interface that represents a system of the engine	101
Blade::Texture	103
Blade::ThreadPool	103
Blade::ThumbStick	
Thumbstick structure to hold X/Y axis information	104
Blade::Timer	104
Blade::VBO	105
Blade::Vertex	105
Blade::Win32Window	106
Blade::Window	106
Blade::WindowFunctionCallbacks	107
Blade::WindowingService	108
Blade::XInputDevice	108

Chapter 4

Namespace Documentation

4.1 Blade Namespace Reference

Key values for keyboard input.

Classes

- class [AbstractViewport](#)

Describes an implementation agnostic Viewport.

- class [Animation](#)
- class [Application](#)
- class [AudioManager](#)
- class [AudioSample](#)
- class [AudioSource](#)
- class [AudioStream](#)
- struct [AudioStreamBuffer](#)
- class [BehaviourComponent](#)
- class [BehaviourSystem](#)

A [System](#) responsible to process and manage the BehaviourComponents by calling the Update method on every component.

- class [BoundingSphere](#)
- class [Camera](#)
- class [CameraComponent](#)

Represents a [CameraComponent](#). This component contains all the information needed for the view and projection transformations. Managed by the [CameraSystem](#).

- struct [CameraDesc](#)
- class [CameraSystem](#)

A [System](#) responsible to process and manage the CameraComponents by swapping the current active camera and providing the current active camera's matrices.

- class [Collider](#)
- class [ColliderComponent](#)
- class [Command](#)
- class [Component](#)

Base [Component](#) class of the engine. All the components of the engine derive from this class. Component inherits from the [Observer](#) class so it can register and receive specific messages.

- class [ConfigEntry](#)
- class [ConfigFile](#)

- struct [ConnectionInfo](#)
- class [ContactManifold](#)
- class [D3D11BlendState](#)
- class [D3D11Context](#)
- class [D3D11DepthStencilState](#)
- class [D3D11IBO](#)
- class [D3D11RasterizerState](#)
- class [D3D11RenderTarget](#)
- class [D3D11Shader](#)
- class [D3D11ShaderProgram](#)
- class [D3D11Texture](#)
- class [D3D11VBO](#)
- class [D3D11Viewport](#)

D3D11 implementation of the [AbstractViewport](#).

- class [D3D11Window](#)
- class [DirectionalLight](#)
- class [DirectionalLightComponent](#)
- struct [DirectionalLightDesc](#)

A struct describing a directional light.

- class [EmitterComponent](#)
- struct [EmitterDescriptor](#)
- class [EngineContext](#)
- class [Entity](#)
- class [GAPContext](#)
- class [IBO](#)
- class [InputComponent](#)
- class [InputDevice](#)
- class [InputManager](#)
- struct [InputState](#)

[InputState](#) describes the current state of a device.

- class [JoypadInputComponent](#)
- class [KeyboardInput](#)

Keyboard abstraction of the engine.

- class [KeyboardInputComponent](#)
- struct [Keyframe](#)
- class [LightComponent](#)

Abstract class that describes a [LightComponent](#). Provides the base functionality of a [LightComponent](#). It contains the component's type and an index to the entry of the correct light description cache in the [LightSystem](#). Managed by the [LightSystem](#).

- class [LightSystem](#)

A [System](#) responsible for managing [LightComponents](#). This system updates the positions of all the lights in the scene every frame. It is also responsible for caching the light descriptions of each light upon registration of a [LightComponent](#).

- struct [ManifoldEntry](#)
- struct [Material](#)
- class [Mesh](#)
- class [Message](#)
- class [NCF](#)
- class [NetworkManager](#)
- class [NetworkMessage](#)
- class [Observer](#)
- class [ObserverSubject](#)
- class [OggVorbisStream](#)
- struct [Particle](#)
- class [ParticleSystem](#)

- class [Pipeline](#)
Abstract class that describes a pipeline that processes the specified object data type.
- class [PipelineData](#)
An abstract data container for the data returned by a [PipelineStage](#).
- class [PipelineStage](#)
This class describes an abstract stage of a pipeline that processes the specified type of data and returns the specified type of data.
- class [PlaneCollider](#)
Bounding Plane class is a collider.
- class [PointLight](#)
- class [PointLightComponent](#)
- struct [PointLightDesc](#)
A struct describing a point light.
- class [RefCountedContainer](#)
- class [RenderComponent](#)
Represents a [RenderComponent](#). The [RenderComponent](#) makes an entity renderable. This component is processed by the [RenderSystem](#).
- class [RenderState](#)
- class [RenderStateManager](#)
- class [RenderSystem](#)
A [System](#) responsible for processing the [RenderComponents](#) by passing them through a specified pipeline.
- class [RenderTarget](#)
- class [Resource](#)
- class [ResourceManager](#)
- struct [SamplePlaylist](#)
- class [Scene](#)
- class [SceneManager](#)
- class [Serializable](#)
- class [Shader](#)
- class [ShaderProgram](#)
- struct [ShaderProgramDesc](#)
- class [ShaderProgramManager](#)
- class [SimulationComponent](#)
- struct [SimulationComponentState](#)
- class [SimulationSystem](#)
The simulation system of the engine.
- class [Socket](#)
- class [Spotlight](#)
- class [SpotlightComponent](#)
- struct [SpotlightDesc](#)
A struct describing a spotlight.
- struct [StreamPlaylist](#)
- class [System](#)
An interface that represents a system of the engine.
- class [Texture](#)
- class [ThreadPool](#)
- struct [ThumbStick](#)
Thumbstick structure to hold X/Y axis information.
- class [Timer](#)
- class [VBO](#)
- struct [Vertex](#)
- class [Win32Window](#)
- class [Window](#)
- struct [WindowFunctionCallbacks](#)
- class [WindowingService](#)
- class [XInputDevice](#)

Typedefs

- using [Recti](#) = Vec4i
Type alias of a Vec4i.
- using **KeyframeVec3f** = [Keyframe](#)< Vec3f >
- using **KeyframeQuatf** = [Keyframe](#)< Quatf >
- using **LoadEntityCallback** = std::function< bool(const std::wstring &fileName, [Entity](#) *thisObject)>
- using **PointLightDescTuple** = std::tuple< [PointLightDesc](#) *, [LightComponent](#) * >
- using **DirectionalLightDescTuple** = std::tuple< [DirectionalLightDesc](#) *, [LightComponent](#) * >
- using **SpotlightDescTuple** = std::tuple< [SpotlightDesc](#) *, [LightComponent](#) * >
- using **OnNewClientCallback** = std::function< void([ConnectionInfo](#) connectionInfo)>
- using **OnNewPacketCallback** = std::function< void(std::vector< Byte >)>
- using **OnClientDisconnectCallback** = std::function< void()>
- using **ConnectionMap** = std::map< unsigned long, std::unique_ptr< [Socket](#) > >
- using **MessageQueue** = std::queue< std::shared_ptr< [NetworkMessage](#) > >
- using **SocketHandle** = int
- template<typename T >
using **TimePoint** = std::chrono::time_point< T >
- using **HighResolutionClock** = std::chrono::high_resolution_clock
- using **HighResolutionTimePoint** = std::chrono::time_point< std::chrono::high_resolution_clock >
- using **Vec2i** = glm::ivec2
- using **Vec3i** = glm::ivec3
- using **Vec4i** = glm::ivec4
- using **Vec2ui** = glm::uvec2
- using **Vec3ui** = glm::uvec3
- using **Vec4ui** = glm::uvec4
- using **Vec2f** = glm::vec2
- using **Vec3f** = glm::vec3
- using **Vec4f** = glm::vec4
- using **Vec2d** = glm::dvec2
- using **Vec3d** = glm::dvec3
- using **Vec4d** = glm::dvec4
- using **Mat2f** = glm::mat2
- using **Mat3f** = glm::mat3
- using **Mat4f** = glm::mat4
- using **Mat2d** = glm::dmat2
- using **Mat3d** = glm::dmat3
- using **Mat4d** = glm::dmat4
- using **Quatf** = glm::quat
- using **Quatd** = glm::dquat
- using **Byte** = char
- template<typename T >
using **ComPtr** = Microsoft::WRL::ComPtr< T >
- template<typename T >
using **MessageContainer** = [RefCountedContainer](#)< [Message](#)< T > >
- using **ReshapeFunc** = void(*) (int, int)
- using **KeyboardFunc** = void(*) (unsigned char, int, int)
- using **KeyboardUpFunc** = void(*) (unsigned char, int, int)
- using **SpecialFunc** = void(*) (int, int, int)
- using **SpecialUpFunc** = void(*) (int, int, int)
- using **MouseFunc** = void(*) (int, bool, int, int)
- using **MotionFunc** = void(*) (int, int)
- using **PassiveMotionFunc** = void(*) (int, int)
- using **AddRemoveInputDeviceFunc** = void(*) ()

Enumerations

- enum **DeviceType** { **KEYBAORD**, **JOYPAD**, **OTHER**, **DEVTYPE_ERROR** }
- enum **AnalogDeadzone** { **ANALOG_STICK_LEFT**, **ANALOG_STICK_RIGHT**, **ANALOG_TRIGGER** }
- enum **JoypadNumber** { **JOYPAD1**, **JOYPAD2**, **JOYPAD3**, **JOYPAD4** }
- enum **MouseButton** { **LEFT** = 0, **RIGHT** = 1 }
- enum **InputSensor** {
BTN_FACE_1 = JOYBTN_FACE1, **BTN_FACE_2** = JOYBTN_FACE2, **BTN_FACE_3** = JOYBTN_FACE3,
BTN_FACE_4 = JOYBTN_FACE4,
BTN_STICK_L = JOYBTN_STICKL, **BTN_STICK_R** = JOYBTN_STICKR, **BTN_SHOULDER_L** = JOYBT↵
N_SHOULDER1, **BTN_SHOULDER_R** = JOYBTN_SHOULDER2,
BTN_OPTION_1 = JOYBTN_OPTION1, **BTN_OPTION_2** = JOYBTN_OPTION2, **DPAD_UP** = JOYDPAD↵
_UP, **DPAD_DOWN** = JOYDPAD_DOWN,
DPAD_LEFT = JOYDPAD_LEFT, **DPAD_RIGHT** = JOYDPAD_RIGHT, **TRIGGER_LEFT**, **TRIGGER_RIG↵
HT**,
STICK_LEFT, **STICK_RIGHT** }
- enum **LightType** { **POINT**, **DIRECTIONAL**, **SPOTLIGHT** }
An enumeration used to specify a light type.
- enum **VertexWinding** { **CLOCKWISE**, **ANTICLOCKWISE** }
- enum **RenderStateType** {
BS_BLEND_DISABLED, **BS_BLEND_ADDITIVE**, **BS_BLEND_ALPHA**, **RS_CULL_FRONT**,
RS_CULL_BACK, **RS_DRAW_WIRE**, **RS_DRAW_SOLID**, **DSS_DEPTH_MASK_0**,
DSS_DEPTH_MASK_1, **DSS_DEPTH_TEST_DISABLE**, **DSS_DEPTH_TEST_ENABLE** }
- enum **RenderTargetBindType** { **COLOR_AND_DEPTH**, **DEPTH** }
- enum **ShaderType** {
VERTEX_SHADER, **HULL_SHADER**, **DOMAIN_SHADER**, **GEOMETRY_SHADER**,
FRAGMENT_SHADER, **SHADER_COUNT** }
- enum **InputLayoutMask** {
IL_POSITION = 0x02, **IL_NORMAL** = 0x04, **IL_TANGENT** = 0x08, **IL_TEXCOORD** = 0x10,
IL_COLOR = 0x20 }
- enum **AudioPlaymode** { **AUDIO_PLAYMODE_ONCE**, **AUDIO_PLAYMODE_LOOP** }
- enum **TextureType** {
TEX_DIFFUSE, **TEX_SPECULAR**, **TEX_NORMAL**, **TEX_EMISSION**,
TEX_AMBIENT_OCCLUSION, **SUPPORTED_TEX_COUNT** }
- enum **CpuCoreNumber** {
CPU_0 = 1, **CPU_1** = 2, **CPU_2** = 4, **CPU_3** = 8,
CPU_4 = 16, **CPU_5** = 32, **CPU_6** = 64, **CPU_7** = 128 }
- enum **PrimitiveTopology** { **TRIANGLE_LIST**, **TRIANGLE_STRIP** }
- enum **VirtualKey** : int {
KEY_BACKSPACE = VK_BACK, **KEY_TAB** = VK_TAB, **KEY_RETURN** = VK_RETURN, **KEY_PAUSE** =
VK_PAUSE,
KEY_ESC = VK_ESCAPE, **KEY_SPACE** = VK_SPACE, **KEY_PGUP** = VK_PRIOR, **KEY_PGDN** = VK_N↵
EXT,
KEY_END = VK_END, **KEY_HOME** = VK_HOME, **KEY_LEFT** = VK_LEFT, **KEY_RIGHT** = VK_RIGHT,
KEY_UP = VK_UP, **KEY_DOWN** = VK_DOWN, **KEY_SELECT** = VK_SELECT, **KEY_PRINT** = VK_PRINT,
KEY_PRTSCRN = VK_SNAPSHOT, **KEY_INSERT** = VK_INSERT, **KEY_DELETE** = VK_DELETE, **KEY_↵
HELP** = VK_HELP,
KEY_0 = 0x30, **KEY_1** = 0x31, **KEY_2** = 0x32, **KEY_3** = 0x33,
KEY_4 = 0x34, **KEY_5** = 0x35, **KEY_6** = 0x36, **KEY_7** = 0x37,
KEY_8 = 0x38, **KEY_9** = 0x39, **KEY_A** = 0x41, **KEY_B** = 0x42,
KEY_C = 0x43, **KEY_D** = 0x44, **KEY_E** = 0x45, **KEY_F** = 0x46,
KEY_G = 0x47, **KEY_H** = 0x48, **KEY_I** = 0x49, **KEY_J** = 0x4A,
KEY_K = 0x4B, **KEY_L** = 0x4C, **KEY_M** = 0x4D, **KEY_N** = 0x4E,
KEY_O = 0x4F, **KEY_P** = 0x50, **KEY_Q** = 0x51, **KEY_R** = 0x52,
KEY_S = 0x53, **KEY_T** = 0x54, **KEY_U** = 0x55, **KEY_V** = 0x56,
KEY_W = 0x57, **KEY_X** = 0x58, **KEY_Y** = 0x59, **KEY_Z** = 0x5A,

```

KEY_NUM_0 = VK_NUMPAD0, KEY_NUM_1 = VK_NUMPAD0, KEY_NUM_2 = VK_NUMPAD0, KEY_NUM_3 = VK_NUMPAD0,
KEY_NUM_4 = VK_NUMPAD0, KEY_NUM_5 = VK_NUMPAD0, KEY_NUM_6 = VK_NUMPAD0, KEY_NUM_7 = VK_NUMPAD0,
KEY_NUM_8 = VK_NUMPAD0, KEY_NUM_9 = VK_NUMPAD0, KEY_NUM_MULTIPLY = VK_MULTIPLY,
KEY_NUM_ADD = VK_ADD,
KEY_NUM_SUBTRACT = VK_SUBTRACT, KEY_NUM_DECIMAL = VK_DECIMAL, KEY_NUM_DIVIDE = VK_DIVIDE,
KEY_F1 = VK_F1,
KEY_F2 = VK_F2, KEY_F3 = VK_F3, KEY_F4 = VK_F4, KEY_F5 = VK_F5,
KEY_F6 = VK_F6, KEY_F7 = VK_F7, KEY_F8 = VK_F8, KEY_F9 = VK_F9,
KEY_F10 = VK_F10, KEY_F11 = VK_F11, KEY_F12 = VK_F12, KEY_F13 = VK_F13,
KEY_F14 = VK_F14, KEY_F15 = VK_F15, KEY_F16 = VK_F16, KEY_F17 = VK_F17,
KEY_F18 = VK_F18, KEY_F19 = VK_F19, KEY_F20 = VK_F20, KEY_F21 = VK_F21,
KEY_F22 = VK_F22, KEY_F23 = VK_F23, KEY_F24 = VK_F24, KEY_LSHIFT = VK_LSHIFT,
KEY_RSHIFT = VK_RSHIFT, KEY_LCTRL = VK_LCONTROL, KEY_RCTRL = VK_RCONTROL, KEY_LMENU = VK_LMENU,
KEY_RMENU = VK_RMENU, KEY_NUM_SEPR = VK_SEPARATOR, KEY_WIN_L = VK_LWIN, KEY_WIN_R = VK_RWIN,
KEY_APPS = VK_APPS }

```

Functions

- bool **AttachCurrentThreadToCore** (unsigned int coreNumber)

4.1.1 Detailed Description

Key values for keyboard input.

Chapter 5

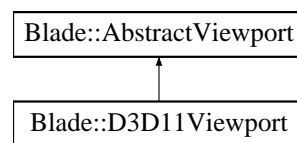
Class Documentation

5.1 Blade::AbstractViewport Class Reference

Describes an implementation agnostic Viewport.

```
#include <abstract_viewport.h>
```

Inheritance diagram for Blade::AbstractViewport:



Public Member Functions

- [AbstractViewport](#) ()=default
AbstractViewport's default constructor.
- [AbstractViewport](#) (const [Recti](#) &rect)
AbstractViewport's constructor.
- virtual [~AbstractViewport](#) ()=default
AbstractViewport's default destructor.
- const [Recti](#) & [GetRect](#) () const noexcept
Provides the dimensions of the Viewport.
- void [SetRect](#) (const [Recti](#) &rect) noexcept
Sets the dimensions of the Viewport.
- virtual void [Set](#) () const noexcept=0
Sets the Viewport to the Rasterizer.

5.1.1 Detailed Description

Describes an implementation agnostic Viewport.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 AbstractViewport()

```
Blade::AbstractViewport::AbstractViewport (
    const Recti & rect ) [inline], [explicit]
```

[AbstractViewport](#)'s constructor.

Parameters

<i>rect</i>	The dimensions of the Viewport.
-------------	---------------------------------

5.1.3 Member Function Documentation

5.1.3.1 GetRect()

```
const Recti & Blade::AbstractViewport::GetRect ( ) const [noexcept]
```

Provides the dimensions of the Viewport.

Returns

The dimensions of the Viewport.

5.1.3.2 SetRect()

```
void Blade::AbstractViewport::SetRect (
    const Recti & rect ) [noexcept]
```

Sets the dimensions of the Viewport.

Parameters

<i>rect</i>	The dimensions of the Viewport.
-------------	---------------------------------

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

- include/abstract_viewport.h
- src/abstract_viewport.cpp

5.2 Blade::Animation Class Reference

Public Member Functions

- **Animation** (const std::string &name, bool loopState)
- void **SetName** (const std::string &name)
- void **SetLoopping** (bool loopState)
- const std::string & **GetName** () const noexcept
- const [KeyframeVec3f](#) & **GetPositionKeyframe** (unsigned int idx) const noexcept
- const [KeyframeQuatf](#) & **GetRotationKeyframe** (unsigned int idx) const noexcept
- const [KeyframeVec3f](#) & **GetScalingKeyframe** (unsigned int idx) const noexcept
- size_t **GetPositionKeyframeCount** () const noexcept
- size_t **GetRotationKeyframeCount** () const noexcept
- size_t **GetScalingKeyframeCount** () const noexcept
- void **SetAnimationSpeed** (float speed) noexcept
- float **GetAnimationSpeed** () const noexcept
- bool **HasPositionKeyframes** () const noexcept
- bool **HasRotationKeyframes** () const noexcept
- bool **HasScalingKeyframes** () const noexcept
- bool **DoesLoop** () const noexcept
- void **AddPositionKeyframe** (const [KeyframeVec3f](#) &pos) noexcept
- void **AddRotationKeyframe** (const [KeyframeQuatf](#) &rot) noexcept
- void **AddScalingKeyframe** (const [KeyframeVec3f](#) &scaling) noexcept
- void **ReplacePositionKeyframe** (const [KeyframeVec3f](#) &pos, unsigned int idx) noexcept
- void **ReplaceRotationKeyframe** (const [KeyframeQuatf](#) &rot, unsigned int idx) noexcept
- void **ReplaceScalingKeyframe** (const [KeyframeVec3f](#) &scaling, unsigned int idx) noexcept
- void **ClearKeyframes** () noexcept
- void **SortPositionKeyframes** () noexcept
- void **SortRotationKeyframes** () noexcept
- void **SortScalingKeyframes** () noexcept

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

- include/animation.h
- src/animation.cpp

5.3 Blade::Application Class Reference

Public Member Functions

- **Application** (const [Application](#) &application)=delete
- [Application](#) & **operator=** (const [Application](#) &application)=delete
- void **SetTermination** (bool state) noexcept
- bool **ShouldTerminate** () const noexcept
- double **GetDelta** () const noexcept
- long **GetMsec** () const noexcept
- double **GetSec** () const noexcept
- void **Pause** () noexcept
- void **UnPause** () noexcept
- bool **IsPaused** () const noexcept
- void **SetLoadEntityCallback** (const LoadEntityCallback &callback) noexcept

- const LoadEntityCallback & **GetLoadEntityCallback** () const noexcept
- virtual bool **Initialize** (int *argc, char *argv[])
- virtual void **Update** () noexcept=0
- virtual void **Draw** () const noexcept=0
- virtual int **Run** () noexcept=0

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

- include/application.h
- src/application.cpp

5.4 Blade::AudioManager Class Reference

Public Member Functions

- void **SetSourcesVolume** (float volume)
- void **SetStreamsVolume** (float volume)
- void **SetMasterVolume** (float volume)
- [OggVorbisStream](#) * **GetAudioStream** (int idx)
- [AudioSource](#) * **GetAudioSource** (int idx)
- [AudioSource](#) * **GetAudioSource** ([AudioSample](#) *sample)
- void **PlayStream** (const std::wstring &fname, float volume, AudioPlaymode mode, int *stream_idx=nullptr)
- void **PlaySample** ([AudioSample](#) *sample, float volume, AudioPlaymode mode, const Vec3f &position=Vec3f{ 0, 0, 0 }, int *src_idx=nullptr)
- void **PlayStreamPlaylist** ([StreamPlaylist](#) *playlist, float volume)
- void **PlaySamplePlaylist** ([SamplePlaylist](#) *playlist, float volume)
- void **StopStream** (int stream_idx)
- void **StopSource** (int source_idx)
- void **StopStreams** ()
- void **StopSources** ()
- void **PauseStreams** ()
- void **PauseSources** ()
- void **ResumeStreams** ()
- void **ResumeSources** ()
- void **RegulateVolumes** ()

Static Public Member Functions

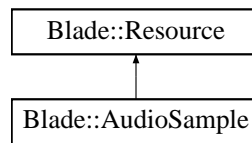
- static void **SetListenerPosition** (const Vec3f &pos=Vec3f{ 0, 0, 0 })
- static void **SetListenerOrientation** (const Vec3f &dir, const Vec3f &up=Vec3f{ 0, 1, 0 })

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

- include/audio_manager.h
- src/audio_manager.cpp

5.5 Blade::AudioSample Class Reference

Inheritance diagram for Blade::AudioSample:



Public Member Functions

- bool **Load** (const std::wstring &fileName) noexcept override
- unsigned int **GetBuffer** () const noexcept

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

- include/sample.h
- src/sample.cpp

5.6 Blade::AudioSource Class Reference

Public Member Functions

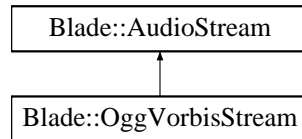
- void **SetSample** (const [AudioSample](#) *sample) noexcept
- const [AudioSample](#) * **GetSample** () const noexcept
- void **SetPosition** (const Vec3f &pos, bool viewspace=false) const noexcept
- Vec3f **GetPosition** () const noexcept
- void **SetVolume** (float vol) noexcept
- float **GetVolume** () const noexcept
- void **SetPlaybackVolume** (float vol) const noexcept
- float **GetPlaybackVolume** () const noexcept
- void **SetLooping** (bool state) const noexcept
- void **SetReferenceDist** (float rdist) const noexcept
- float **GetReferenceDist** () const noexcept
- void **SetMaxDist** (float dist) const noexcept
- float **GetMaxDist** () const noexcept
- bool **IsPlaying** () const noexcept
- bool **IsPaused** () const noexcept
- void **Play** () const noexcept
- void **Stop** () const noexcept
- void **Pause** () const noexcept

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

- include/source.h
- src/source.cpp

5.7 Blade::AudioStream Class Reference

Inheritance diagram for Blade::AudioStream:



Public Member Functions

- void **PollLoop** () noexcept
- void **SetVolume** (float vol) noexcept
- float **GetVolume** () const noexcept
- void **SetPlaybackVolume** (float vol) noexcept
- float **GetPlaybackVolume** () const noexcept
- virtual void **Play** (AudioPlaymode mode) noexcept
- virtual void **Stop** () noexcept
- virtual void **Rewind** () noexcept=0
- virtual bool **IsPlaying** () const noexcept
- virtual bool **IsLooping** () const noexcept
- virtual int **FreqCount** (int bin) const noexcept
- virtual int **FreqCount** (int range_start, int range_end) const noexcept

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

- include/stream.h
- src/stream.cpp

5.8 Blade::AudioStreamBuffer Struct Reference

Public Attributes

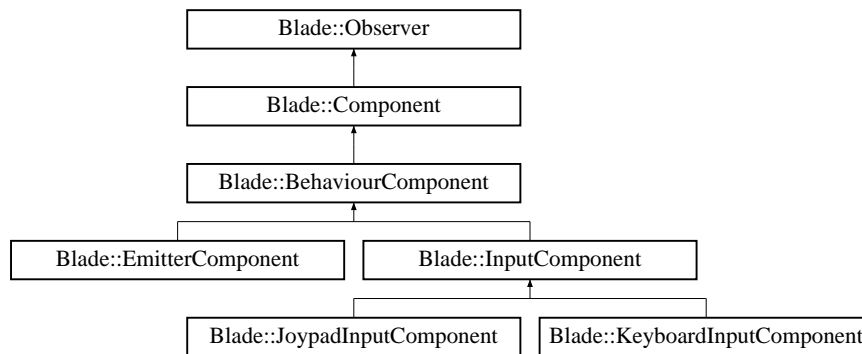
- Byte **samples** [AUDIO_BUFFER_BYTES]
- int **sampleCount**
- int **channels**
- int **sampleRate**

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

- include/stream.h

5.9 Blade::BehaviourComponent Class Reference

Inheritance diagram for Blade::BehaviourComponent:



Public Member Functions

- **BehaviourComponent** (const std::string &type, Entity *parent)
Component constructor.
- **BehaviourComponent** (const BehaviourComponent &other)=delete
- **BehaviourComponent & operator=** (const BehaviourComponent &other)=delete
- virtual void **Update** (const float dt, const long time=0) noexcept=0
Updates the Component on each frame.
- virtual void **Setup** () noexcept=0
Performs setup actions after the BehaviourComponent's creation.
- virtual void **Teardown** () noexcept=0
Performs actions before the BehaviourComponent is destroyed.
- virtual void **OnCollision** (Entity *other) noexcept

5.9.1 Constructor & Destructor Documentation

5.9.1.1 BehaviourComponent()

```

Blade::BehaviourComponent::BehaviourComponent (
    const std::string & type,
    Entity * parent )

```

Component constructor.

Parameters

<i>type</i>	The type of the Component as a string.
<i>parent</i>	The Entity the Component will be attached to.

5.9.2 Member Function Documentation

5.9.2.1 Update()

```
virtual void Blade::BehaviourComponent::Update (
    const float dt,
    const long time = 0 ) [pure virtual], [noexcept]
```

Updates the [Component](#) on each frame.

Parameters

<i>dt</i>	The time elapsed from the previous frame of the Application .
<i>time</i>	The elapsed time since the start of the Application .

Implemented in [Blade::EmitterComponent](#), [Blade::InputComponent](#), [Blade::JoypadInputComponent](#), and [Blade::KeyboardInputComponent](#).

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

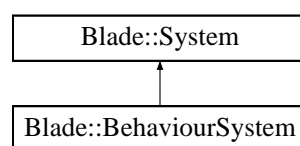
- include/behaviour_component.h
- src/behaviour_component.cpp

5.10 Blade::BehaviourSystem Class Reference

A [System](#) responsible to process and manage the BehaviourComponents by calling the Update method on every component.

```
#include <behaviour_system.h>
```

Inheritance diagram for Blade::BehaviourSystem:



Public Member Functions

- void [Process](#) (float deltaTime=.0f, long time=0) noexcept override
Processes the [BehaviourComponent](#).
- bool [Initialize](#) () noexcept override
Initializes the [BehaviourSystem](#).
- void [RegisterComponent](#) ([BehaviourComponent](#) *behaviourComponent) noexcept
Registers the specified [BehaviourComponent](#) to the [BehaviourSystem](#).
- void [UnregisterComponent](#) (int id) noexcept
Unregisters a [BehaviourComponent](#) from the [BehaviourSystem](#).
- virtual void [Setup](#) () noexcept
Setup all the [BehaviourComponent](#) that are currently registered with the [BehaviourSystem](#).
- virtual void [Teardown](#) () noexcept
Teardown all the [BehaviourComponent](#) that are currently registered with the [BehaviourSystem](#).

5.10.1 Detailed Description

A [System](#) responsible to process and manage the BehaviourComponents by calling the Update method on every component.

5.10.2 Member Function Documentation

5.10.2.1 Initialize()

```
bool Blade::BehaviourSystem::Initialize ( ) [override], [virtual], [noexcept]
```

Initializes the [BehaviourSystem](#).

Returns

TRUE if initialization is successful, FALSE otherwise.

Implements [Blade::System](#).

5.10.2.2 Process()

```
void Blade::BehaviourSystem::Process (
    float deltaTime = .0f,
    long time = 0 ) [override], [virtual], [noexcept]
```

Processes the [BehaviourComponent](#).

Parameters

<i>deltaTime</i>	The time elapsed from the previous frame of the application.
------------------	--

Implements [Blade::System](#).

5.10.2.3 RegisterComponent()

```
void Blade::BehaviourSystem::RegisterComponent (
    BehaviourComponent * behaviourComponent ) [noexcept]
```

Registers the specified [BehaviourComponent](#) to the [BehaviourSystem](#).

Parameters

<i>behaviourComponent</i>	The BehaviourComponent to be registered to the BehaviourSystem for processing.
---------------------------	--

5.10.2.4 UnregisterComponent()

```
void Blade::BehaviourSystem::UnregisterComponent (
    int id ) [noexcept]
```

Unregisters a [BehaviourComponent](#) from the [BehaviourSystem](#).

Parameters

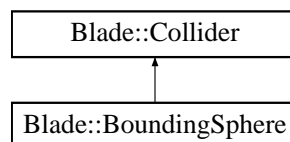
<i>id</i>	The unique id of the BehaviourComponent to be unregistered.
-----------	---

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

- include/behaviour_system.h
- src/behaviour_sytem.cpp

5.11 Blade::BoundingSphere Class Reference

Inheritance diagram for Blade::BoundingSphere:



Public Member Functions

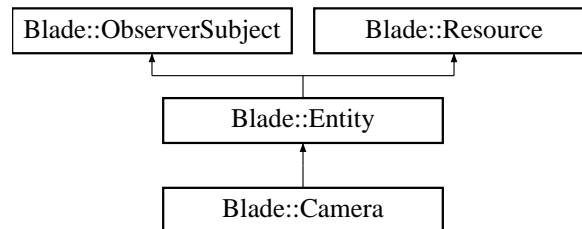
- **BoundingSphere** (float radius)
- bool **Collide** (const [Collider](#) *collider, [ContactManifold](#) &manifold) const noexcept override
- bool **Collide** (const [BoundingSphere](#) *bsphere, [ContactManifold](#) &manifold) const noexcept override
- bool **Collide** (const [PlaneCollider](#) *plane, [ContactManifold](#) &manifold) const noexcept override
- const float **GetRadius** () const noexcept

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

- include/bounding_sphere.h
- src/bounding_sphere.cpp

5.12 Blade::Camera Class Reference

Inheritance diagram for Blade::Camera:



Public Member Functions

- **Camera** (const std::string &name, const [CameraDesc](#) &cameraDescription)
- void **Update** (float dt, long time=0) noexcept override

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

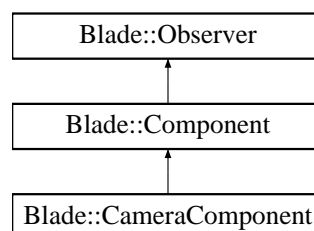
- include/camera.h
- src/camera.cpp

5.13 Blade::CameraComponent Class Reference

Represents a [CameraComponent](#). This component contains all the information needed for the view and projection transformations. Managed by the [CameraSystem](#).

```
#include <camera_component.h>
```

Inheritance diagram for Blade::CameraComponent:



Public Member Functions

- [CameraComponent](#) ([Entity](#) *parent)
CameraComponent's constructor.
- [CameraComponent](#) ([Entity](#) *parent, float fov, const Viewport &viewport, float nearPlane, float farPlane)
- [CameraComponent](#) ([Entity](#) *parent, float fov, const Viewport &viewport, const Vec2f &clippingPlanes)
- [~CameraComponent](#) ()
CameraComponent's destructor.
- float [GetFov](#) () const noexcept
Provides the field of view.
- void [SetFov](#) (float fov) noexcept
Sets the field of view.
- const Viewport & [GetViewport](#) () const noexcept
Provides the Viewport.
- void [SetViewport](#) (const Viewport &viewport) noexcept
Sets the Viewport.
- const Vec2f & [GetClippingPlanes](#) () const noexcept
Provides the clipping planes as a Vec2f.
- void [SetClippingPlanes](#) (float nearPlane, float farPlane) noexcept
Sets the near and the far clipping planes.
- void [SetClippingPlanes](#) (const Vec2f &clippingPlanes) noexcept
- float [GetNearPlane](#) () const noexcept
Provides the near clipping plane.
- void [SetNearPlane](#) (float nearPlane) noexcept
Sets the near clipping plane.
- float [GetFarPlane](#) () const noexcept
Provides the far clipping plane.
- void [SetFarPlane](#) (float farPlane) noexcept
Sets the far clipping plane.
- const Mat4f & [GetViewMatrix](#) () const noexcept
Provides the view matrix.
- void [SetViewMatrix](#) (const Mat4f &viewMatrix) noexcept
Sets the view matrix.
- const Mat4f & [GetProjectionMatrix](#) () const noexcept
Provides the projection matrix.
- void [UsePerspectiveProjection](#) () noexcept
Set the projection matrix with perspectiveLH.

5.13.1 Detailed Description

Represents a [CameraComponent](#). This component contains all the information needed for the view and projection transformations. Managed by the [CameraSystem](#).

5.13.2 Constructor & Destructor Documentation

5.13.2.1 CameraComponent() [1/3]

```
Blade::CameraComponent::CameraComponent (
    Entity * parent ) [explicit]
```

[CameraComponent](#)'s constructor.

Registers the component to the [CameraSystem](#).

Parameters

<i>parent</i>	The entity the CameraComponent will be attached to.
---------------	---

5.13.2.2 CameraComponent() [2/3]

```
Blade::CameraComponent::CameraComponent (
    Entity * parent,
    float fov,
    const Viewport & viewport,
    float nearPlane,
    float farPlane )
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

<i>parent</i>	The entity the CameraComponent will be attached to.
<i>fov</i>	The field of view.
<i>viewport</i>	The viewport of the camera.
<i>nearPlane</i>	The near clipping plane.
<i>farPlane</i>	The far clipping plane.

5.13.2.3 CameraComponent() [3/3]

```
Blade::CameraComponent::CameraComponent (
    Entity * parent,
    float fov,
    const Viewport & viewport,
    const Vec2f & clippingPlanes )
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

<i>parent</i>	The entity the CameraComponent will be attached to.
<i>fov</i>	The field of view.
<i>viewport</i>	The viewport of the camera.
<i>clippingPlanes</i>	The clipping planes for the projection.

clippingPlanes.x - The near clipping plane.

clippingPlanes.y - The far clipping plane.

5.13.2.4 ~CameraComponent()

```
Blade::CameraComponent::~~CameraComponent ( )
```

[CameraComponent](#)'s destructor.

Unregisters the component from the [CameraSystem](#).

5.13.3 Member Function Documentation

5.13.3.1 GetClippingPlanes()

```
const Vec2f & Blade::CameraComponent::GetClippingPlanes ( ) const [noexcept]
```

Provides the clipping planes as a Vec2f.

x - The near clipping plane.

y - The far clipping plane.

Returns

The clipping planes as a Vec2f.

5.13.3.2 GetFarPlane()

```
float Blade::CameraComponent::GetFarPlane ( ) const [noexcept]
```

Provides the far clipping plane.

Returns

The far clipping plane.

5.13.3.3 GetFov()

```
float Blade::CameraComponent::GetFov ( ) const [noexcept]
```

Provides the field of view.

Returns

The field of view.

5.13.3.4 GetNearPlane()

```
float Blade::CameraComponent::GetNearPlane ( ) const [noexcept]
```

Provides the near clipping plane.

Returns

The near clipping plane.

5.13.3.5 GetProjectionMatrix()

```
const Mat4f & Blade::CameraComponent::GetProjectionMatrix ( ) const [noexcept]
```

Provides the projection matrix.

Returns

The projection matrix.

5.13.3.6 GetViewMatrix()

```
const Mat4f & Blade::CameraComponent::GetViewMatrix ( ) const [noexcept]
```

Provides the view matrix.

Returns

The view matrix.

5.13.3.7 GetViewport()

```
const Viewport & Blade::CameraComponent::GetViewport ( ) const [noexcept]
```

Provides the Viewport.

Returns

The Viewport.

5.13.3.8 SetClippingPlanes() [1/2]

```
void Blade::CameraComponent::SetClippingPlanes (
    float nearPlane,
    float farPlane ) [noexcept]
```

Sets the near and the far clipping planes.

Parameters

<i>nearPlane</i>	The near clipping plane.
<i>farPlane</i>	The far clipping plane.

5.13.3.9 SetClippingPlanes() [2/2]

```
void Blade::CameraComponent::SetClippingPlanes (
    const Vec2f & clippingPlanes ) [noexcept]
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

x - The near clipping plane.

y - The far clipping plane.

Parameters

<i>clippingPlanes</i>	The clipping planes as a Vec2f.
-----------------------	---------------------------------

5.13.3.10 SetFarPlane()

```
void Blade::CameraComponent::SetFarPlane (
    float farPlane ) [noexcept]
```

Sets the far clipping plane.

Parameters

<i>farPlane</i>	The far clipping plane.
-----------------	-------------------------

5.13.3.11 SetFov()

```
void Blade::CameraComponent::SetFov (
    float fov ) [noexcept]
```

Sets the field of view.

Parameters

<i>fov</i>	The field of view.
------------	--------------------

5.13.3.12 SetNearPlane()

```
void Blade::CameraComponent::SetNearPlane (
    float nearPlane ) [noexcept]
```

Sets the near clipping plane.

Parameters

<i>nearPlane</i>	The near clipping plane.
------------------	--------------------------

5.13.3.13 SetViewMatrix()

```
void Blade::CameraComponent::SetViewMatrix (
    const Mat4f & viewMatrix ) [noexcept]
```

Sets the view matrix.

Parameters

<i>farPlane</i>	The new view matrix
-----------------	---------------------

5.13.3.14 SetViewport()

```
void Blade::CameraComponent::SetViewport (
    const Viewport & viewport ) [noexcept]
```

Sets the Viewport.

Parameters

<i>viewport</i>	The Viewport.
-----------------	---------------

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

- include/camera_component.h
- src/camera_component.cpp

5.14 Blade::CameraDesc Struct Reference

Public Attributes

- Viewport **viewport**
- float **nearPlane**
- float **farPlane**
- float **fov**

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

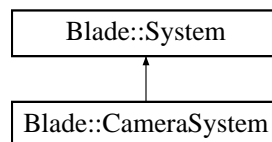
- include/camera.h

5.15 Blade::CameraSystem Class Reference

A [System](#) responsible to process and manage the CameraComponents by swapping the current active camera and providing the current active camera's matrices.

```
#include <camera_system.h>
```

Inheritance diagram for Blade::CameraSystem:



Public Member Functions

- void [RegisterComponent](#) ([CameraComponent](#) *cameraComponent) noexcept
Registers the specified [CameraComponent](#) to the [CameraSystem](#).
- void [UnregisterComponent](#) (int id) noexcept
Unregisters a [CameraComponent](#) from the [CameraSystem](#).
- void [SetActiveCamera](#) (const std::string &name) noexcept
Set the camera with the specified name as the active camera.
- const Mat4f & [GetActiveCameraViewMatrix](#) () const noexcept
Provides the active camera's view matrix.
- const Mat4f & [GetActiveCameraProjectionMatrtix](#) () const noexcept
Provides the active camera's projection matrix.
- const Viewport & [GetActiveCameraViewport](#) () const noexcept
Provides the active camera's Viewport.
- [CameraComponent](#) * [GetActiveCamera](#) () const noexcept
Provides the active camera's [CameraComponent](#).
- [CameraComponent](#) * [GetCamera](#) (const std::string &name) noexcept
Provides the [CameraComponent](#) of the camera with the specified name.

5.15.1 Detailed Description

A [System](#) responsible to process and manage the CameraComponents by swapping the current active camera and providing the current active camera's matrices.

5.15.2 Member Function Documentation

5.15.2.1 GetActiveCamera()

```
CameraComponent * Blade::CameraSystem::GetActiveCamera ( ) const [noexcept]
```

Provides the active camera's [CameraComponent](#).

Returns

The active camera's [CameraComponent](#).

5.15.2.2 GetActiveCameraProjectionMatrtix()

```
const Mat4f & Blade::CameraSystem::GetActiveCameraProjectionMatrtix ( ) const [noexcept]
```

Provides the active camera's projection matrix.

Returns

The active camera's projection matrix.

5.15.2.3 GetActiveCameraViewMatrix()

```
const Mat4f & Blade::CameraSystem::GetActiveCameraViewMatrix ( ) const [noexcept]
```

Provides the active camera's view matrix.

Returns

The active camera's view matrix.

5.15.2.4 GetActiveCameraViewport()

```
const Viewport & Blade::CameraSystem::GetActiveCameraViewport ( ) const [noexcept]
```

Provides the active camera's Viewport.

Returns

The active camera's Viewport.

5.15.2.5 GetCamera()

```
CameraComponent * Blade::CameraSystem::GetCamera (
    const std::string & name ) [noexcept]
```

Provides the [CameraComponent](#) of the camera with the specified name.

Parameters

<i>name</i>	The name of the camera to be returned.
-------------	--

Returns

The [CameraComponent](#) of the camera with the specified name.

5.15.2.6 RegisterComponent()

```
void Blade::CameraSystem::RegisterComponent (
    CameraComponent * cameraComponent ) [noexcept]
```

Registers the specified [CameraComponent](#) to the [CameraSystem](#).

Parameters

<i>cameraComponent</i>	The CameraComponent to be registered to the CameraSystem for processing.
------------------------	--

5.15.2.7 SetActiveCamera()

```
void Blade::CameraSystem::SetActiveCamera (
    const std::string & name ) [noexcept]
```

Set the camera with the specified name as the active camera.

Parameters

<i>name</i>	The name of the camera to be set as active.
-------------	---

5.15.2.8 UnregisterComponent()

```
void Blade::CameraSystem::UnregisterComponent (
    int id ) [noexcept]
```

Unregisters a [CameraComponent](#) from the [CameraSystem](#).

Parameters

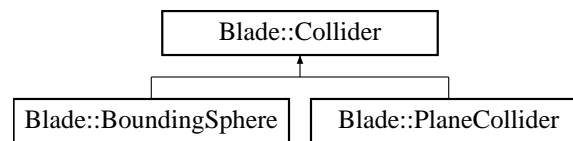
<i>id</i>	The unique id of the CameraComponent to be unregistered.
-----------	--

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

- include/camera_system.h
- src/camera_system.cpp

5.16 Blade::Collider Class Reference

Inheritance diagram for Blade::Collider:



Public Member Functions

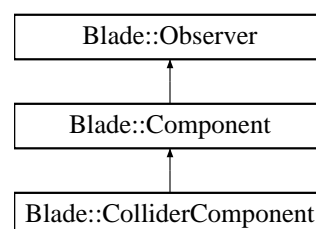
- virtual bool **Collide** (const [Collider](#) *collider, [ContactManifold](#) &manifold) const noexcept=0
- virtual bool **Collide** (const [BoundingSphere](#) *bsphere, [ContactManifold](#) &manifold) const noexcept=0
- virtual bool **Collide** (const [PlaneCollider](#) *plane, [ContactManifold](#) &manifold) const noexcept=0
- [ColliderComponent](#) * **GetColliderComponent** () const noexcept
- void **SetParent** ([ColliderComponent](#) *cc) noexcept

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

- include/collider.h

5.17 Blade::ColliderComponent Class Reference

Inheritance diagram for Blade::ColliderComponent:



Public Member Functions

- **ColliderComponent** ([Entity](#) *parent, std::unique_ptr< [Collider](#) > collider)
- **ColliderComponent** ([ColliderComponent](#) &)=delete
- [ColliderComponent](#) & **operator=** ([ColliderComponent](#) &)=delete
- void **SetCollider** (std::unique_ptr< [Collider](#) > collider) noexcept
- [Collider](#) * **GetCollider** () const noexcept
- bool **IsActive** () const noexcept
- void **SetCollisionResponseFlag** (bool flag) noexcept
- void **AddListener** ([BehaviourComponent](#) *listener) noexcept
- void **NotifyCollisionListeners** ([Entity](#) *entity) noexcept

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

- include/collider_component.h
- src/collider_component.cpp

5.18 Blade::Command Class Reference

Public Member Functions

- **Command** (bool online=false)
- virtual void **Execute** ([Entity](#) *entity, const float dt)=0

Protected Attributes

- bool **m_Online**

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

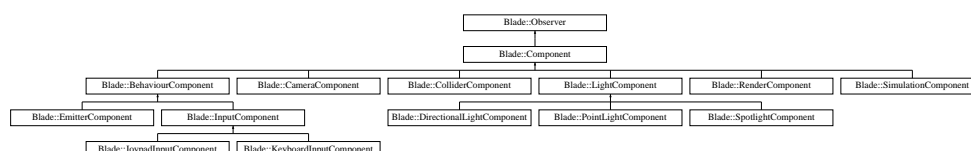
- include/command.h

5.19 Blade::Component Class Reference

Base [Component](#) class of the engine. All the components of the engine derive from this class. Component inherits from the [Observer](#) class so it can register and receive specific messages.

```
#include <component.h>
```

Inheritance diagram for Blade::Component:



Public Member Functions

- [Component](#) (const std::string &type, [Entity](#) *parent)
Component constructor.
- **Component** (const [Component](#) &other)=delete
- [Component](#) & **operator=** (const [Component](#) &other)=delete
- virtual [~Component](#) ()
Default destructor of the Component.
- const std::string & [GetType](#) () const noexcept
Returns the type of the Component.
- [Entity](#) * [GetParent](#) () const noexcept
Returns the Entity that the Component is attached to.
- void **SetParent** ([Entity](#) *parent) noexcept
- int [GetId](#) () const noexcept
Returns the unique Component ID.
- void [OnMessage](#) (const [MessageContainer](#)< std::string > &msg) override
Broadcasts the recieved message to the current active Scene through the SceneManager.

5.19.1 Detailed Description

Base [Component](#) class of the engine. All the components of the engine derive from this class. Component inherits from the [Observer](#) class so it can register and receive specific messages.

5.19.2 Constructor & Destructor Documentation

5.19.2.1 Component()

```
Blade::Component::Component (
    const std::string & type,
    Entity * parent )
```

[Component](#) constructor.

Parameters

<i>type</i>	The type of the Component as a string.
<i>parent</i>	The Entity the Component will be attached to.

5.19.3 Member Function Documentation

5.19.3.1 GetId()

```
int Blade::Component::GetId ( ) const [noexcept]
```

Returns the unique [Component](#) ID.

Returns

The unique [Component](#) ID.

5.19.3.2 GetParent()

```
Entity * Blade::Component::GetParent ( ) const [noexcept]
```

Returns the [Entity](#) that the [Component](#) is attached to.

Returns

The [Entity](#) that the [Component](#) is attached to.

5.19.3.3 GetType()

```
const std::string & Blade::Component::GetType ( ) const [noexcept]
```

Returns the type of the [Component](#).

Returns

The type of the [Component](#).

5.19.3.4 OnMessage()

```
void Blade::Component::OnMessage (
    const MessageContainer< std::string > & msg ) [override], [virtual]
```

Broadcasts the recieved message to the current active [Scene](#) through the [SceneManager](#).

Parameters

<i>msg</i>	The message received.
------------	-----------------------

Implements [Blade::Observer](#).

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

- include/component.h
- src/component.cpp

5.20 Blade::ConfigEntry Class Reference

Public Member Functions

- **ConfigEntry** (const char *name, const char *value)
- bool **IsValid** () const
- const char * **GetName** () const
- const char * **GetValueString** () const
- bool **IsNumber** () const
- int **GetValueInt** () const
- float **GetValueFloat** () const
- Vec4f **GetValueVec4f** () const

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

- include/cfg.h
- src/cfg.cpp

5.21 Blade::ConfigFile Class Reference

Public Member Functions

- **ConfigFile** (const char *fname)
- bool **Open** (const char *fname)
- bool **IsOpen** () const
- [ConfigEntry](#) **Get** (const char *optname) const
- std::list< [ConfigEntry](#) > **GetAll** (const char *groupname) const
- const char * **GetString** (const char *optname, const char *def=nullptr) const
- int **GetInteger** (const char *optname, int def=0) const
- float **GetFloat** (const char *optname, float def=0.0f) const
- Vec4f **GetVec4f** (const char *optname, const Vec4f &def=Vec4f{ 0.0f, 0.0f, 0.0f, 1.0f }) const
- void **SetNcf** ([NCF](#) *n)
- [NCF](#) * **GetNcf** ()

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

- include/cfg.h
- src/cfg.cpp

5.22 Blade::ConnectionInfo Struct Reference

Public Attributes

- `std::tuple< std::string, unsigned long >` **ip**
- unsigned short **port**

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

- `include/socket.h`

5.23 Blade::ContactManifold Class Reference

Public Member Functions

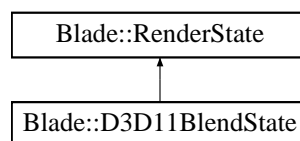
- void **AddEntry** (const [ManifoldEntry](#) &manifoldEntry) noexcept
- const [ManifoldEntry](#) & **GetEntry** (const int index) const noexcept
- const [ManifoldEntry](#) & **operator[]** (const int index) const noexcept
- const size_t **Size** () const noexcept
- void **Clear** () noexcept

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

- `include/contact_manifold.h`
- `src/contact_manifold.cpp`

5.24 Blade::D3D11BlendState Class Reference

Inheritance diagram for Blade::D3D11BlendState:



Public Member Functions

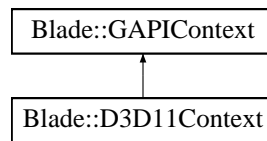
- **D3D11BlendState** (RenderStateType render_state_type)
- void **Set** () const noexcept override

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

- `include/d3d/D3D11_blend_state.h`
- `src/d3d/D3D11_blend_state.cpp`

5.25 Blade::D3D11Context Class Reference

Inheritance diagram for Blade::D3D11Context:



Public Member Functions

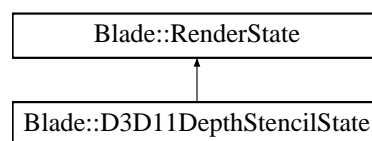
- bool **Create** (LUID *luid) override
- ID3D11Device * **GetDevice** () const
- ID3D11DeviceContext * **GetDeviceContext** () const
- ID3D11Debug * **GetDebugInterface** () const noexcept
- ID3D11Texture2D * **GetBackBuffer** () const noexcept
- ID3D11Texture2D ** **GetAddressOfBackBuffer** () noexcept
- ID3D11RenderTargetView * **GetDefaultRenderTargetView** () const noexcept
- ID3D11DepthStencilView * **GetDefaultDepthStencilView** () const noexcept
- ID3D11RenderTargetView ** **GetGetAddressOfDefaultRenderTargetView** () noexcept
- ID3D11DepthStencilView ** **GetAddressOfDefaultDepthStencilView** () noexcept
- unsigned int **GetMSAAQuality** (int sample_count) const

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

- include/d3d/D3D11_context.h
- src/d3d/D3D11_context.cpp

5.26 Blade::D3D11DepthStencilState Class Reference

Inheritance diagram for Blade::D3D11DepthStencilState:



Public Member Functions

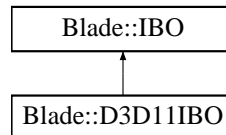
- **D3D11DepthStencilState** (RenderStateType renderStateType)
- void **Set** () const noexcept override

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

- include/d3d/D3D11_depth_stencil_state.h
- src/d3d/D3D11_depth_stencil_state.cpp

5.27 Blade::D3D11IBO Class Reference

Inheritance diagram for Blade::D3D11IBO:



Public Member Functions

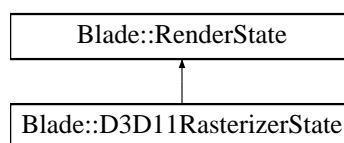
- bool **Create** (const std::vector< unsigned int > &indices) noexcept override
- void **Bind** () const noexcept override
- void **Draw** () const noexcept override

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

- include/d3d/D3D11_IBO.h
- src/d3d/D3D11_IBO.cpp

5.28 Blade::D3D11RasterizerState Class Reference

Inheritance diagram for Blade::D3D11RasterizerState:



Public Member Functions

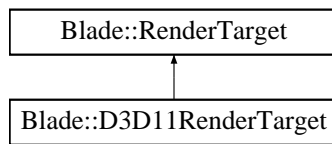
- **D3D11RasterizerState** (RenderStateType renderStateType)
- void **Set** () const noexcept override

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

- include/d3d/D3D11_rasterizer_state.h
- src/d3d/D3D11_rasterizer_state.cpp

5.29 Blade::D3D11RenderTarget Class Reference

Inheritance diagram for Blade::D3D11RenderTarget:



Public Member Functions

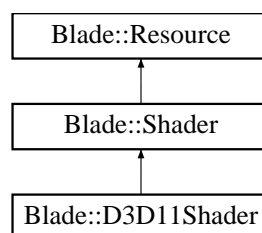
- **D3D11RenderTarget** (const Vec2i &size, bool MSAA, int sampleCount)
- bool **Create** (const Vec2i &size) override
- bool **Bind** (RenderTargetBindType bindType) const override
- bool **Unbind** () const override
- void **Clear** (float *color) const noexcept
- void **SetColorAttachment** (ID3D11Texture2D *colorAttachment, DXGI_FORMAT format) noexcept
- ID3D11ShaderResourceView * **GetColorAttachment** () const noexcept
- ID3D11ShaderResourceView * **GetDepthAttachment** () const noexcept

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

- include/d3d/D3D11_render_target.h
- src/d3d/D3D11_render_target.cpp

5.30 Blade::D3D11Shader Class Reference

Inheritance diagram for Blade::D3D11Shader:



Public Member Functions

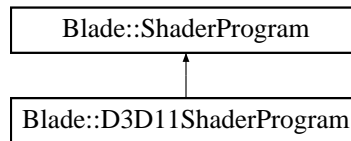
- ID3DBlob * **GetBlob** () const noexcept
- bool **Load** (const std::wstring &fileName) noexcept override

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

- include/d3d/D3D11_shader.h
- src/d3d/D3D11_shader.cpp

5.31 Blade::D3D11ShaderProgram Class Reference

Inheritance diagram for Blade::D3D11ShaderProgram:



Public Member Functions

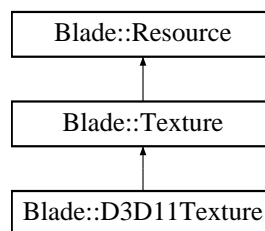
- bool **Create** (const [ShaderProgramDesc](#) &shaderProgramDesc) noexcept override
- void **Bind** () const noexcept override

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

- include/d3d/D3D11_shader_program.h
- src/d3d/D3D11_shader_program.cpp

5.32 Blade::D3D11Texture Class Reference

Inheritance diagram for Blade::D3D11Texture:



Public Member Functions

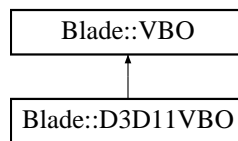
- **D3D11Texture** (TextureType textureType)
- bool **Load** (const std::wstring &fileName) noexcept override
- void **Bind** () const noexcept override

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

- include/d3d/D3D11_texture.h
- src/d3d/D3D11_texture.cpp

5.33 Blade::D3D11VBO Class Reference

Inheritance diagram for Blade::D3D11VBO:



Public Member Functions

- bool **Create** (const std::vector< [Vertex](#) > &vertices, PrimitiveTopology primitiveTopology) noexcept override
- void **Bind** () const noexcept override
- void **Draw** () const noexcept override

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

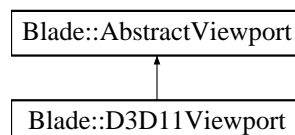
- include/d3d/D3D11_VBO.h
- src/d3d/D3D11_VBO.cpp

5.34 Blade::D3D11Viewport Class Reference

D3D11 implementation of the [AbstractViewport](#).

```
#include <D3D11_viewport.h>
```

Inheritance diagram for Blade::D3D11Viewport:



Public Member Functions

- [D3D11Viewport](#) ()=default
D3D11Viewport default constructor.
- [D3D11Viewport](#) (const [Recti](#) &rect, float minDepth, float maxDepth)
D3D11Viewport constructor.
- void [Set](#) () const noexcept override
Sets the Viewport to the Rasterizer.

5.34.1 Detailed Description

D3D11 implementation of the [AbstractViewport](#).

5.34.2 Constructor & Destructor Documentation

5.34.2.1 D3D11Viewport()

```
Blade::D3D11Viewport::D3D11Viewport (
    const Recti & rect,
    float minDepth,
    float maxDepth ) [inline]
```

[D3D11Viewport](#) constructor.

Parameters

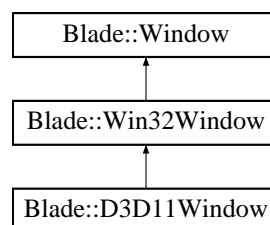
<i>rect</i>	The dimensions of the viewport.
<i>minDepth</i>	The minimum value of the depth buffer.
<i>maxDepth</i>	The maximum value of the depth buffer.

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

- include/d3d/D3D11_viewport.h
- src/d3d/D3D11_viewport.cpp

5.35 Blade::D3D11Window Class Reference

Inheritance diagram for Blade::D3D11Window:



Public Member Functions

- **D3D11Window** (const std::wstring &title, const Vec2i &size, const Vec2i &position, const unsigned int windowId, const bool focused, const bool minimized, const bool resizeable, const bool showCursor, const bool enableMSAA, const int msaaSampleCount, const [WindowFunctionCallbacks](#) &callbacks)
- void **EnableMSAA** (bool state) noexcept
- bool **MSAAEnabled** () const noexcept
- int **GetSampleCount** () const noexcept
- unsigned int **GetMSAAQuality** () const noexcept
- void **SwapBuffers** (unsigned syncInterval) const noexcept override

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

- include/d3d/D3D11_window.h
- src/d3d/D3D11_window.cpp

5.36 Blade::MathUtils::Derivative Struct Reference

Public Attributes

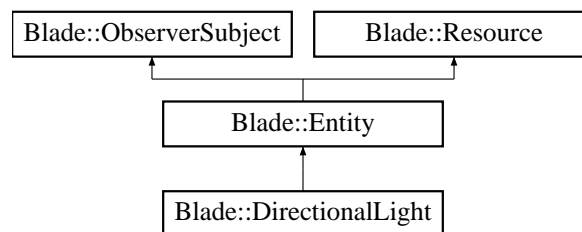
- float **dx** { 0.0f }
- float **dv** { 0.0f }

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

- include/math_utils.h

5.37 Blade::DirectionalLight Class Reference

Inheritance diagram for Blade::DirectionalLight:



Public Member Functions

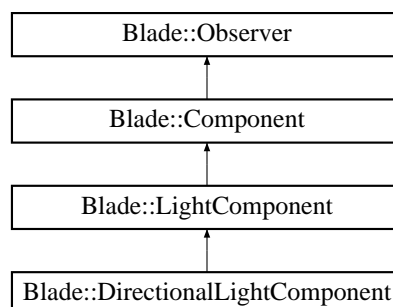
- **DirectionalLight** (const std::string &name, const [DirectionalLightDesc](#) &lightDescription)

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

- include/directional_light.h
- src/directional_light.cpp

5.38 Blade::DirectionalLightComponent Class Reference

Inheritance diagram for Blade::DirectionalLightComponent:



Public Member Functions

- **DirectionalLightComponent** (const [DirectionalLightDesc](#) &lightDesc, [Entity](#) *parent)
- const [DirectionalLightDesc](#) & **GetLightDescription** () const noexcept
- [DirectionalLightDesc](#) * **GetLightDescriptionPtr** () noexcept

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

- include/directional_light_component.h
- src/directional_light_component.cpp

5.39 Blade::DirectionalLightDesc Struct Reference

A struct describing a directional light.

```
#include <light_component.h>
```

Public Attributes

- Vec4f **ambientIntensity**
- Vec4f **diffuseIntensity**
- Vec4f **specularIntensity**
- Vec3f **direction**
- float **pad**

5.39.1 Detailed Description

A struct describing a directional light.

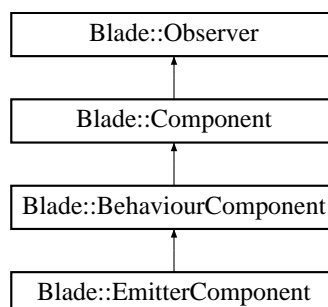
This struct is also used to represent a directional light in shaders.

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

- include/light_component.h

5.40 Blade::EmitterComponent Class Reference

Inheritance diagram for Blade::EmitterComponent:



Public Member Functions

- **EmitterComponent** ([Entity](#) *parent)
- **EmitterComponent** ([Entity](#) *entity, [EmitterDescriptor](#) descriptor)
- **EmitterComponent** (const [EmitterComponent](#) &other)=default
- [EmitterComponent](#) & **operator=** (const [EmitterComponent](#) &other)=default
- const std::vector< [Particle](#) > & **GetParticles** () const noexcept
- const [EmitterDescriptor](#) & **GetEmitterDescriptor** () const noexcept
- void **SetDescriptor** (const [EmitterDescriptor](#) &descriptor) noexcept
- float **GetSpawnRate** () const noexcept
- void **SetSpawnRate** (const float spawnRate) noexcept
- float **GetLifeSpan** () const noexcept
- void **SetLifeSpan** (const float lifespan) noexcept
- float **GetMaxParticles** () const noexcept
- void **SetMaxParticles** (const float maxParticles) noexcept
- float **GetSpawnRadius** () const noexcept
- void **SetSpawnRadius** (const float spawnRadius) noexcept
- float **GetParticleSize** () const noexcept
- void **SetParticleSize** (const float particleSize) noexcept
- const Vec4f & **GetStartColor** () const noexcept
- void **SetStartColor** (const Vec4f &startColor) noexcept
- const Vec4f & **GetEndColor** () const noexcept
- void **SetEndColor** (const Vec4f &endColor) noexcept
- bool **IsActive** () const noexcept
- void **SetActive** (const bool active) noexcept
- const Vec3f & **GetVelocity** () const noexcept
- void **SetVelocity** (const Vec3f &velocity) noexcept
- float **GetVelocityRange** () const noexcept
- void **SetVelocityRange** (const float velocityRange) noexcept
- const Vec3f & **GetExternalForce** () const noexcept
- void **SetExternalForce** (const Vec3f &externalFroce) noexcept
- [Mesh](#) * **GetMesh** () const noexcept
- void **SetMesh** ([Mesh](#) *mesh) noexcept
- [Texture](#) * **GetTexture** () const noexcept
- void **SetTexture** ([Texture](#) *texture) noexcept
- RenderStateType **GetBlendStateType** () const noexcept
- void **SetBlendStateType** (RenderStateType blendStateType) noexcept
- void **Update** (const float dt, const long time) noexcept override
- Updates the [Component](#) on each frame.*
- void **Setup** () noexcept override
- Performs setup actions after the [BehaviourComponent](#)'s creation.*
- void **Teardown** () noexcept override
- Performs actions before the [BehaviourComponent](#) is destroyed.*

5.40.1 Member Function Documentation

5.40.1.1 Update()

```
void Blade::EmitterComponent::Update (
    const float dt,
    const long time ) [override], [virtual], [noexcept]
```

Updates the [Component](#) on each frame.

Parameters

<i>dt</i>	The time elapsed from the previous frame of the Application .
<i>time</i>	The elapsed time since the start of the Application .

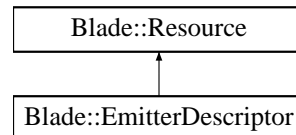
Implements [Blade::BehaviourComponent](#).

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

- include/emitter_component.h
- src/emitter_component.cpp

5.41 Blade::EmitterDescriptor Struct Reference

Inheritance diagram for Blade::EmitterDescriptor:



Public Member Functions

- bool **Load** (const std::wstring &file_name) noexcept override

Public Attributes

- Vec3f **velocity**
- Vec3f **externalForce**
- float **spawnRate**
- float **lifespan**
- float **maxParticles**
- float **spawnRadius**
- float **particleSize**
- Vec4f **startColor**
- Vec4f **endColor**
- [Texture](#) * **texture**
- RenderStateType **blendStateType**
- float **particlesToSpawn** { 0 }
- float **velocityRange**
- bool **active**

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

- include/emitter_component.h
- src/emitter_component.cpp

5.42 Blade::EngineContext Class Reference

Public Member Functions

- **EngineContext** (const [EngineContext](#) &context)=delete
- [EngineContext](#) & **operator=** (const [EngineContext](#) &context)=delete

Static Public Member Functions

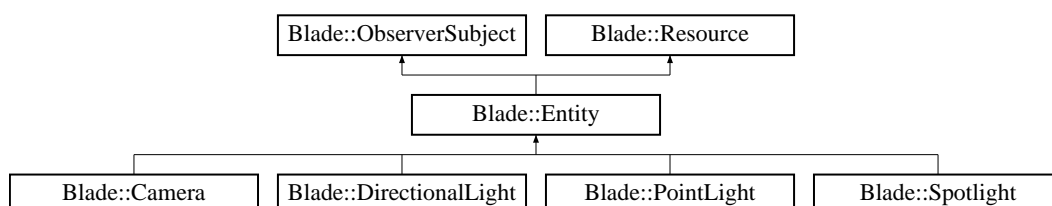
- static bool **Initialize** ()
- static [ThreadPool](#) & **GetThreadPool** () noexcept
- static [RenderSystem](#) & **GetRenderSystem** () noexcept
- static [CameraSystem](#) & **GetCameraSystem** () noexcept
- static [LightSystem](#) & **GetLightSystem** () noexcept
- static [SimulationSystem](#) & **GetSimulationSystem** () noexcept
- static [BehaviourSystem](#) & **GetBehaviourSystem** () noexcept
- static [NetworkManager](#) & **GetNetworkManager** () noexcept
- static [RenderStateManager](#) & **GetRenderStateManager** () noexcept
- static [ResourceManager](#) & **GetResourceManager** () noexcept
- static [SceneManager](#) & **GetSceneManager** () noexcept
- static [ShaderProgramManager](#) & **GetShaderProgramManager** () noexcept
- static [InputManager](#) & **GetInputManager** () noexcept
- static [ParticleSystem](#) & **GetParticleSystem** () noexcept
- static void **RegisterApplication** ([Application](#) *application) noexcept
- static [Application](#) & **GetApplication** () noexcept
- static [AudioManager](#) & **GetAudioManager** () noexcept

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

- include/engine_context.h
- src/engine_context.cpp

5.43 Blade::Entity Class Reference

Inheritance diagram for Blade::Entity:



Public Member Functions

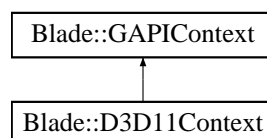
- **Entity** (const std::string &name)
- **Entity** (const [Entity](#) &other)
- [Entity](#) & **operator=** (const [Entity](#) &other)
- const std::string & **GetName** () const noexcept
- const Vec3f & **GetLocalPosition** () const noexcept
- Vec3f **GetWorldPosition** () noexcept
- void **SetPosition** (const Vec3f &position) noexcept
- const Quatf & **GetOrientation** () const noexcept
- void **SetOrientation** (const Quatf &orientation) noexcept
- void **SetOrientation** (const Vec3f &axis, float angle) noexcept
- const Vec3f & **GetScale** () const noexcept
- void **SetScale** (const Vec3f &scale) noexcept
- [Entity](#) * **GetParent** () const noexcept
- void **SetParent** ([Entity](#) *entity) noexcept
- const std::vector< [Entity](#) * > & **GetChildren** () const noexcept
- [Entity](#) * **GetChild** (int index) const noexcept
- [Entity](#) * **GetEntityFromHierarchy** (const std::string &name) noexcept
- void **AddChild** ([Entity](#) *entity) noexcept
- size_t **GetChildrenCount** () const noexcept
- const Mat4f & **GetXform** () const noexcept
- void **SetXform** (const Mat4f &xform) noexcept
- void **CalculateXform** () noexcept
- [Component](#) * **GetComponent** (const std::string &type) const noexcept
- void **Entity::RemoveComponent** (const int id) noexcept
- std::vector< [Component](#) * > **GetComponents** (const std::string &type) const noexcept
- void **AddComponent** ([Component](#) *component) noexcept
- bool **IsAlive** () const noexcept
- void **SetAlive** (bool state) noexcept
- virtual void **Update** (float dt, long time=0) noexcept
- bool **Load** (const std::wstring &fileName) noexcept override

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

- include/entity.h
- src/entity.cpp

5.44 Blade::GAPIContext Class Reference

Inheritance diagram for Blade::GAPIContext:



Public Member Functions

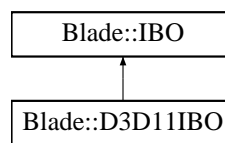
- virtual bool **Create** (LUID *luid)=0

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

- include/GAPI_context.h

5.45 Blade::IBO Class Reference

Inheritance diagram for Blade::IBO:



Public Member Functions

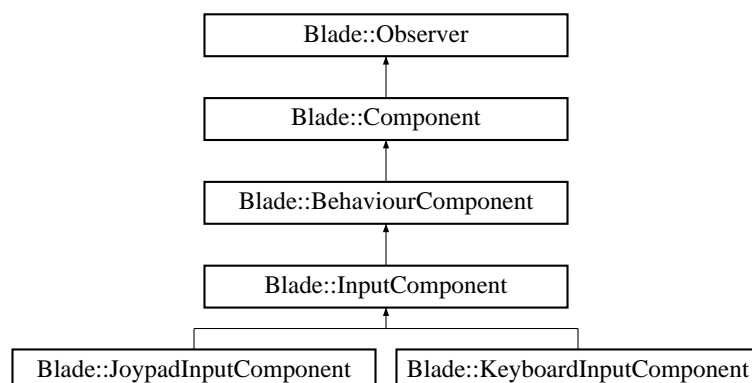
- void **SetIndexCount** (unsigned int idxCount) noexcept
- unsigned int **GetIndexCount** () const noexcept
- virtual bool **Create** (const std::vector< unsigned int > &indices) noexcept=0
- virtual void **Bind** () const noexcept=0
- virtual void **Draw** () const noexcept=0

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

- include/IBO.h
- src/IBO.cpp

5.46 Blade::InputComponent Class Reference

Inheritance diagram for Blade::InputComponent:



Public Member Functions

- **InputComponent** (const std::string &type, [Entity](#) *parent, bool online=false)
- **InputComponent** (const [InputComponent](#) &other)=delete
- **InputComponent & operator=** (const [InputComponent](#) &other)=delete
- virtual void **Update** (const float dt, const long time=0) noexcept=0
Updates the [Component](#) on each frame.
- virtual void **Setup** () noexcept=0
Performs setup actions after the [BehaviourComponent](#)'s creation.
- virtual void **Teardown** () noexcept=0
Performs actions before the [BehaviourComponent](#) is destroyed.

Protected Attributes

- bool **m_Online**

5.46.1 Member Function Documentation

5.46.1.1 Update()

```
virtual void Blade::InputComponent::Update (
    const float dt,
    const long time = 0 ) [pure virtual], [noexcept]
```

Updates the [Component](#) on each frame.

Parameters

<i>dt</i>	The time elapsed from the previous frame of the Application .
<i>time</i>	The elapsed time since the start of the Application .

Implements [Blade::BehaviourComponent](#).

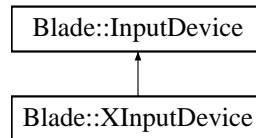
Implemented in [Blade::JoypadInputComponent](#), and [Blade::KeyboardInputComponent](#).

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

- include/input_component.h
- src/input_component.cpp

5.47 Blade::InputDevice Class Reference

Inheritance diagram for [Blade::InputDevice](#):



Public Member Functions

- **InputDevice** (const [InputDevice](#) &)=delete
- [InputDevice](#) & **operator=** (const [InputDevice](#) &rhs)=delete
- **InputDevice** ([InputDevice](#) &&src)=delete
- [InputDevice](#) & **operator=** ([InputDevice](#) &&rhs)=delete
- **InputDevice** (int device_id, DeviceType devType)
- const [InputState](#) & **GetInputState** () const
- int **GetDeviceID** () const
- virtual void **Update** (float fDeltaTime)=0
- virtual bool **SetVibration** (float leftMotor, float rightMotor) const =0
- void **SetDeadzone** (AnalogDeadzone flag, float value)
- float **GetDeadzone** (AnalogDeadzone flag) const
- virtual bool **IsConnected** () const =0
- DeviceType **GetDeviceType** () const
- const [InputState](#) & **GetCurrentState** () const
- const [InputState](#) & **GetPreviousState** () const

Protected Member Functions

- void **SetDeviceID** (int id)
- void **SetDeviceType** (DeviceType devType)
- void **SetInputState** (const [InputState](#) &state)
- virtual bool **Initialize** ()=0

Static Protected Member Functions

- static void **FilterStateData** (const [InputState](#) &stateIn, [InputState](#) &stateOut)

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

- include/input_device.h
- src/input_device.cpp

5.48 Blade::InputManager Class Reference

Public Member Functions

- Vec2f [GetAnalogStickVector](#) (JoypadNumber joypadNumber, InputSensor sensor)
Returns a vector indicating the position of the analog stick requested.
- bool [QueryKeyState](#) (VirtualKey key) const noexcept
Query the keyboard device for the state of a key.
- bool [QueryAllKeyStates](#) (std::map< VirtualKey, bool > &destMap) const noexcept
Query the Keyboard device for the state of ALL keys associated to the device.
- Vec2f [QueryMouseMovement](#) ()
Query the Keyboard device for the state of ALL keys associated to the device.
- Vec2f [QueryMouseMovementNormalized](#) ()
Query the Keyboard device for the state of ALL keys associated to the device.
- Vec2i [QueryMousePosition](#) () const noexcept
Query the Keyboard device for the state of ALL keys associated to the device.
- bool [QueryMouseButtonState](#) (MouseButton button)
Query the state of the mouse buttons (providing an enum per button)
- bool [QueryDeviceState](#) (JoypadNumber joypadNumber, InputSensor sensor)
Query the state of a sensor on an active pad linked.
- bool [QueryDeviceAllStates](#) (JoypadNumber joypadNumber, std::map< InputSensor, bool > &map)
Query the input states of sensors on an active device linked to player, return in supplied map.
- void [Update](#) (float deltaTime)
Update the states of managed input devices, and re-enumerate input devices.
- bool [Initialize](#) () noexcept
Initialize the input manager.
- int [EnumerateDevices](#) () noexcept
Counts and store the number of connected devices to the machine.
- DeviceType [DevicePoolQueryType](#) (int deviceId)
Query a device pool for its type.
- bool [PooledDeviceExists](#) (int deviceId)
Search the device pool for a device with id equal to deviceId.
- bool [ActiveDeviceExists](#) (int deviceId)
Search the active device map for a device with id equal to deviceId.
- bool [AssignDeviceToPlayer](#) (JoypadNumber joypadNumber, int deviceNumber)
Assigns a player to an input device.
- bool [UnassignDevice](#) (JoypadNumber joypadNumber)
Unassigns an input device from a player (by player ID)
- InputDevice * [GetActiveDevice](#) (JoypadNumber joypadNumber)
Returns an active (not in the pool) assigned input device, searched by player.
- void [UpdateMousePos](#) (Vec2i mousepos)
- void [SetMouseButtonState](#) (MouseButton state, bool value)

5.48.1 Member Function Documentation

5.48.1.1 ActiveDeviceExists()

```
bool Blade::InputManager::ActiveDeviceExists (
    int deviceId )
```

Search the active device map for a device with id equal to deviceId.

Parameters

<i>device↔ Id</i>	The device number you need to find in the map.
-----------------------	--

Returns

True if the device is found, otherwise false

5.48.1.2 AssignDeviceToPlayer()

```
bool Blade::InputManager::AssignDeviceToPlayer (
    JoypadNumber joypadNumber,
    int deviceNumber )
```

Assigns a player to an input device.

Parameters

<i>joypadNumber</i>	The joy pad number.
<i>deviceNumber</i>	The number of the device from the pool.

Returns

True if successful, false otherwise

5.48.1.3 DevicePoolQueryType()

```
DeviceType Blade::InputManager::DevicePoolQueryType (
    int deviceId )
```

Query a device pool for its type.

Parameters

<i>device↔ Id</i>	The device number.
-----------------------	--------------------

Returns

DeviceType enum of the device in the pool denoted by devIndex

Remarks

If the device is not found, or an error has occurred, DEVTYPE_ERROR is returned

5.48.1.4 EnumerateDevices()

```
int Blade::InputManager::EnumerateDevices ( ) [noexcept]
```

Counts and store the number of connected devices to the machine.

Returns

An integer representing the number of connected input devices

5.48.1.5 GetActiveDevice()

```
InputDevice * Blade::InputManager::GetActiveDevice (
    JoypadNumber joypadNumber )
```

Returns an active (not in the pool) assigned input device, searched by player.

Parameters

<i>joypadNumber</i>	The number of the joy pad number.
---------------------	-----------------------------------

Returns

Active input device for player id, nullptr otherwise

5.48.1.6 GetAnalogStickVector()

```
Vec2f Blade::InputManager::GetAnalogStickVector (
    JoypadNumber joypadNumber,
    InputSensor sensor )
```

Returns a vector indicating the position of the analog stick requested.

Parameters

<i>joypadNumber</i>	The joypad number.
<i>sensor</i>	The stick that need to be checked

Returns

The vector that represent the stick position normalized outside dead zones.

5.48.1.7 Initialize()

```
bool Blade::InputManager::Initialize ( ) [noexcept]
```

Initialize the input manager.

Returns

True if the initialization is successful, false otherwise

5.48.1.8 PooledDeviceExists()

```
bool Blade::InputManager::PooledDeviceExists (
    int deviceId )
```

Search the device pool for a device with id equal to deviceId.

Parameters

<i>deviceId</i>	The device number.
-----------------	--------------------

Returns

True if the device is found, otherwise false

5.48.1.9 QueryAllKeyStates()

```
bool Blade::InputManager::QueryAllKeyStates (
    std::map< VirtualKey, bool > & destMap ) const [noexcept]
```

Query the Keyboard device for the state of ALL keys associated to the device.

Parameters

<i>map</i>	The reference to a map of all keyboard states.
------------	--

Returns

True if successful, false otherwise

5.48.1.10 QueryDeviceAllStates()

```
bool Blade::InputManager::QueryDeviceAllStates (
    JoypadNumber joypadNumber,
    std::map< InputSensor, bool > & map )
```

Query the input states of sensors on an active device linked to player, return in supplied map.

Parameters

<i>joypadNumber</i>	The queried joypad number.
<i>map</i>	A reference to a map of the state of the joypad.

Returns

TRUE if the JoypadNumber is active, false otherwise

5.48.1.11 QueryDeviceState()

```
bool Blade::InputManager::QueryDeviceState (
    JoypadNumber joypadNumber,
    InputSensor sensor )
```

Query the state of a sensor on an active pad linked.

Parameters

<i>joypadNumber</i>	The queried joypad number
<i>sensor</i>	The sensor to check

Returns

TRUE if the the sensor is active (button pressed, stick out of the deadzone), FALSE otherwise.

5.48.1.12 QueryKeyState()

```
bool Blade::InputManager::QueryKeyState (
    VirtualKey key ) const [noexcept]
```

Query the keyboard device for the state of a key.

Parameters

<i>key</i>	A keyboard key.
------------	-----------------

Returns

True if the key is a PRESSED state (down), false otherwise

5.48.1.13 QueryMouseButtonState()

```
bool Blade::InputManager::QueryMouseButtonState (
    MouseButton button )
```

Query the state of the mouse buttons (providing an enum per button)

Parameters

<i>button</i>	The mouse button
---------------	------------------

Returns

True if pressed, false otherwise

5.48.1.14 QueryMouseMovement()

```
Vec2f Blade::InputManager::QueryMouseMovement ( )
```

Query the Keyboard device for the state of ALL keys associated to the device.

Returns

True if successful, false otherwise

5.48.1.15 QueryMouseMovementNormalized()

```
Vec2f Blade::InputManager::QueryMouseMovementNormalized ( )
```

Query the Keyboard device for the state of ALL keys associated to the device.

Returns

True if successful, false otherwise

5.48.1.16 QueryMousePosition()

```
Vec2i Blade::InputManager::QueryMousePosition ( ) const [noexcept]
```

Query the Keyboard device for the state of ALL keys associated to the device.

Returns

True if successful, false otherwise

5.48.1.17 UnassignDevice()

```
bool Blade::InputManager::UnassignDevice (
    JoypadNumber joypadNumber )
```

Unassigns an input device from a player (by player ID)

Parameters

<i>joypadNumber</i>	The number of the joy pad player number.
---------------------	--

Returns

Destroy the association between player and device, and mark device as inactive

5.48.1.18 Update()

```
void Blade::InputManager::Update (
    float deltaTime )
```

Update the states of managed input devices, and re-enumerate input devices.

Parameters

<i>deltaTime</i>	The delta time.
------------------	-----------------

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

- include/input_manager.h
- src/input_manager.cpp

5.49 Blade::InputState Struct Reference

[InputState](#) describes the current state of a device.

```
#include <input_state.h>
```

Public Member Functions

- **InputState** (const [InputState](#) &src) noexcept=default
- [InputState](#) & **operator=** (const [InputState](#) &rhs) noexcept=default
- **InputState** ([InputState](#) &&src) noexcept=default
- [InputState](#) & **operator=** ([InputState](#) &&rhs) noexcept=default

Public Attributes

- int **digitalButtonData** { 0 }
- [ThumbStick](#) **stickLeft** { 0 }
- [ThumbStick](#) **stickRight** { 0 }
- float **triggerLeft** { 0.0f }
- float **triggerRight** { 0.0f }

5.49.1 Detailed Description

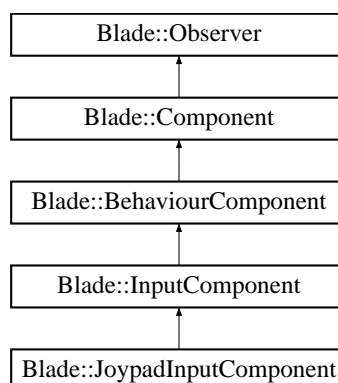
[InputState](#) describes the current state of a device.

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

- include/input_state.h
- src/input_state.cpp

5.50 Blade::JoypadInputComponent Class Reference

Inheritance diagram for Blade::JoypadInputComponent:



Public Member Functions

- **JoypadInputComponent** ([Entity](#) *parent, JoypadNumber joypad_number, bool online)
- **JoypadInputComponent** (const [JoypadInputComponent](#) &other)=delete
- **JoypadInputComponent & operator=** (const [JoypadInputComponent](#) &other)=delete
- virtual void **Update** (const float dt, const long time=0) noexcept=0
Updates the [Component](#) on each frame.
- virtual void **Setup** () noexcept=0
Performs setup actions after the [BehaviourComponent](#)'s creation.
- virtual void **Teardown** () noexcept=0
Performs actions before the [BehaviourComponent](#) is destroyed.
- bool **LoadConfiguration** (const std::vector< InputSensor > &control, const std::vector< std::shared_ptr< [Command](#) >> &commands) noexcept
- bool **LoadConfiguration** (const JoypadCommandMap &map)
- const JoypadCommandMap & **GetCommandMap** () const noexcept

Public Attributes

- JoypadNumber **m_JoypadNum**

Protected Attributes

- JoypadCommandMap **m_JoypadCommandMap**

5.50.1 Member Function Documentation

5.50.1.1 Update()

```
virtual void Blade::JoypadInputComponent::Update (
    const float dt,
    const long time = 0 ) [pure virtual], [noexcept]
```

Updates the [Component](#) on each frame.

Parameters

<i>dt</i>	The time elapsed from the previous frame of the Application .
<i>time</i>	The elapsed time since the start of the Application .

Implements [Blade::InputComponent](#).

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

- include/joypad_input_component.h
- src/joypad_input_component.cpp

5.51 Blade::KeyboardInput Class Reference

Keyboard abstraction of the engine.

```
#include <keyboard_input.h>
```

Static Public Member Functions

- static bool [QueryKeyState](#) (VirtualKey value) noexcept
Query the state of a virtual key.
- static bool [QueryAllKeyStates](#) (std::map< VirtualKey, bool > &destMap) noexcept
Query all virtual key states for attached keyboard.

5.51.1 Detailed Description

Keyboard abstraction of the engine.

5.51.2 Member Function Documentation

5.51.2.1 QueryAllKeyStates()

```
bool Blade::KeyboardInput::QueryAllKeyStates (
    std::map< VirtualKey, bool > & destMap ) [static], [noexcept]
```

Query all virtual key states for attached keyboard.

Parameters

<i>destMap</i>	A reference to a map storing all the keyboard keys.
----------------	---

Returns

True if successful, false otherwise

5.51.2.2 QueryKeyState()

```
bool Blade::KeyboardInput::QueryKeyState (
    VirtualKey value ) [static], [noexcept]
```

Query the state of a virtual key.

Parameters

<i>value</i>	The key that to check.
--------------	------------------------

Returns

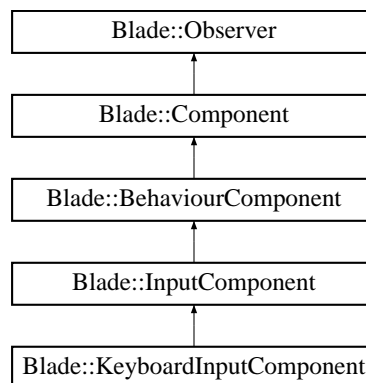
True if the key being queried is PRESSED (down), false otherwise

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

- include/keyboard_input.h
- src/keyboard_input.cpp

5.52 Blade::KeyboardInputComponent Class Reference

Inheritance diagram for Blade::KeyboardInputComponent:



Public Types

- using **KeyboardCommandMap** = std::map< VirtualKey, std::shared_ptr< [Command](#) > >

Public Member Functions

- **KeyboardInputComponent** ([Entity](#) *parent, bool online)
- **KeyboardInputComponent** (const [KeyboardInputComponent](#) &other)=delete
- [KeyboardInputComponent](#) & **operator=** (const [KeyboardInputComponent](#) &other)=delete
- virtual void **Update** (const float dt, const long time=0) noexcept=0
Updates the [Component](#) on each frame.
- virtual void **Setup** () noexcept=0
Performs setup actions after the [BehaviourComponent](#)'s creation.
- virtual void **Teardown** () noexcept=0
Performs actions before the [BehaviourComponent](#) is destroyed.
- bool **LoadConfiguration** (std::vector< VirtualKey > &keys, const std::vector< std::shared_ptr< [Command](#) >> &commands) noexcept
- bool **LoadConfiguration** (const KeyboardCommandMap &map)
- const KeyboardCommandMap & **GetKeyboardCommandMap** () const noexcept

Protected Attributes

- KeyboardCommandMap **m_KeyboardCommandMap**

5.52.1 Member Function Documentation

5.52.1.1 Update()

```
virtual void Blade::KeyboardInputComponent::Update (
    const float dt,
    const long time = 0 ) [pure virtual], [noexcept]
```

Updates the [Component](#) on each frame.

Parameters

<i>dt</i>	The time elapsed from the previous frame of the Application .
<i>time</i>	The elapsed time since the start of the Application .

Implements [Blade::InputComponent](#).

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

- include/keyboard_input_component.h
- src/keyboard_input_component.cpp

5.53 Blade::Keyframe< T > Struct Template Reference

Public Member Functions

- **Keyframe** (const T &value, long time)
- bool **operator**< (const [Keyframe](#)< T > &other) const noexcept

Public Attributes

- T **value**
- long **time** { 0 }

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

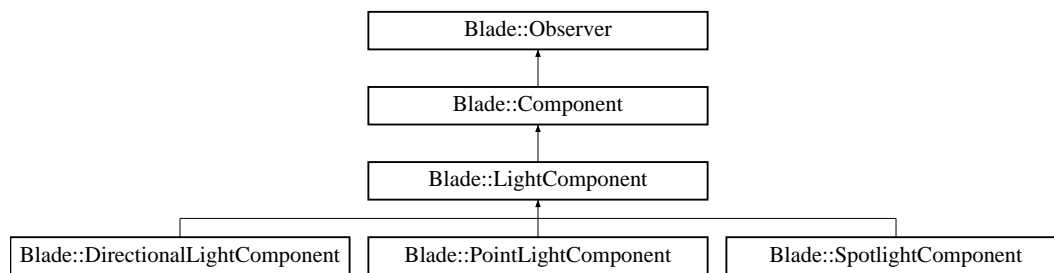
- include/animation.h

5.54 Blade::LightComponent Class Reference

Abstract class that describes a [LightComponent](#). Provides the base functionality of a [LightComponent](#). It contains the component's type and an index to the entry of the correct light description cache in the [LightSystem](#). Managed by the [LightSystem](#).

```
#include <light_component.h>
```

Inheritance diagram for Blade::LightComponent:



Public Member Functions

- **LightComponent** ([LightType](#) lightType, [Entity](#) *parent)
- [LightType](#) **GetLightType** () const noexcept
- int **GetLightDescCacheIndex** () const noexcept
- void **SetLightDescCacheIndex** (int index) noexcept

5.54.1 Detailed Description

Abstract class that describes a [LightComponent](#). Provides the base functionality of a [LightComponent](#). It contains the component's type and an index to the entry of the correct light description cache in the [LightSystem](#). Managed by the [LightSystem](#).

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

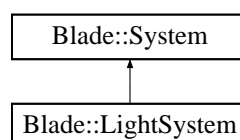
- include/light_component.h

5.55 Blade::LightSystem Class Reference

A [System](#) responsible for managing [LightComponents](#). This system updates the positions of all the lights in the scene every frame. It is also responsible for caching the light descriptions of each light upon registration of a [LightComponent](#).

```
#include <light_system.h>
```

Inheritance diagram for Blade::LightSystem:



Public Member Functions

- void [RegisterComponent](#) ([LightComponent](#) *lightComponent) noexcept
Registers a [LightComponent](#) to the system.
- void [UnregisterComponent](#) (int id) noexcept
Unregisters a [LightComponent](#) from the system.
- std::vector< [PointLightDesc](#) > [GetPointLightDescriptions](#) () const noexcept
Provides a vector of the cached point light description structs.
- std::vector< [DirectionalLightDesc](#) > [GetDirectionalLightDescriptions](#) () const noexcept
Provides a vector of the cached directional light description structs.
- std::vector< [SpotlightDesc](#) > [GetSpotlightDescriptions](#) () const noexcept
Provides a vector of the cached spotlight description structs.
- bool [Initialize](#) () noexcept override
Pure virtual method implemented by the engine's systems to perform their initialization.
- void [Process](#) (float deltaTime=.0f, long time=0) noexcept override
Processes the [LightComponents](#).

5.55.1 Detailed Description

A [System](#) responsible for managing [LightComponents](#). This system updates the positions of all the lights in the scene every frame. It is also responsible for caching the light descriptions of each light upon registration of a [LightComponent](#).

5.55.2 Member Function Documentation

5.55.2.1 [GetDirectionalLightDescriptions\(\)](#)

```
std::vector< DirectionalLightDesc > Blade::LightSystem::GetDirectionalLightDescriptions ( )
const [noexcept]
```

Provides a vector of the cached directional light description structs.

Returns

A vector of the cached directional light description structs.

5.55.2.2 [GetPointLightDescriptions\(\)](#)

```
std::vector< PointLightDesc > Blade::LightSystem::GetPointLightDescriptions ( ) const [noexcept]
```

Provides a vector of the cached point light description structs.

Returns

A vector of the cached point light description structs.

5.55.2.3 GetSpotlightDescriptions()

```
std::vector< SpotlightDesc > Blade::LightSystem::GetSpotlightDescriptions ( ) const [noexcept]
```

Provides a vector of the cached spotlight description structs.

Returns

A vector of the cached spotlight description structs.

5.55.2.4 Initialize()

```
bool Blade::LightSystem::Initialize ( ) [override], [virtual], [noexcept]
```

Pure virtual method implemented by the engine's systems to perform their initialization.

Returns

TRUE if initialization is successful, FALSE otherwise.

Implements [Blade::System](#).

5.55.2.5 Process()

```
void Blade::LightSystem::Process (
    float deltaTime = .0f,
    long time = 0 ) [override], [virtual], [noexcept]
```

Processes the LightComponents.

This method iterates through all the active LightComponents. Based on their type it updates the position/direction data members of each [LightComponent](#)'s light description contained in the matching cache.

Parameters

<i>deltaTime</i>	The time elapsed from the previous frame of the application.
------------------	--

Implements [Blade::System](#).

5.55.2.6 RegisterComponent()

```
void Blade::LightSystem::RegisterComponent (
    LightComponent * lightComponent ) [noexcept]
```

Registers a [LightComponent](#) to the system.

This method registers a [LightComponent](#) to the system. It maps the [LightComponent](#) with a name and based on it's type it puts the light description contained in the [LightComponent](#) to the correct light description cache.

Parameters

<i>lightComponent</i>	The LightComponent to be registered.
-----------------------	--

5.55.2.7 UnregisterComponent()

```
void Blade::LightSystem::UnregisterComponent (
    int id ) [noexcept]
```

Unregisters a [LightComponent](#) from the system.

This method unregisters a [LightComponent](#) and based on it's type removes it's light description from the correct light description cache.

Parameters

<i>id</i>	The id of the LightComponent to unregister.
-----------	---

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

- include/light_system.h
- src/light_system.cpp

5.56 Blade::ManifoldEntry Struct Reference

Public Attributes

- const [Collider](#) * **collider1**
- const [Collider](#) * **collider2**
- Vec3f **contactNormal**
- float **t** { 0.0f }
- float **penetration** { 0.0f }

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

- include/contact_manifold.h

5.57 Blade::Material Struct Reference

Public Member Functions

- **Material** (const [Material](#) &other)=default
- **Material** & **operator=** (const [Material](#) &other)=default

Public Attributes

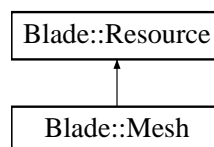
- `std::array< Texture *, SUPPORTED_TEX_COUNT > textures`
- `Vec4f diffuse`
- `Vec4f specular`
- `Mat4f textureMatrix`
- `std::string shaderProgramName { "sdrprog_default" }`
- `RenderStateType blendState`

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

- `include/material.h`
- `src/material.cpp`

5.58 Blade::Mesh Class Reference

Inheritance diagram for Blade::Mesh:



Public Member Functions

- `Mesh (const Mesh &other)=default`
- `Mesh & operator= (const Mesh &other)=default`
- `VBO * GetVbo () const noexcept`
- `IBO * GetIbo () const noexcept`
- `size_t GetVertexCount () const noexcept`
- `size_t GetIndexCount () const noexcept`
- `void SetName (const std::string &name) noexcept`
- `const std::string & GetName () const noexcept`
- `void InitiazeBufferObjects (PrimitiveTopology primitiveTopology=PrimitiveTopology::TRIANGLE_LIST) const noexcept`
- `void SetVertexData (const Vertex *vertices, int vertexCount) noexcept`
- `Vertex * GetVertexData () const noexcept`
- `void AddVertex (const Vertex &vertex) noexcept`
- `void SetIndexData (const unsigned int *indices, int indexCount) noexcept`
- `unsigned int * GetIndexData () const noexcept`
- `void AddIndex (unsigned int index) noexcept`
- `bool Load (const std::wstring &fileName) noexcept override`
- `void GenerateIndices (VertexWinding winding) noexcept`

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

- `include/mesh.h`
- `src/mesh.cpp`

5.59 Blade::Message< T > Class Template Reference

Public Member Functions

- **Message** (T &&type)
- const T & **GetType** () const noexcept

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

- include/message.h

5.60 Blade::NCF Class Reference

Public Member Functions

- void **SetSource** (const char *file)
- const char * **GetSource** () const
- void **Purge** ()
- int **Parse** ()
- int **Dump** (const char *file, int create=1) const
- bool **QueryProperty** (const char *name) const
- bool **QueryGroup** (const char *name) const
- unsigned int **CountProperties** () const
- unsigned int **CountGroups** () const
- void **SetProperty** (const char *name, const char *value)
- const char * **GetPropertyByName** (const char *name) const
- const char * **GetPropertyByIndex** (unsigned int index) const
- const char * **GetPropertyNameByIndex** (unsigned int index) const
- [NCF](#) * **GetGroupByName** (const char *name) const
- [NCF](#) * **GetGroupByIndex** (unsigned int index) const
- const char * **GetName** () const
- [NCF](#) (const [NCF](#) &)=delete
- [NCF](#) & **operator=** (const [NCF](#) &)=delete

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

- include/ncf.h
- src/ncf.cpp

5.61 Blade::NetworkManager Class Reference

Public Member Functions

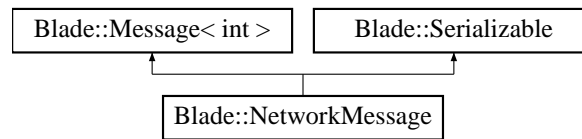
- bool **Initialize** () noexcept
- void **Listen** (const unsigned short port) noexcept
- void **Connect** (const std::string &host, const unsigned short port) noexcept
- void **QueueMessage** (const std::shared_ptr< [NetworkMessage](#) > &message) noexcept
- size_t **GetConnectionCount** () noexcept
- void **SetOnNewPacketCallback** (const OnNewPacketCallback &callback) noexcept
- void **SetOnNewClientCallback** (const OnNewClientCallback &callback) noexcept
- void **SetOnClientDisconnectCallback** (const OnClientDisconnectCallback &callback) noexcept

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

- include/network_manager.h
- src/network_manager.cpp

5.62 Blade::NetMessage Class Reference

Inheritance diagram for Blade::NetMessage:



Public Member Functions

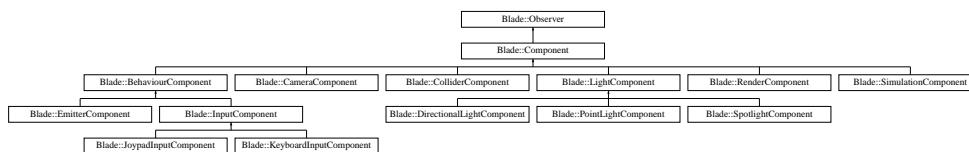
- **NetMessage** (int &&type, long recipientId)

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

- include/network_message.h

5.63 Blade::Observer Class Reference

Inheritance diagram for Blade::Observer:



Public Member Functions

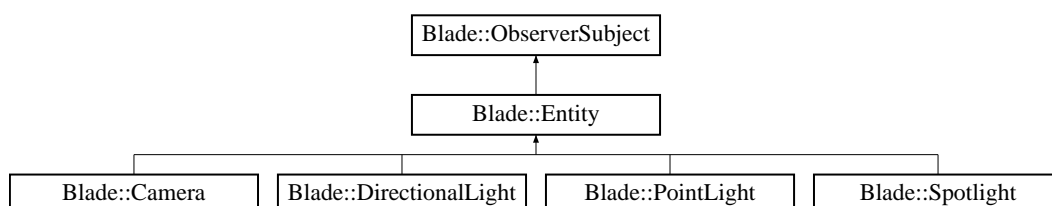
- virtual void **OnMessage** (const [MessageContainer](#)< std::string > &msg)=0

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

- include/observer.h
- src/observer.cpp

5.64 Blade::ObserverSubject Class Reference

Inheritance diagram for Blade::ObserverSubject:



Public Member Functions

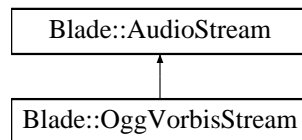
- void **RegisterObserver** (const std::string &msg, [Observer](#) *o) noexcept
- void **UnregisterObserver** (const std::string &msg, [Observer](#) *o) noexcept
- void **BroadcastMessage** (const [MessageContainer](#)< std::string > &msg) const noexcept

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

- include/observer_subject.h
- src/observer_subject.cpp

5.65 Blade::OggVorbisStream Class Reference

Inheritance diagram for Blade::OggVorbisStream:



Public Member Functions

- bool **Open** (const char *fname) noexcept
- void **Close** () noexcept
- void **Play** (AudioPlaymode mode) noexcept override
- void **Rewind** () noexcept override

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

- include/ovstream.h
- src/ovstream.cpp

5.66 Blade::Particle Struct Reference

Public Attributes

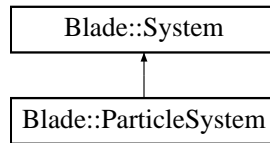
- Vec3f **position**
- Vec4f **color**
- float **size**
- Vec3f **velocity**
- float **life**
- bool **active**
- double **spawn_time**

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

- include/emitter_component.h

5.67 Blade::ParticleSystem Class Reference

Inheritance diagram for Blade::ParticleSystem:



Public Member Functions

- void **RegisterComponent** ([EmitterComponent](#) *emitterComponent) noexcept
- void **UnregisterComponent** (const int id) noexcept
- bool [Initialize](#) () noexcept override

Pure virtual method implemented by the engine's systems to perform their initialization.
- std::vector< [EmitterComponent](#) * > & **GetEmitterComponents** () noexcept
- void [Process](#) (float deltaTime=.0f, long time=0) noexcept override

Pure virtual method implemented by the engine's systems to process the registered components.

5.67.1 Member Function Documentation

5.67.1.1 Initialize()

```
bool Blade::ParticleSystem::Initialize ( ) [override], [virtual], [noexcept]
```

Pure virtual method implemented by the engine's systems to perform their initialization.

Returns

TRUE if initialization is successfull, FALSE otherwise.

Implements [Blade::System](#).

5.67.1.2 Process()

```
void Blade::ParticleSystem::Process (
    float deltaTime = .0f,
    long time = 0 ) [override], [virtual], [noexcept]
```

Pure virtual method implemented by the engine's systems to process the registered components.

Parameters

<i>deltaTime</i>	The time elapsed from the previous frame of the application.
------------------	--

Implements [Blade::System](#).

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

- include/particle_system.h
- src/particle_system.cpp

5.68 Blade::Pipeline< T, Tdata > Class Template Reference

Abstract class that describes a pipeline that processes the specified object data type.

```
#include <pipeline.h>
```

Public Member Functions

- void [AddStage](#) ([PipelineStage](#)< T, Tdata > *stage)
Adds a [PipelineStage](#) to the [Pipeline](#).
- void [Execute](#) (const std::vector< T > &data)
Processes the objects provided by passing then through each [PipelineStage](#).

5.68.1 Detailed Description

```
template<typename T, typename Tdata>
class Blade::Pipeline< T, Tdata >
```

Abstract class that describes a pipeline that processes the specified object data type.

Template Parameters

<i>T</i>	The type of data that the Pipeline 's PipelineStages will process.
<i>Tdata</i>	The type of data that the PipelineStages will return after executed.

5.68.2 Member Function Documentation

5.68.2.1 AddStage()

```
template<typename T , typename Tdata >
void Blade::Pipeline< T, Tdata >::AddStage (
    PipelineStage< T, Tdata > * stage ) [inline]
```

Adds a [PipelineStage](#) to the [Pipeline](#).

Parameters

<code>stage</code>	The PipelineStage to be added to the Pipeline .
--------------------	---

5.68.2.2 Execute()

```
template<typename T , typename Tdata >
void Blade::Pipeline< T, Tdata >::Execute (
    const std::vector< T > & data ) [inline]
```

Processes the objects provided by passing them through each [PipelineStage](#).

Parameters

<code>data</code>	The objects to be processed by the Pipeline 's stages.
-------------------	--

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

- include/pipeline.h

5.69 Blade::PipelineData< T > Class Template Reference

An abstract data container for the data returned by a [PipelineStage](#).

```
#include <pipeline_stage.h>
```

Public Member Functions

- [PipelineData](#) (T data)
[PipelineData](#) constructor.
- T [Get](#) () const noexcept
Returns the data contained in the [PipelineData](#) container.

5.69.1 Detailed Description

```
template<typename T>
class Blade::PipelineData< T >
```

An abstract data container for the data returned by a [PipelineStage](#).

Template Parameters

<i>T</i>	The type of data the container will hold.
----------	---

5.69.2 Constructor & Destructor Documentation

5.69.2.1 PipelineData()

```
template<typename T>
Blade::PipelineData< T >::PipelineData (
    T data ) [inline], [explicit]
```

[PipelineData](#) constructor.

Parameters

<i>data</i>	The data to store in the container.
-------------	-------------------------------------

5.69.3 Member Function Documentation

5.69.3.1 Get()

```
template<typename T>
T Blade::PipelineData< T >::Get ( ) const [inline], [noexcept]
```

Returns the data contained in the [PipelineData](#) container.

Returns

The data contained in the container.

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

- include/pipeline_stage.h

5.70 Blade::PipelineStage< T, Tdata > Class Template Reference

This class describes an abstract stage of a pipeline that processes the specified type of data and returns the specified type of data.

```
#include <pipeline_stage.h>
```

Public Member Functions

- [PipelineStage](#) (const std::string &name)
[PipelineStage](#) constructor.
- virtual [~PipelineStage](#) ()=default
Default destructor of the [PipelineStage](#).
- virtual bool [Initialize](#) ()=0
Initializes the [PipelineStage](#).
- virtual [PipelineData](#)< Tdata > [Execute](#) (const std::vector< T > &data, const [PipelineData](#)< Tdata > &tdata) noexcept=0
Processes the vector of objects provided and return the result.

5.70.1 Detailed Description

```
template<typename T, typename Tdata>
class Blade::PipelineStage< T, Tdata >
```

This class describes an abstract stage of a pipeline that processes the specified type of data and returns the specified type of data.

Template Parameters

<i>T</i>	The type of data the PipelineStage will process.
<i>Tdata</i>	The type of data the PipelineStage will return.

5.70.2 Constructor & Destructor Documentation

5.70.2.1 PipelineStage()

```
template<typename T, typename Tdata>
Blade::PipelineStage< T, Tdata >::PipelineStage (
    const std::string & name ) [inline], [explicit]
```

[PipelineStage](#) constructor.

Parameters

<i>name</i>	The name of the PipelineStage .
-------------	---

5.70.3 Member Function Documentation

5.70.3.1 Execute()

```
template<typename T, typename Tdata>
virtual PipelineData<Tdata> Blade::PipelineStage< T, Tdata >::Execute (
    const std::vector< T > & data,
    const PipelineData< Tdata > & tdata ) [pure virtual], [noexcept]
```

Processes the vector of objects provided and return the result.

Parameters

<i>data</i>	The type of data the PipelineStage will process.
<i>tdata</i>	The type of data the PipelineStage will return.

Returns

A PipelineData container with the the appropriate data type encapsulated.

5.70.3.2 Initialize()

```
template<typename T, typename Tdata>
virtual bool Blade::PipelineStage< T, Tdata >::Initialize ( ) [pure virtual]
```

Initializes the PipelineStage.

Returns

TRUE if initialization succeeded, FALSE otherwise.

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

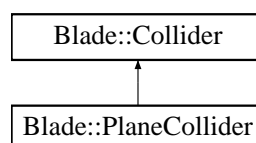
- include/pipeline_stage.h

5.71 Blade::PlaneCollider Class Reference

Bounding Plane class is a collider.

```
#include <plane_collider.h>
```

Inheritance diagram for Blade::PlaneCollider:



Public Member Functions

- **PlaneCollider** (const Vec3f &planeNormal, const float offset)
- bool **Collide** (const [Collider](#) *collider, [ContactManifold](#) &manifold) const noexcept override
- bool **Collide** (const [BoundingSphere](#) *bsphere, [ContactManifold](#) &manifold) const noexcept override
- bool **Collide** (const [PlaneCollider](#) *plane, [ContactManifold](#) &manifold) const noexcept override
- const Vec3f & **GetPlaneNormal** () const noexcept
- void **SetPlaneNormal** (const Vec3f &normal) noexcept
- float **GetOffset** () const noexcept
- void **SetOffset** (const float offset) noexcept

5.71.1 Detailed Description

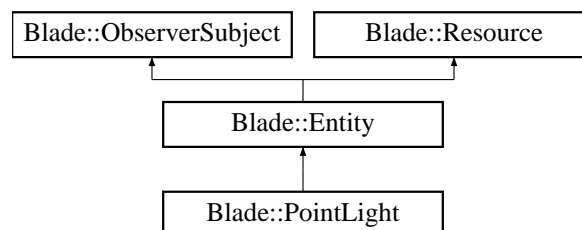
Bounding Plane class is a collider.

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

- include/plane_collider.h
- src/plane_collider.cpp

5.72 Blade::PointLight Class Reference

Inheritance diagram for Blade::PointLight:



Public Member Functions

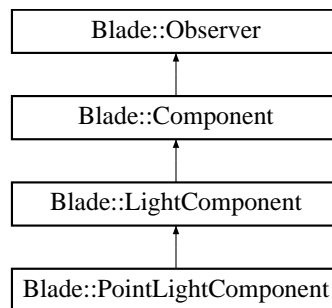
- **PointLight** (const std::string &name, const [PointLightDesc](#) &lightDescription)

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

- include/point_light.h
- src/point_light.cpp

5.73 Blade::PointLightComponent Class Reference

Inheritance diagram for Blade::PointLightComponent:



Public Member Functions

- **PointLightComponent** (const [PointLightDesc](#) &lightDesc, [Entity](#) *parent)
- const [PointLightDesc](#) & **GetLightDescription** () const noexcept
- [PointLightDesc](#) * **GetLightDescriptionPtr** () noexcept

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

- include/point_light_component.h
- src/point_light_component.cpp

5.74 Blade::PointLightDesc Struct Reference

A struct describing a point light.

```
#include <light_component.h>
```

Public Attributes

- Vec4f **ambientIntensity**
- Vec4f **diffuseIntensity**
- Vec4f **specularIntensity**
- Vec3f **position**
- float **constantAttenuation**
- float **linearAttenuation**
- float **quadraticAttenuation**
- Vec2f **pad**

5.74.1 Detailed Description

A struct describing a point light.

This struct is also used to represent a point light in shaders.

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

- include/light_component.h

5.75 Blade::RefCountedContainer< T > Class Template Reference

Public Member Functions

- **RefCountedContainer** (T *item)
- **RefCountedContainer** (const [RefCountedContainer](#) &other)
- **RefCountedContainer** ([RefCountedContainer](#) &&other) noexcept=delete
- [RefCountedContainer](#) & **operator=** (const [RefCountedContainer](#) &other)
- [RefCountedContainer](#) & **operator=** ([RefCountedContainer](#) &&other) noexcept=delete
- void **AddReference** () noexcept
- void **SubtractReference** () noexcept
- int **GetReferenceCount** () const noexcept
- T * **Get** () const noexcept

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

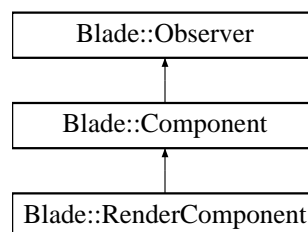
- include/ref_counted_container.h

5.76 Blade::RenderComponent Class Reference

Represents a [RenderComponent](#). The [RenderComponent](#) makes an entity renderable. This component is processed by the [RenderSystem](#).

```
#include <render_component.h>
```

Inheritance diagram for Blade::RenderComponent:



Public Member Functions

- [RenderComponent](#) (Entity *parent)
[RenderComponent](#)'s constructor. Registers the [RenderComponent](#) to the [RenderSystem](#).
- [~RenderComponent](#) ()
[RenderComponent](#)'s destructor. Unregisters the [RenderComponent](#) from the [RenderSystem](#).
- Mesh * [GetMesh](#) () const noexcept
Provides a pointer to the [Mesh](#) contained in the [RenderComponent](#).
- void [SetMesh](#) (Mesh *mesh) noexcept
Sets the specified [Mesh](#) to the [RenderComponent](#).
- const [Material](#) & [GetMaterial](#) () const noexcept
Provides the [Material](#) of the [RenderComponent](#).
- void [SetMaterial](#) (const [Material](#) &material) noexcept
Sets the specified [Material](#) to the [RenderComponent](#).

5.76.1 Detailed Description

Represents a [RenderComponent](#). The [RenderComponent](#) makes an entity renderable. This component is processed by the [RenderSystem](#).

5.76.2 Constructor & Destructor Documentation

5.76.2.1 RenderComponent()

```
Blade::RenderComponent::RenderComponent (
    Entity * parent ) [explicit]
```

[RenderComponent](#)'s constructor. Registers the [RenderComponent](#) to the [RenderSystem](#).

Parameters

<i>parent</i>	The entity the component will be attached to.
---------------	---

5.76.3 Member Function Documentation

5.76.3.1 GetMaterial()

```
const Material & Blade::RenderComponent::GetMaterial ( ) const [noexcept]
```

Provides the [Material](#) of the [RenderComponent](#).

Returns

The [Material](#) of the [RenderComponent](#).

5.76.3.2 GetMesh()

```
Mesh * Blade::RenderComponent::GetMesh ( ) const [noexcept]
```

Provides a pointer to the [Mesh](#) contained in the [RenderComponent](#).

Returns

The pointer to the [Mesh](#) of the [RenderComponent](#).

5.76.3.3 SetMaterial()

```
void Blade::RenderComponent::SetMaterial (
    const Material & material ) [noexcept]
```

Sets the specified [Material](#) to the [RenderComponent](#).

Parameters

<i>material</i>	The Material to be set.
-----------------	---

5.76.3.4 SetMesh()

```
void Blade::RenderComponent::SetMesh (
    Mesh * mesh ) [noexcept]
```

Sets the specified [Mesh](#) to the [RenderComponent](#).

Parameters

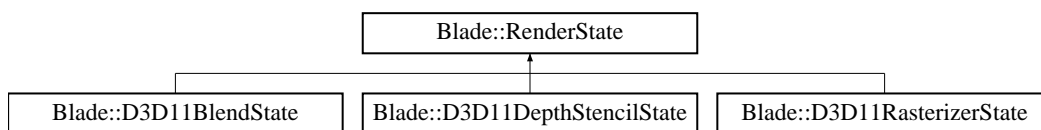
<i>mesh</i>	The mesh to be used when rendering.
-------------	-------------------------------------

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

- include/render_component.h
- src/render_component.cpp

5.77 Blade::RenderState Class Reference

Inheritance diagram for Blade::RenderState:



Public Member Functions

- virtual void **Set** () const noexcept=0

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

- include/render_state.h
- src/render_state.cpp

5.78 Blade::RenderStateManager Class Reference

Public Member Functions

- void **Initialize** () noexcept
- void **Set** (RenderStateType renderState) noexcept

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

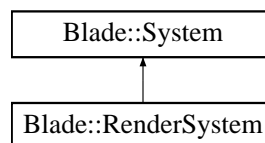
- include/render_state_manager.h
- src/render_state_manager.cpp

5.79 Blade::RenderSystem Class Reference

A [System](#) responsible for processing the RenderComponents by passing them through a specified pipeline.

```
#include <render_system.h>
```

Inheritance diagram for Blade::RenderSystem:



Public Member Functions

- [~RenderSystem](#) ()
Destructor of the [RenderSystem](#). Deallocates the pipeline member.
- void [RegisterComponent](#) ([RenderComponent](#) *renderComponent) noexcept
Registers a [RenderComponent](#) to the [RenderSystem](#).
- void [UnregisterComponent](#) (int id) noexcept
Unregisters a [RenderComponent](#) from the [RenderSystem](#).
- void [SetRenderPassPipeline](#) (RenderPassPipeline *renderPassPipeline) noexcept
Sets the pipeline that the [RenderSystem](#) will pass the RenderComponents through.
- void [ClearRenderPassPipeline](#) () noexcept
Removed the pipeline from the [RenderSystem](#) if one is set.
- bool [Initialize](#) () noexcept override
Initializes the [RenderSystem](#).
- void [Process](#) (float deltaTime=.0f, long time=0) noexcept override
Processes the RenderComponents by passing them through the RenderPassPipeline.
- void **SetSorting** (bool sorting) noexcept

5.79.1 Detailed Description

A [System](#) responsible for processing the RenderComponents by passing them through a specified pipeline.

5.79.2 Member Function Documentation

5.79.2.1 Initialize()

```
bool Blade::RenderSystem::Initialize ( ) [override], [virtual], [noexcept]
```

Initializes the [RenderSystem](#).

Returns

TRUE if initialization is successfull, FALSE otherwise.

Implements [Blade::System](#).

5.79.2.2 Process()

```
void Blade::RenderSystem::Process (
    float deltaTime = .0f,
    long time = 0 ) [override], [virtual], [noexcept]
```

Processes the RenderComponents by passing them through the RenderPassPipeline.

Parameters

<i>deltaTime</i>	The time elapsed from the previous frame of the application.
------------------	--

Implements [Blade::System](#).

5.79.2.3 RegisterComponent()

```
void Blade::RenderSystem::RegisterComponent (
    RenderComponent * renderComponent ) [noexcept]
```

Registers a [RenderComponent](#) to the [RenderSystem](#).

Parameters

<i>renderComponent</i>	The component to be registered to the RenderSytstem for processing.
------------------------	---

5.79.2.4 SetRenderPassPipeline()

```
void Blade::RenderSystem::SetRenderPassPipeline (
    RenderPassPipeline * renderPassPipeline ) [noexcept]
```

Sets the pipeline that the [RenderSystem](#) will pass the RenderComponents through.

Parameters

<i>renderPassPipeline</i>	The pipeline that processes the RenderComponents.
---------------------------	---

5.79.2.5 UnregisterComponent()

```
void Blade::RenderSystem::UnregisterComponent (
    int id ) [noexcept]
```

Unregisters a [RenderComponent](#) from the [RenderSystem](#).

Parameters

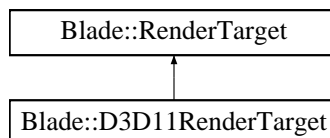
<i>id</i>	The unique id of the RenderComponent to be unregistered.
-----------	--

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

- include/render_system.h
- src/render_system.cpp

5.80 Blade::RenderTarget Class Reference

Inheritance diagram for Blade::RenderTarget:



Public Member Functions

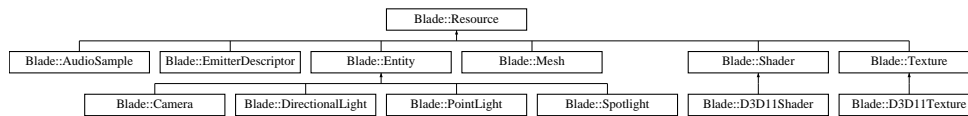
- **RenderTarget** (const Vec2i &size)
- virtual bool **Create** (const Vec2i &size)=0
- virtual bool **Bind** (RenderTargetBindType bind_type) const =0
- virtual bool **Unbind** () const =0
- void **SetSize** (const Vec2i &size) noexcept
- const Vec2i & **GetSize** () const noexcept

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

- include/render_target.h
- src/render_target.cpp

5.81 Blade::Resource Class Reference

Inheritance diagram for Blade::Resource:



Public Member Functions

- **Resource** (unsigned int id)
- unsigned int **GetId** () const noexcept
- void **SetId** (unsigned int id) noexcept
- virtual bool **Load** (const std::wstring &file_name) noexcept=0

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

- include/resource.h
- src/resource.cpp

5.82 Blade::ResourceManager Class Reference

Public Member Functions

- template<typename T >
bool **Load** (const std::wstring &fileName)
- template<typename T >
T * **Get** (const std::wstring &fileName)
- void **RegisterResource** ([Resource](#) *resource, const std::wstring &name)

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

- include/resource_manager.h

5.83 Blade::MathUtils::RungeKutta4Integrator Class Reference

Static Public Member Functions

- static void **Integrate** (Vec3f &position, Vec3f &velocity, const Vec3f &force, float mass, float timeSec, float deltaTime) noexcept

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

- include/math_utils.h
- src/math_utils.cpp

5.84 Blade::SamplePlaylist Struct Reference

Public Attributes

- `std::list< AudioSample * > samples`
- `std::list< AudioSample * >::iterator it`
- `bool loop`
- `bool started { false }`
- `int source_idx { -1 }`

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

- `include/audio_manager.h`
- `src/audio_manager.cpp`

5.85 Blade::Scene Class Reference

Public Member Functions

- `Scene (const Scene &other)=delete`
- `Scene & operator= (const Scene &other)=delete`
- `virtual bool Initialize ()=0`
- `void AddEntity (Entity *object) noexcept`
- `void RemoveEntity (const std::string &name) noexcept`
- `void RemoveEntities () noexcept`
- `const std::vector< Entity * > & GetEntities () const noexcept`
- `Entity * GetEntityByName (const std::string &name) noexcept`
- `virtual void OnKeyDown (unsigned char key, int x, int y) noexcept=0`
- `virtual void OnKeyUp (unsigned char key, int x, int y) noexcept=0`
- `virtual void OnMouseMotion (int x, int y) noexcept=0`
- `virtual void OnMouseClicked (int button, bool state, int x, int y) noexcept=0`
- `virtual void Update (float deltaTime, long time=0) noexcept`
- `virtual void Draw () const noexcept=0`
- `virtual void OnMessage (const MessageContainer< std::string > &msg) const noexcept`

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

- `include/scene.h`
- `src/scene.cpp`

5.86 Blade::SceneManager Class Reference

Public Member Functions

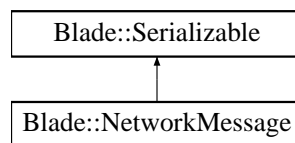
- bool **PushScene** (std::unique_ptr< [Scene](#) > scene) noexcept
- void **PopScene** () noexcept
- void **OnKeyDown** (unsigned char key, int x, int y) noexcept
- void **OnKeyUp** (unsigned char key, int x, int y) noexcept
- void **OnMouseMotion** (int x, int y) noexcept
- void **OnMouseClicked** (int button, bool state, int x, int y) noexcept
- void **OnMessage** (const [MessageContainer](#)< std::string > &msg) noexcept
- void **Update** (float delta_time, long time) noexcept
- void **Draw** () noexcept
- [Scene](#) * **GetCurrentScene** () const noexcept

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

- include/scene_manager.h
- src/scene_manager.cpp

5.87 Blade::Serializable Class Reference

Inheritance diagram for Blade::Serializable:



Public Member Functions

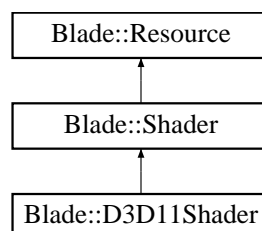
- virtual std::vector< Byte > **Serialize** () noexcept=0

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

- include/serializable.h

5.88 Blade::Shader Class Reference

Inheritance diagram for Blade::Shader:



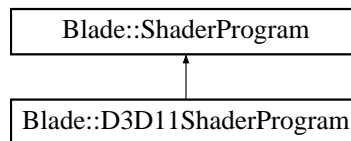
Additional Inherited Members

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

- include/shader.h

5.89 Blade::ShaderProgram Class Reference

Inheritance diagram for Blade::ShaderProgram:



Public Member Functions

- **ShaderProgram** (const [ShaderProgram](#) &)=default
- [ShaderProgram](#) & **operator=** (const [ShaderProgram](#) &)=default
- virtual bool **Create** (const [ShaderProgramDesc](#) &shaderProgramDesc) noexcept=0
- virtual void **Bind** () const noexcept=0

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

- include/shader_program.h
- src/shader_program.cpp

5.90 Blade::ShaderProgramDesc Struct Reference

Public Attributes

- std::string **name**
- unsigned int **inputLayoutMask**
- std::wstring **vertexShader**
- std::wstring **fragmentShader**
- std::wstring **hullShader**
- std::wstring **domainShader**
- std::wstring **geometryShader**

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

- include/shader_program.h

5.91 Blade::ShaderProgramManager Class Reference

Public Member Functions

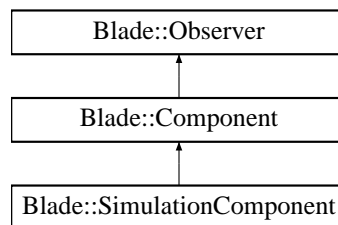
- bool **Create** (const [ShaderProgramDesc](#) &shaderProgramDesc) noexcept
- [ShaderProgram](#) * **Get** (const std::string &progName) noexcept

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

- include/shader_program_manager.h
- src/shader_program_manager.cpp

5.92 Blade::SimulationComponent Class Reference

Inheritance diagram for Blade::SimulationComponent:



Public Member Functions

- **SimulationComponent** ([Entity](#) *parent, float mass)
- void **SetAcceleration** (const Vec3f &acc) noexcept
- const Vec3f & **GetAcceleration** () const noexcept
- void **AddForce** (const Vec3f &force) noexcept
- void **SetForce** (const Vec3f &force) noexcept
- void **SetPreviousForce** (const Vec3f &force) noexcept
- const Vec3f & **GetForce** () const noexcept
- const Vec3f & **GetPreviousForce** () const noexcept
- void **ResetForce** () noexcept
- void **SetVelocity** (const Vec3f &velocity) noexcept
- void **SetPreviousVelocity** (const Vec3f &velocity) noexcept
- const Vec3f & **GetVelocity** () const noexcept
- const Vec3f & **GetPreviousVelocity** () const noexcept
- void **SetPreviousPosition** (const Vec3f &position) noexcept
- const Vec3f & **GetPreviousPosition** () const noexcept
- float **GetMass** () const noexcept
- float **GetInverseMass** () const noexcept
- bool **IsActive** () const noexcept
- void **SetActive** (bool active) noexcept

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

- include/simulation_component.h
- src/simulation_component.cpp

5.93 Blade::SimulationComponentState Struct Reference

Public Attributes

- Vec3f **force**
- Vec3f **velocity**
- float **mass**
- [SimulationComponent](#) * **parent** { nullptr }

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

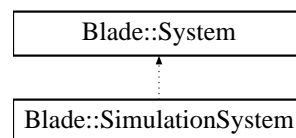
- include/simulation_component.h

5.94 Blade::SimulationSystem Class Reference

The simulation system of the engine.

```
#include <simulation_system.h>
```

Inheritance diagram for Blade::SimulationSystem:



Public Member Functions

- [SimulationSystem](#) & **operator=** ([SimulationSystem](#) &)=delete
- **SimulationSystem** ([SimulationSystem](#) &)=delete
- bool **Initialize** () noexcept override

Pure virtual method implemented by the engine's systems to perform their initialization.
- void **Process** (float deltaTime=.0f, long time=0) noexcept override

Pure virtual method implemented by the engine's systems to process the registered components.
- void **RegisterComponent** ([SimulationComponent](#) *simComp) noexcept
- void **RegisterComponent** ([ColliderComponent](#) *colComp) noexcept
- void **UnregisterComponent** ([SimulationComponent](#) *simComp) noexcept
- void **UnregisterComponent** ([ColliderComponent](#) *colComp) noexcept
- const std::vector< [SimulationComponent](#) * > & **GetSimulationComponents** () const noexcept

Public Attributes

- float **timeSec**

Static Public Attributes

- static float **frequency** = 2000.0f
- static float **elasticity** = 0.3f
- static float **friction** = 1.0f
- static float **dt** = 0.0f
- static float **dtScale** = 1.0f

5.94.1 Detailed Description

The simulation system of the engine.

Performs the simulation routine: update, detection, response using threads.

5.94.2 Member Function Documentation

5.94.2.1 Initialize()

```
bool Blade::SimulationSystem::Initialize ( ) [override], [virtual], [noexcept]
```

Pure virtual method implemented by the engine's systems to perform their initialization.

Returns

TRUE if initialization is successfull, FALSE otherwise.

Implements [Blade::System](#).

5.94.2.2 Process()

```
void Blade::SimulationSystem::Process (
    float deltaTime = .0f,
    long time = 0 ) [override], [virtual], [noexcept]
```

Pure virtual method implemented by the engine's systems to process the registered components.

Parameters

<i>deltaTime</i>	The time elapsed from the previous frame of the application.
------------------	--

Implements [Blade::System](#).

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

- include/simulation_system.h
- src/simulation_system.cpp

5.95 Blade::Socket Class Reference

Public Member Functions

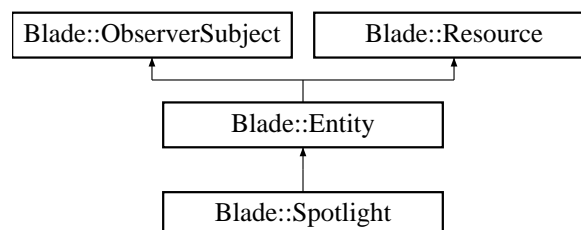
- **Socket** (SocketHandle handle)
- **Socket** (const [Socket](#) &other)=delete
- **Socket** ([Socket](#) &&other) noexcept
- bool **Connect** (const std::string &host, unsigned short port, [ConnectionInfo](#) *connection_info=nullptr) const noexcept
- bool **Listen** (unsigned short port, int maxQueueSize=8) const noexcept
- void **Close** () noexcept
- [Socket](#) **Accept** ([ConnectionInfo](#) *connectionInfo=nullptr) const noexcept
- bool **IsValid** () const noexcept
- SocketHandle **GetHandle** () const noexcept
- void **SetHandle** (SocketHandle handle) noexcept
- bool **Send** (const char *buffer, int size) const noexcept
- int **Receive** (char *buffer, int size) const noexcept

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

- include/socket.h
- src/socket.cpp

5.96 Blade::Spotlight Class Reference

Inheritance diagram for Blade::Spotlight:



Public Member Functions

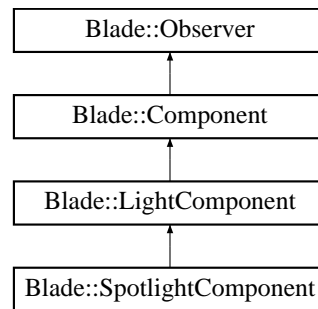
- **Spotlight** (const std::string &name, const [SpotlightDesc](#) &lightDescription)

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

- include/spotlight.h
- src/spotlight.cpp

5.97 Blade::SpotlightComponent Class Reference

Inheritance diagram for Blade::SpotlightComponent:



Public Member Functions

- **SpotlightComponent** (const [SpotlightDesc](#) &lightDesc, [Entity](#) *parent)
- const [SpotlightDesc](#) & **GetLightDescription** () const noexcept
- [SpotlightDesc](#) * **GetLightDescriptionPtr** () noexcept

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

- include/spotlight_component.h
- src/spotlight_component.cpp

5.98 Blade::SpotlightDesc Struct Reference

A struct describing a spotlight.

```
#include <light_component.h>
```

Public Attributes

- Vec4f **ambientIntensity**
- Vec4f **diffuseIntensity**
- Vec4f **specularIntensity**
- Vec3f **position**
- float **constantAttenuation**
- float **linearAttenuation**
- float **quadraticAttenuation**
- Vec3f **direction**
- float **spotCutoff**
- float **spotExponent**
- float **pad**

5.98.1 Detailed Description

A struct describing a spotlight.

This struct is also used to represent a spotlight in shaders.

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

- include/light_component.h

5.99 Blade::MathUtils::State Struct Reference

Public Attributes

- float **x** { 0.0f }
- float **v** { 0.0f }
- float **force** { 0.0f }
- float **mass** { 0.0f }

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

- include/math_utils.h

5.100 Blade::StreamPlaylist Struct Reference

Public Attributes

- std::list< std::string > **files**
- std::list< std::string >::iterator **it**
- bool **loop**
- bool **started** { false }
- int **stream_idx** { -1 }

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

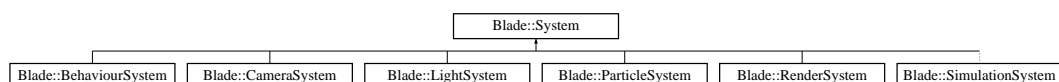
- include/audio_manager.h
- src/audio_manager.cpp

5.101 Blade::System Class Reference

An interface that represents a system of the engine.

```
#include <system.h>
```

Inheritance diagram for Blade::System:



Public Member Functions

- [System](#) ()=default
Default constructor of a [System](#).
- virtual [~System](#) ()
Default destructor of a [System](#).
- virtual bool [Initialize](#) () noexcept=0
Pure virtual method implemented by the engine's systems to perform their initialization.
- virtual void [Process](#) (float deltaTime=.0f, long time=0) noexcept=0
Pure virtual method implemented by the engine's systems to process the registered components.

5.101.1 Detailed Description

An interface that represents a system of the engine.

5.101.2 Member Function Documentation

5.101.2.1 Initialize()

```
virtual bool Blade::System::Initialize ( ) [pure virtual], [noexcept]
```

Pure virtual method implemented by the engine's systems to perform their initialization.

Returns

TRUE if initialization is successfull, FALSE otherwise.

Implemented in [Blade::SimulationSystem](#), [Blade::LightSystem](#), [Blade::RenderSystem](#), [Blade::ParticleSystem](#), and [Blade::BehaviourSystem](#).

5.101.2.2 Process()

```
virtual void Blade::System::Process (
    float deltaTime = .0f,
    long time = 0 ) [pure virtual], [noexcept]
```

Pure virtual method implemented by the engine's systems to process the registered components.

Parameters

<i>deltaTime</i>	The time elapsed from the previous frame of the application.
------------------	--

Implemented in [Blade::SimulationSystem](#), [Blade::LightSystem](#), [Blade::RenderSystem](#), [Blade::ParticleSystem](#), and

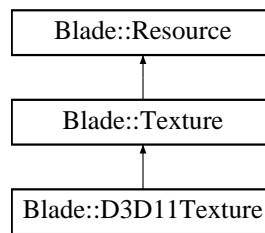
[Blade::BehaviourSystem](#).

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

- include/system.h
- src/system.cpp

5.102 Blade::Texture Class Reference

Inheritance diagram for Blade::Texture:



Public Member Functions

- **Texture** (TextureType textureType)
- virtual void **Bind** () const noexcept=0
- void **SetTextureType** (TextureType texture_type) noexcept
- TextureType **GetTextureType** () const noexcept

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

- include/texture.h

5.103 Blade::ThreadPool Class Reference

Public Member Functions

- bool **Initialize** ()
- void **Wait** ()
- void **Terminate** ()
- void **AddTask** (std::function< void()> job)
- void **AddTasks** (const std::vector< std::function< void()>> &jobs)
- size_t **QueuedTaskCount** () const
- size_t **ActiveTaskCount** () const
- size_t **PendingTaskCount** () const

5.103.1 Member Function Documentation

5.103.1.1 Initialize()

```
bool Blade::ThreadPool::Initialize ( )
```

Get the system's supported thread count.

Spawn the worker threads.

The workers will execute an infinite loop function and will wait for a job to enter the job queue. Once a job is in the the queue the threads will wake up to acquire and execute it.

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

- include/thread_pool.h
- src/thread_pool.cpp

5.104 Blade::ThumbStick Struct Reference

Thumbstick structure to hold X/Y axis information.

```
#include <input_state.h>
```

Public Attributes

- float **axisX**
- float **axisY**

5.104.1 Detailed Description

Thumbstick structure to hold X/Y axis information.

Uses STICK_THRESHOLD to normalize to floating point values in [0..1] range

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

- include/input_state.h

5.105 Blade::Timer Class Reference

Public Member Functions

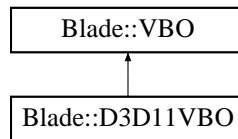
- void **Reset** () noexcept
- void **Start** () noexcept
- void **Stop** () noexcept
- bool **IsRunning** () const noexcept
- long long **GetMsec** () const noexcept
- double **GetSec** () const noexcept
- double **GetDelta** () const noexcept

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

- include/timer.h

5.106 Blade::VBO Class Reference

Inheritance diagram for Blade::VBO:



Public Member Functions

- void **SetVertexCount** (const unsigned int vertexCount) noexcept
- unsigned int **GetVertexCount** () const noexcept
- void **SetPrimitiveTopology** (PrimitiveTopology primitiveTopology) noexcept
- PrimitiveTopology **GetPrimitiveTopology** () const noexcept
- virtual bool **Create** (const std::vector< [Vertex](#) > &vertices, PrimitiveTopology primitiveTopology) noexcept=0
- virtual void **Bind** () const noexcept=0
- virtual void **Draw** () const noexcept=0

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

- include/VBO.h
- src/VBO.cpp

5.107 Blade::Vertex Struct Reference

Public Member Functions

- **Vertex** (const Vec3f &p, const Vec3f &n, const Vec3f &tan, const Vec2f &tcoord, const Vec4f &col)

Public Attributes

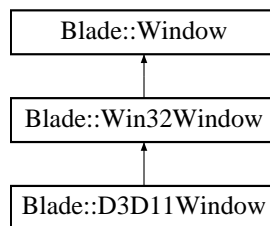
- Vec3f **position**
- Vec3f **normal**
- Vec3f **tangent**
- Vec2f **texcoord**
- Vec4f **color**

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

- include/vertex.h

5.108 Blade::Win32Window Class Reference

Inheritance diagram for Blade::Win32Window:



Public Member Functions

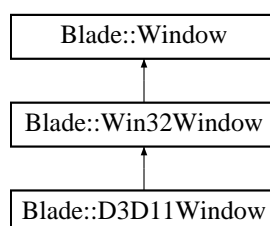
- **Win32Window** (const std::wstring &title, const Vec2i &size, const Vec2i &position, const unsigned int windowId, const bool focused, const bool minimized, const bool resizable, const bool show_cursor, const [WindowFunctionCallbacks](#) &callbacks)
- virtual LRESULT CALLBACK **WinProc** (HWND handle, UINT msg, WPARAM wParam, LPARAM lParam)
- **Win32Window** (const [Win32Window](#) &)=delete
- [Win32Window](#) & **operator=** (const [Win32Window](#) &)=delete
- HWND **GetHandle** () const
- void **SetHandle** (HWND hwnd)
- HWND **GetParent** () const
- void **SetParent** (HWND hwnd)
- HMENU **GetMenu** () const
- void **SetMenu** (HMENU hmenu)
- unsigned int **GetFlags** () const
- void **SetFlags** (unsigned int flags)
- unsigned int **GetFlagsEx** () const
- void **SetFlagsEx** (unsigned int flags_ex)

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

- include/win32_window.h
- src/win32_window.cpp

5.109 Blade::Window Class Reference

Inheritance diagram for Blade::Window:



Public Member Functions

- **Window** (const std::wstring &title, const Vec2i &size, const Vec2i &position, const unsigned int windowId, const bool focused, const bool minimized, const bool resizable, const bool showCursor, const [WindowFunctionCallbacks](#) &callbacks)
- **Window** (const [Window](#) &win)=delete
- **Window** & **operator=** (const [Window](#) &win)=delete
- const std::wstring & **GetTitle** () const noexcept
- void **SetSize** (const Vec2i &size) noexcept
- const Vec2i & **GetSize** () const noexcept
- unsigned int **GetId** () const noexcept
- const Vec2i & **GetMousePosition** () const noexcept
- void **SetMousePosition** (const Vec2i &mousePos) noexcept
- void **SetFocus** (const bool focus) noexcept
- bool **IsFocused** () const noexcept
- void **SetMinimized** (const bool minimized) noexcept
- bool **IsMinimized** () const noexcept
- void **SetResizable** (const bool resizable) noexcept
- bool **IsResizable** () const noexcept
- void **SetChangedSize** (const bool state) noexcept
- bool **ChangedSize** () const noexcept
- void **SetRedisplay** (const bool redisplay) noexcept
- void **SetShowCursor** (bool show) noexcept
- bool **ShowCursor** () const noexcept
- void **SetWindowCallbacks** (const [WindowFunctionCallbacks](#) &callbacks) noexcept
- const [WindowFunctionCallbacks](#) & **GetCallbacks** () const noexcept
- virtual void **SwapBuffers** (unsigned syncInterval=0) const noexcept=0

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

- include/window.h
- src/window.cpp

5.110 Blade::WindowFunctionCallbacks Struct Reference

Public Attributes

- AddRemoveInputDeviceFunc **device_change_func** { nullptr }
- ReshapeFunc **reshape_func** { nullptr }
- KeyboardFunc **keyboard_func** { nullptr }
- KeyboardUpFunc **keyboard_up_func** { nullptr }
- SpecialFunc **special_func** { nullptr }
- SpecialUpFunc **special_up_func** { nullptr }
- MouseFunc **mouse_func** { nullptr }
- MotionFunc **motion_func** { nullptr }
- PassiveMotionFunc **passive_motion_func** { nullptr }

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

- include/windowing_types.h

5.111 Blade::WindowingService Class Reference

Public Member Functions

- **WindowingService** (const [WindowingService](#) &service)=delete
- [WindowingService](#) & **operator=** (const [WindowingService](#) &service)=delete

Static Public Member Functions

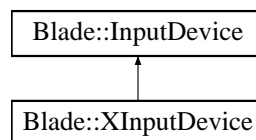
- static void **AddWindow** ([Window](#) *window)
- static void **Create** (const std::wstring &title, const Vec2i &size, const Vec2i &position, const bool focused, const bool minimized, const bool resizable, const bool showCursor, const bool enableMSAA, const int MSAASampleCount, const [WindowFunctionCallbacks](#) &callbacks)
- static void **DestroyWindow** (unsigned int win_id)
- static [Window](#) * **GetWindow** (unsigned int win_id) noexcept
- static [Window](#) * **GetWindow** (const std::wstring &title) noexcept
- static size_t **GetWindowCount** () noexcept
- static void **SwapBuffers** (int syncInterval) noexcept

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

- include/windowing_service.h
- src/windowing_service.cpp

5.112 Blade::XInputDevice Class Reference

Inheritance diagram for Blade::XInputDevice:



Public Member Functions

- **XInputDevice** (int device_id, DeviceType devType)
- void **Update** (float deltaTime) override
- bool **SetVibration** (float leftMotor, float rightMotor) const override
- bool **IsConnected** () const override

Protected Member Functions

- bool **Initialize** () override

Additional Inherited Members

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

- include/xinput_device.h
- src/xinput_device.cpp

Index

~CameraComponent
Blade::CameraComponent, [29](#)

AbstractViewport
Blade::AbstractViewport, [18](#)

ActiveDeviceExists
Blade::InputManager, [58](#)

AddStage
Blade::Pipeline, [79](#)

AssignDeviceToPlayer
Blade::InputManager, [59](#)

BehaviourComponent
Blade::BehaviourComponent, [23](#)

Blade, [11](#)

Blade::AbstractViewport, [17](#)
AbstractViewport, [18](#)
GetRect, [18](#)
SetRect, [18](#)

Blade::Animation, [19](#)

Blade::Application, [19](#)

Blade::AudioManager, [20](#)

Blade::AudioSample, [21](#)

Blade::AudioSource, [21](#)

Blade::AudioStream, [22](#)

Blade::AudioStreamBuffer, [22](#)

Blade::BehaviourComponent, [23](#)
BehaviourComponent, [23](#)
Update, [24](#)

Blade::BehaviourSystem, [24](#)
Initialize, [25](#)
Process, [25](#)
RegisterComponent, [25](#)
UnregisterComponent, [26](#)

Blade::BoundingSphere, [26](#)

Blade::Camera, [27](#)

Blade::CameraComponent, [27](#)
~CameraComponent, [29](#)
CameraComponent, [28](#), [29](#)
GetClippingPlanes, [30](#)
GetFarPlane, [30](#)
GetFov, [30](#)
GetNearPlane, [30](#)
GetProjectionMatrix, [31](#)
GetViewMatrix, [31](#)
GetViewport, [31](#)
SetClippingPlanes, [31](#), [32](#)
SetFarPlane, [32](#)
SetFov, [32](#)
SetNearPlane, [33](#)

SetViewMatrix, [33](#)

SetViewport, [33](#)

Blade::CameraDesc, [33](#)

Blade::CameraSystem, [34](#)

GetActiveCamera, [35](#)

GetActiveCameraProjectionMatrtix, [35](#)

GetActiveCameraViewMatrix, [35](#)

GetActiveCameraViewport, [35](#)

GetCamera, [35](#)

RegisterComponent, [36](#)

SetActiveCamera, [36](#)

UnregisterComponent, [36](#)

Blade::Collider, [37](#)

Blade::ColliderComponent, [37](#)

Blade::Command, [38](#)

Blade::Component, [38](#)

Component, [39](#)

GetId, [39](#)

GetParent, [40](#)

GetType, [40](#)

OnMessage, [40](#)

Blade::ConfigEntry, [41](#)

Blade::ConfigFile, [41](#)

Blade::ConnectionInfo, [42](#)

Blade::ContactManifold, [42](#)

Blade::D3D11BlendState, [42](#)

Blade::D3D11Context, [43](#)

Blade::D3D11DepthStencilState, [43](#)

Blade::D3D11IBO, [44](#)

Blade::D3D11RasterizerState, [44](#)

Blade::D3D11RenderTarget, [45](#)

Blade::D3D11Shader, [45](#)

Blade::D3D11ShaderProgram, [46](#)

Blade::D3D11Texture, [46](#)

Blade::D3D11VBO, [47](#)

Blade::D3D11Viewport, [47](#)

D3D11Viewport, [48](#)

Blade::D3D11Window, [48](#)

Blade::DirectionalLight, [49](#)

Blade::DirectionalLightComponent, [49](#)

Blade::DirectionalLightDesc, [50](#)

Blade::EmitterComponent, [50](#)

Update, [51](#)

Blade::EmitterDescriptor, [52](#)

Blade::EngineContext, [53](#)

Blade::Entity, [53](#)

Blade::GAPIContext, [54](#)

Blade::IBO, [55](#)

Blade::InputComponent, [55](#)

- Update, 56
- Blade::InputDevice, 56
- Blade::InputManager, 58
 - ActiveDeviceExists, 58
 - AssignDeviceToPlayer, 59
 - DevicePoolQueryType, 59
 - EnumerateDevices, 60
 - GetActiveDevice, 60
 - GetAnalogStickVector, 60
 - Initialize, 61
 - PooledDeviceExists, 61
 - QueryAllKeyStates, 61
 - QueryDeviceAllStates, 62
 - QueryDeviceState, 62
 - QueryKeyState, 62
 - QueryMouseButtonState, 63
 - QueryMouseMovement, 63
 - QueryMouseMovementNormalized, 63
 - QueryMousePosition, 63
 - UnassignDevice, 64
 - Update, 64
- Blade::InputState, 64
- Blade::JoypadInputComponent, 65
 - Update, 66
- Blade::KeyboardInput, 67
 - QueryAllKeyStates, 67
 - QueryKeyState, 67
- Blade::KeyboardInputComponent, 68
 - Update, 69
- Blade::Keyframe< T >, 69
- Blade::LightComponent, 70
- Blade::LightSystem, 70
 - GetDirectionalLightDescriptions, 71
 - GetPointLightDescriptions, 71
 - GetSpotlightDescriptions, 71
 - Initialize, 72
 - Process, 72
 - RegisterComponent, 72
 - UnregisterComponent, 73
- Blade::ManifoldEntry, 73
- Blade::Material, 73
- Blade::MathUtils::Derivative, 49
- Blade::MathUtils::RungeKutta4Integrator, 92
- Blade::MathUtils::State, 101
- Blade::Mesh, 74
- Blade::Message< T >, 75
- Blade::NCF, 75
- Blade::NetworkManager, 75
- Blade::NetworkMessage, 76
- Blade::Observer, 76
- Blade::ObserverSubject, 76
- Blade::OggVorbisStream, 77
- Blade::Particle, 77
- Blade::ParticleSystem, 78
 - Initialize, 78
 - Process, 78
- Blade::Pipeline
 - AddStage, 79
 - Execute, 80
- Blade::Pipeline< T, Tdata >, 79
- Blade::PipelineData
 - Get, 81
 - PipelineData, 81
- Blade::PipelineData< T >, 80
- Blade::PipelineStage
 - Execute, 82
 - Initialize, 83
 - PipelineStage, 82
- Blade::PipelineStage< T, Tdata >, 81
- Blade::PlaneCollider, 83
- Blade::PointLight, 84
- Blade::PointLightComponent, 85
- Blade::PointLightDesc, 85
- Blade::RefCountedContainer< T >, 86
- Blade::RenderComponent, 86
 - GetMaterial, 87
 - GetMesh, 87
 - RenderComponent, 87
 - SetMaterial, 87
 - SetMesh, 88
- Blade::RenderState, 88
- Blade::RenderStateManager, 89
- Blade::RenderSystem, 89
 - Initialize, 90
 - Process, 90
 - RegisterComponent, 90
 - SetRenderPassPipeline, 90
 - UnregisterComponent, 91
- Blade::RenderTarget, 91
- Blade::Resource, 92
- Blade::ResourceManager, 92
- Blade::SamplePlaylist, 93
- Blade::Scene, 93
- Blade::SceneManager, 94
- Blade::Serializable, 94
- Blade::Shader, 94
- Blade::ShaderProgram, 95
- Blade::ShaderProgramDesc, 95
- Blade::ShaderProgramManager, 96
- Blade::SimulationComponent, 96
- Blade::SimulationComponentState, 97
- Blade::SimulationSystem, 97
 - Initialize, 98
 - Process, 98
- Blade::Socket, 99
- Blade::Spotlight, 99
- Blade::SpotlightComponent, 100
- Blade::SpotlightDesc, 100
- Blade::StreamPlaylist, 101
- Blade::System, 101
 - Initialize, 102
 - Process, 102
- Blade::Texture, 103
- Blade::ThreadPool, 103
 - Initialize, 103
- Blade::ThumbStick, 104

- Blade::Timer, 104
- Blade::VBO, 105
- Blade::Vertex, 105
- Blade::Win32Window, 106
- Blade::Window, 106
- Blade::WindowFunctionCallbacks, 107
- Blade::WindowingService, 108
- Blade::XInputDevice, 108
- CameraComponent
 - Blade::CameraComponent, 28, 29
- Component
 - Blade::Component, 39
- D3D11Viewport
 - Blade::D3D11Viewport, 48
- DevicePoolQueryType
 - Blade::InputManager, 59
- EnumerateDevices
 - Blade::InputManager, 60
- Execute
 - Blade::Pipeline, 80
 - Blade::PipelineStage, 82
- Get
 - Blade::PipelineData, 81
- GetActiveCamera
 - Blade::CameraSystem, 35
- GetActiveCameraProjectionMatrix
 - Blade::CameraSystem, 35
- GetActiveCameraViewMatrix
 - Blade::CameraSystem, 35
- GetActiveCameraViewport
 - Blade::CameraSystem, 35
- GetActiveDevice
 - Blade::InputManager, 60
- GetAnalogStickVector
 - Blade::InputManager, 60
- GetCamera
 - Blade::CameraSystem, 35
- GetClippingPlanes
 - Blade::CameraComponent, 30
- GetDirectionalLightDescriptions
 - Blade::LightSystem, 71
- GetFarPlane
 - Blade::CameraComponent, 30
- GetFov
 - Blade::CameraComponent, 30
- GetId
 - Blade::Component, 39
- GetMaterial
 - Blade::RenderComponent, 87
- GetMesh
 - Blade::RenderComponent, 87
- GetNearPlane
 - Blade::CameraComponent, 30
- GetParent
 - Blade::Component, 40
- GetPointLightDescriptions
 - Blade::LightSystem, 71
- GetProjectionMatrix
 - Blade::CameraComponent, 31
- GetRect
 - Blade::AbstractViewport, 18
- GetSpotlightDescriptions
 - Blade::LightSystem, 71
- GetType
 - Blade::Component, 40
- GetViewMatrix
 - Blade::CameraComponent, 31
- GetViewport
 - Blade::CameraComponent, 31
- Initialize
 - Blade::BehaviourSystem, 25
 - Blade::InputManager, 61
 - Blade::LightSystem, 72
 - Blade::ParticleSystem, 78
 - Blade::PipelineStage, 83
 - Blade::RenderSystem, 90
 - Blade::SimulationSystem, 98
 - Blade::System, 102
 - Blade::ThreadPool, 103
- OnMessage
 - Blade::Component, 40
- PipelineData
 - Blade::PipelineData, 81
- PipelineStage
 - Blade::PipelineStage, 82
- PooledDeviceExists
 - Blade::InputManager, 61
- Process
 - Blade::BehaviourSystem, 25
 - Blade::LightSystem, 72
 - Blade::ParticleSystem, 78
 - Blade::RenderSystem, 90
 - Blade::SimulationSystem, 98
 - Blade::System, 102
- QueryAllKeyStates
 - Blade::InputManager, 61
 - Blade::KeyboardInput, 67
- QueryDeviceAllStates
 - Blade::InputManager, 62
- QueryDeviceState
 - Blade::InputManager, 62
- QueryKeyState
 - Blade::InputManager, 62
 - Blade::KeyboardInput, 67
- QueryMouseButtonState
 - Blade::InputManager, 63
- QueryMouseMovement
 - Blade::InputManager, 63
- QueryMouseMovementNormalized
 - Blade::InputManager, 63

- QueryMousePosition
 - Blade::InputManager, [63](#)
- RegisterComponent
 - Blade::BehaviourSystem, [25](#)
 - Blade::CameraSystem, [36](#)
 - Blade::LightSystem, [72](#)
 - Blade::RenderSystem, [90](#)
- RenderComponent
 - Blade::RenderComponent, [87](#)
- SetActiveCamera
 - Blade::CameraSystem, [36](#)
- SetClippingPlanes
 - Blade::CameraComponent, [31](#), [32](#)
- SetFarPlane
 - Blade::CameraComponent, [32](#)
- SetFov
 - Blade::CameraComponent, [32](#)
- SetMaterial
 - Blade::RenderComponent, [87](#)
- SetMesh
 - Blade::RenderComponent, [88](#)
- SetNearPlane
 - Blade::CameraComponent, [33](#)
- SetRect
 - Blade::AbstractViewport, [18](#)
- SetRenderPassPipeline
 - Blade::RenderSystem, [90](#)
- SetViewMatrix
 - Blade::CameraComponent, [33](#)
- SetViewport
 - Blade::CameraComponent, [33](#)
- UnassignDevice
 - Blade::InputManager, [64](#)
- UnregisterComponent
 - Blade::BehaviourSystem, [26](#)
 - Blade::CameraSystem, [36](#)
 - Blade::LightSystem, [73](#)
 - Blade::RenderSystem, [91](#)
- Update
 - Blade::BehaviourComponent, [24](#)
 - Blade::EmitterComponent, [51](#)
 - Blade::InputComponent, [56](#)
 - Blade::InputManager, [64](#)
 - Blade::JoypadInputComponent, [66](#)
 - Blade::KeyboardInputComponent, [69](#)