

# Namespace Cobilas.GodotEngine.Utility

## Classes

[Coroutine](#)

[CoroutineManager](#)

[GDDirectory](#)

[GDFeature](#)

[GDFile](#)

[GDFileBase](#)

[GDIONull](#)

[Gizmos](#)

[Randomico](#)

[Screen](#)

## Structs

[FixedRunTimeSecond](#)

[RunTimeSecond](#)

## Interfaces

[IYieldCoroutine](#)

[IYieldFixedUpdate](#)

[IYieldUpdate](#)

[IYieldVolatile](#)

## Enums

[GDFileAttributes](#)

[ScreenMode](#)

# Class Coroutine

Namespace: [Cobilas.GodotEngine.Utility](#)

Assembly: com.cobilas.godot.utility.dll

```
public sealed class Coroutine : IEnumerable, IDisposable
```

## Inheritance

[object](#) ← Coroutine

## Implements

[IEnumerable](#), [IDisposable](#)

## Inherited Members

[object.ToString\(\)](#), [object.Equals\(object\)](#), [object.Equals\(object, object\)](#),  
[object.ReferenceEquals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#)

# Constructors

## Coroutine(IEnumerator?, string?)

```
public Coroutine(IEnumerator? enumerator, string? iD)
```

## Parameters

enumerator [IEnumerator](#)

iD [string](#)

# Properties

## ID

```
public string ID { get; }
```

## Property Value

[string](#)

## IsCancellationRequested

```
public bool IsCancellationRequested { get; }
```

Property Value

[bool](#)

## IsRunning

```
public bool IsRunning { get; }
```

Property Value

[bool](#)

## Methods

### Cancel()

```
public void Cancel()
```

### CancelAfter(int)

```
public void CancelAfter(int millisecondsDelay)
```

Parameters

millisecondsDelay [int](#)

## CancelAfter(TimeSpan)

```
public void CancelAfter(TimeSpan delay)
```

### Parameters

delay [TimeSpan](#)

## Dispose()

Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources.

```
public void Dispose()
```

## ~Coroutine()

```
protected ~Coroutine()
```

# Class CoroutineManager

Namespace: [Cobilas.GodotEngine.Utility](#)

Assembly: com.cobilas.godot.utility.dll

```
[RunTimeInitializationClass("CoroutineManager")]
public class CoroutineManager : Node, IDisposable
```

## Inheritance

[object](#) ← Object ← Node ← CoroutineManager

## Implements

[IDisposable](#)

## Inherited Members

Node.NotificationEnterTree , Node.NotificationExitTree , Node.NotificationMovedInParent ,  
Node.NotificationReady , Node.NotificationPaused , Node.NotificationUnpaused ,  
Node.NotificationPhysicsProcess , Node.NotificationProcess , Node.NotificationParented ,  
Node.NotificationUnparented , Node.NotificationInstanced , Node.NotificationDragBegin ,  
Node.NotificationDragEnd , Node.NotificationPathChanged , Node.NotificationInternalProcess ,  
Node.NotificationInternalPhysicsProcess , Node.NotificationPostEnterTree ,  
Node.NotificationResetPhysicsInterpolation , Node.NotificationWmMouseEnter ,  
Node.NotificationWmMouseExit , Node.NotificationWmFocusIn , Node.NotificationWmFocusOut ,  
Node.NotificationWmQuitRequest , Node.NotificationWmGoBackRequest ,  
Node.NotificationWmUnfocusRequest , Node.NotificationOsMemoryWarning ,  
Node.NotificationTranslationChanged , Node.NotificationWmAbout , Node.NotificationCrash ,  
Node.NotificationOslmeUpdate , Node.NotificationAppResumed , Node.NotificationAppPaused ,  
Node.GetNode<T>(NodePath) , Node.GetNodeOrNull<T>(NodePath) , [Node.GetChild<T>\(int\)](#) ,  
[Node.GetChildOrNull<T>\(int\)](#) , Node.GetOwner<T>() , Node.GetOwnerOrNull<T>() ,  
Node.GetParent<T>() , Node.GetParentOrNull<T>() , Node.\_EnterTree() , Node.\_ExitTree() ,  
Node.\_GetConfigurationWarning() , Node.\_Input(InputEvent) , Node.\_UnhandledInput(InputEvent) ,  
Node.\_UnhandledKeyInput(InputEventKey) , [Node.AddChildBelowNode\(Node, Node, bool\)](#) ,  
[NodeSetName\(string\)](#) , Node.GetName() , [Node.AddChild\(Node, bool\)](#) , Node.RemoveChild(Node) ,  
Node.GetChildCount() , Node.GetChildren() , [Node.GetChild\(int\)](#) , Node.HasNode(NodePath) ,  
Node.GetNode(NodePath) , Node.GetNodeOrNull(NodePath) , Node.GetParent() ,  
[Node.FindNode\(string, bool, bool\)](#) , [Node.FindParent\(string\)](#) ,  
Node.HasNodeAndResource(NodePath) , Node.GetNodeAndResource(NodePath) , Node.IsInsideTree() ,  
Node.IsAParentOf(Node) , Node.IsGreaterThanOrEqual(Node) , Node.GetPath() , Node.GetPathTo(Node) ,  
[Node.AddToGroup\(string, bool\)](#) , [Node.RemoveFromGroup\(string\)](#) , [Node.IsInGroup\(string\)](#) ,

[Node.MoveChild\(Node, int\)](#) , Node.GetGroups() , Node.Raise() , Node.SetOwner(Node) ,  
Node.GetOwner() , Node.RemoveAndSkip() , Node.GetIndex() , Node.PrintTree() , Node.PrintTreePretty() ,  
[Node.SetFilename\(string\)](#) , Node.GetFilename() , [Node.PropagateNotification\(int\)](#) ,  
[Node.PropagateCall\(string, Array, bool\)](#) , [Node.SetPhysicsProcess\(bool\)](#) ,  
Node.GetPhysicsProcessDeltaTime() , Node.IsPhysicsProcessing() , Node.GetProcessDeltaTime() ,  
[Node.SetProcess\(bool\)](#) , [Node.SetProcessPriority\(int\)](#) , Node.GetProcessPriority() ,  
Node.IsProcessing() , [Node.SetProcessInput\(bool\)](#) , Node.IsProcessingInput() ,  
[Node.SetProcessUnhandledInput\(bool\)](#) , Node.IsProcessingUnhandledInput() ,  
[Node.SetProcessUnhandledKeyInput\(bool\)](#) , Node.IsProcessingUnhandledKeyInput() ,  
Node.SetPauseMode(Node.PauseModeEnum) , Node.GetPauseMode() , Node.CanProcess() ,  
Node.PrintStrayNodes() , NodeGetPositionInParent() , [Node.SetDisplayFolded\(bool\)](#) ,  
Node.IsDisplayedFolded() , [Node.SetProcessInternal\(bool\)](#) , Node.IsProcessingInternal() ,  
[Node.SetPhysicsProcessInternal\(bool\)](#) , Node.IsPhysicsProcessingInternal() ,  
Node.SetPhysicsInterpolationMode(Node.PhysicsInterpolationModeEnum) ,  
Node.GetPhysicsInterpolationMode() , Node.IsPhysicsInterpolated() ,  
Node.IsPhysicsInterpolatedAndEnabled() , Node.ResetPhysicsInterpolation() , Node.GetTree() ,  
Node.CreateTween() , [Node.Duplicate\(int\)](#) , [Node.ReplaceBy\(Node, bool\)](#) ,  
[Node.setSceneInstanceLoadPlaceholder\(bool\)](#) , Node.GetSceneInstanceLoadPlaceholder() ,  
Node.GetViewport() , Node.QueueFree() , Node.RequestReady() , [Node.SetNetworkMaster\(int, bool\)](#) ,  
Node.GetNetworkMaster() , Node.IsNetworkMaster() , Node.GetMultiplayer() ,  
Node.GetCustomMultiplayer() , Node.SetCustomMultiplayer(MultiplayerAPI) ,  
[Node.RpcConfig\(string, MultiplayerAPI.RPCMode\)](#) ,  
[Node.RsetConfig\(string, MultiplayerAPI.RPCMode\)](#) , [Node.SetUniqueNameInOwner\(bool\)](#) ,  
Node.IsUniqueNameInOwner() , [Node.Rpc\(string, params object\[\]\)](#) ,  
[Node.RpcUnreliable\(string, params object\[\]\)](#) , [Node.Rpcld\(int, string, params object\[\]\)](#) ,  
[Node.RpcUnreliableId\(int, string, params object\[\]\)](#) , [Node.Rset\(string, object\)](#) ,  
[Node.RsetId\(int, string, object\)](#) , [Node.RsetUnreliable\(string, object\)](#) ,  
[Node.RsetUnreliableId\(int, string, object\)](#) , Node.UpdateConfigurationWarning() ,  
Node.EditorDescription , Node.\_ImportPath , Node.PauseMode , Node.PhysicsInterpolationMode ,  
Node.Name , Node.UniqueNameInOwner , Node.Filename , Node.Owner , Node.Multiplayer ,  
Node.CustomMultiplayer , Node.ProcessPriority , Object.NotificationPostInitialize ,  
Object.NotificationPredelete , Object.IsInstanceValid(Object) , Object.WeakRef(Object) , Object.Dispose() ,  
[Object.Dispose\(bool\)](#) , Object.ToString() , [Object.ToSignal\(Object, string\)](#) , [Object.Get\(string\)](#) ,  
Object.\_GetPropertyList() , [Object.Notification\(int\)](#) , [Object.Set\(string, object\)](#) , Object.Free() ,  
Object.GetClass() , [Object.IsClass\(string\)](#) , [Object.Set\(string, object\)](#) , [Object.Get\(string\)](#) ,  
[Object.SetIndexed\(NodePath, object\)](#) , Object.GetIndexed(NodePath) , Object.GetPropertyList() ,  
Object.GetMethodList() , [Object.Notification\(int, bool\)](#) , Object.GetInstanceId() ,  
Object.SetScript(Reference) , Object.GetScript() , [Object.SetMeta\(string, object\)](#) ,  
[Object.RemoveMeta\(string\)](#) , [Object.GetMeta\(string, object\)](#) , [Object.HasMeta\(string\)](#) ,  
Object.GetMetaList() , [Object.AddUserSignal\(string, Array\)](#) , [Object.HasUserSignal\(string\)](#) ,

[Object.EmitSignal\(string, params object\[\]\)](#) , [Object.Call\(string, params object\[\]\)](#) ,  
[Object.CallDeferred\(string, params object\[\]\)](#) , [Object.SetDeferred\(string, object\)](#) ,  
[Object.Cally\(string, Array\)](#) , [Object.HasMethod\(string\)](#) , [Object.HasSignal\(string\)](#) ,  
Object.GetSignalList() , [Object.GetSignalConnectionList\(string\)](#) , Object.GetIncomingConnections() ,  
[Object.Connect\(string, Object, string, Array, uint\)](#) , [Object.Disconnect\(string, Object, string\)](#) ,  
[Object.IsConnected\(string, Object, string\)](#) , [Object.SetBlockSignals\(bool\)](#) , Object.IsBlockingSignals() ,  
Object.PropertyListChangedNotify() , [Object.SetMessageTranslation\(bool\)](#) ,  
Object.CanTranslateMessages() , [Object.Tr\(string\)](#) , Object.IsQueuedForDeletion() ,  
Object.NativeInstance , Object.DynamicObject , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Extension Methods

[Node GD CB Extension.FindNodeByName\(Node, string\)](#) ,  
[Node GD CB Extension.FindNodeByName\(Node, string, bool\)](#) ,  
[Node GD CB Extension.FindNodeByName\(Node, string, Type, bool\)](#) ,  
[Node GD CB Extension.FindNodeByName<T>\(Node, string\)](#) ,  
[Node GD CB Extension.FindNodeByName<T>\(Node, string, bool\)](#) ,  
[Node GD CB Extension.FindNodes\(Node, Type\)](#) , [Node GD CB Extension.FindNodes\(Node, Type, bool\)](#) ,  
[Node GD CB Extension.FindNodes<T>\(Node\)](#) , [Node GD CB Extension.FindNodes<T>\(Node, bool\)](#) ,  
[Node GD CB Extension.GetNodePosition\(Node\)](#) , [Node GD CB Extension.GetNodeRotation\(Node\)](#) ,  
[Node GD CB Extension.GetNodeScale\(Node\)](#) , [Node GD CB Extension.Print\(Node, params object\[\]\)](#) ,  
[Node GD CB Extension.SetNodePosition\(Node, Vector3D\)](#) ,  
[Node GD CB Extension.SetNodeRotation\(Node, Vector3D\)](#) ,  
[Node GD CB Extension.SetNodeScale\(Node, Vector3D\)](#)

# Methods

## GenID()

Generates an ID to be used in a coroutine.

```
public static string GenID()
```

Returns

[string](#)

## StartCoroutine(IEnumerator?)

Starts a collating process from an [IEnumerator](#).

```
public static Coroutine StartCoroutine(IEnumerator? enumerator)
```

Parameters

enumerator [IEnumerator](#)

Returns

[Coroutine](#)

## StopAllCoroutines()

Ends all open Coroutines.

```
public static void StopAllCoroutines()
```

## StopCoroutine(Coroutine?)

Ends all open Coroutines.

```
public static void StopCoroutine(Coroutine? Coroutine)
```

Parameters

Coroutine [Coroutine](#)

## \_PhysicsProcess(float)

Called during the physics processing step of the main loop. Physics processing means that the frame rate is synced to the physics, i.e. the `delta` variable should be constant. `delta` is in seconds.

It is only called if physics processing is enabled, which is done automatically if this method is overridden, and can be toggled with [SetPhysicsProcess\(bool\)](#).

Corresponds to the Godot.Node.NotificationPhysicsProcess notification in [Notification\(int\)](#).

Note: This method is only called if the node is present in the scene tree (i.e. if it's not an orphan).

```
public override void _PhysicsProcess(float delta)
```

## Parameters

**delta** [float](#)

## \_Process(float)

Called during the processing step of the main loop. Processing happens at every frame and as fast as possible, so the **delta** time since the previous frame is not constant. **delta** is in seconds.

It is only called if processing is enabled, which is done automatically if this method is overridden, and can be toggled with [SetProcess\(bool\)](#).

Corresponds to the Godot.Node.NotificationProcess notification in [Notification\(int\)](#).

Note: This method is only called if the node is present in the scene tree (i.e. if it's not an orphan).

```
public override void _Process(float delta)
```

## Parameters

**delta** [float](#)

## \_Ready()

Called when the node is "ready", i.e. when both the node and its children have entered the scene tree. If the node has children, their Godot.Node.\_Ready() callbacks get triggered first, and the parent node will receive the ready notification afterwards.

Corresponds to the Godot.Node.NotificationReady notification in [Notification\(int\)](#). See also the **onready** keyword for variables.

Usually used for initialization. For even earlier initialization, may be used. See also Godot.Node.\_EnterTree().

Note: Godot.Node.\_Ready() may be called only once for each node. After removing a node from the scene tree and adding it again, `_ready` will not be called a second time. This can be bypassed by requesting another call with Godot.Node.RequestReady(), which may be called anywhere before adding the node again.

```
public override void _Ready()
```

# Struct FixedRunTimeSecond

Namespace: [Cobilas.GodotEngine.Utility](#)

Assembly: com.cobilas.godot.utility.dll

```
public readonly struct FixedRunTimeSecond : IYieldFixedUpdate, IYieldCoroutine
```

Implements

[IYieldFixedUpdate](#), [IYieldCoroutine](#)

Inherited Members

[ValueType.Equals\(object\)](#) , [ValueType.GetHashCode\(\)](#) , [ValueType.ToString\(\)](#) ,  
[object.Equals\(object, object\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.GetType\(\)](#)

## Constructors

FixedRunTimeSecond(double)

```
public FixedRunTimeSecond(double second)
```

Parameters

second [double](#)

# Class GDDirectory

Namespace: [Cobilas.GodotEngine.Utility](#)

Assembly: com.cobilas.godot.utility.dll

```
public sealed class GDDirectory : GDFileBase, IDisposable
```

## Inheritance

[object](#) ← [GDFileBase](#) ← GDDirectory

## Implements

[IDisposable](#)

## Inherited Members

[GDFileBase.Null](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#)

# Properties

## Attribute

```
public override GDFileAttributes Attribute { get; protected set; }
```

## Property Value

[GDFileAttributes](#)

## Count

```
public int Count { get; }
```

## Property Value

[int](#)

## Name

```
public override string Name { get; }
```

### Property Value

[string](#) ↗

## NameWithoutExtension

```
public override string NameWithoutExtension { get; }
```

### Property Value

[string](#) ↗

## Parent

```
public override GDFFileBase Parent { get; protected set; }
```

### Property Value

[GDFFileBase](#)

## Path

```
public override string Path { get; protected set; }
```

### Property Value

[string](#) ↗

## Methods

## CreateDirectory(string?)

```
public bool CreateDirectory(string? directoryName)
```

### Parameters

directoryName [string](#)

### Returns

[bool](#)

## Dispose()

Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources.

```
public override void Dispose()
```

## ~GDDirectory()

```
protected ~GDDirectory()
```

## GetDirectories()

```
public GDDirectory[] GetDirectories()
```

### Returns

[GDDirectory](#)[]

## GetDirectory(string?, bool)

```
public GDDirectory? GetDirectory(string? relativePath, bool isSubdirectory = false)
```

Parameters

`relativePath` [string](#)

`isSubdirectory` [bool](#)

Returns

[GDDirectory](#)

## GetFile(string?, bool)

```
public GDFile? GetFile(string? name, bool isSubdirectory = false)
```

Parameters

`name` [string](#)

`isSubdirectory` [bool](#)

Returns

[GDFile](#)

## GetFiles(bool)

```
public GDFile[] GetFiles(bool isSubdirectory = false)
```

Parameters

`isSubdirectory` [bool](#)

Returns

[GDFile\[\]](#)

## GetGDDirectory()

```
public static GDDirectory? GetGDDirectory()
```

Returns

[GDDirectory](#)

## GetGDDirectory(string?)

Opens an existing directory of the filesystem. The path argument can be within the project tree (`res://folder`), the user directory (`user://folder`) or an absolute path of the user filesystem (e.g. `/tmp/folder` or `C:\tmp\folder`).

```
public static GDDirectory? GetGDDirectory(string? path)
```

Parameters

`path` [string](#) ↗

Returns

[GDDirectory](#)

## RemoveDirectory(string?)

```
public bool RemoveDirectory(string? directoryName)
```

Parameters

`directoryName` [string](#) ↗

Returns

[bool](#) ↗

## RemoveFile(string?)

```
public bool RemoveFile(string? fileName)
```

Parameters

fileName [string](#)

Returns

[bool](#)

## ToString()

Returns a string that represents the current object.

```
public override string ToString()
```

Returns

[string](#)

A string that represents the current object.

# Class GDFeature

Namespace: [Cobilas.GodotEngine.Utility](#)

Assembly: com.cobilas.godot.utility.dll

```
public static class GDFeature
```

## Inheritance

[object](#) ← GDFeature

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Properties

## HasARM32

Running on a 32-bit ARM build.

```
public static bool HasARM32 { get; }
```

### Property Value

[bool](#)

## HasARM64

Running on a 64-bit ARM build.

```
public static bool HasARM64 { get; }
```

### Property Value

[bool](#)

## HasAndroid

Running on Android.

```
public static bool HasAndroid { get; }
```

Property Value

[bool](#) ↗

## HasDebug

Running on a debug build (including the editor).

```
public static bool HasDebug { get; }
```

Property Value

[bool](#) ↗

## HasETC1

Textures using ETC1 compression are supported.

```
public static bool HasETC1 { get; }
```

Property Value

[bool](#) ↗

## HasETC2

Textures using ETC2 compression are supported.

```
public static bool HasETC2 { get; }
```

Property Value

[bool](#) ↗

## HasEditor

Running on an editor build.

```
public static bool HasEditor { get; }
```

Property Value

[bool](#) ↗

## HasHTML5

Running on HTML5.

```
public static bool HasHTML5 { get; }
```

Property Value

[bool](#) ↗

## HasIOS

Running on iOS.

```
public static bool HasIOS { get; }
```

Property Value

[bool](#) ↗

## HasJavaScript

JavaScript singleton is available.

```
public static bool HasJavaScript { get; }
```

Property Value

[bool](#) ↗

## HasMobile

Host OS is a mobile platform.

```
public static bool HasMobile { get; }
```

Property Value

[bool](#) ↗

## HasOSX

Running on macOS.

```
public static bool HasOSX { get; }
```

Property Value

[bool](#) ↗

## HasPC

Host OS is a PC platform (desktop/laptop).

```
public static bool HasPC { get; }
```

Property Value

[bool](#) ↗

## HasPVRTC

Textures using PVRTC compression are supported.

```
public static bool HasPVRTC { get; }
```

Property Value

[bool](#) ↗

## HasRelease

Running on a release build.

```
public static bool HasRelease { get; }
```

Property Value

[bool](#) ↗

## HasS3TC

Textures using S3TC (DXT/BC) compression are supported.

```
public static bool HasS3TC { get; }
```

Property Value

[bool](#) ↗

## HasServer

Running on the headless server platform.

```
public static bool HasServer { get; }
```

Property Value

[bool](#) ↗

## HasStandalone

Running on a non-editor build.

```
public static bool HasStandalone { get; }
```

Property Value

[bool](#) ↗

## HasUWP

Running on UWP.

```
public static bool HasUWP { get; }
```

Property Value

[bool](#) ↗

## HasWeb

Host OS is a Web browser.

```
public static bool HasWeb { get; }
```

Property Value

[bool](#)

## HasWindows

Running on Windows.

```
public static bool HasWindows { get; }
```

Property Value

[bool](#)

## HasX11

Running on X11 (Linux/BSD desktop).

```
public static bool HasX11 { get; }
```

Property Value

[bool](#)

## HasX32

Running on a 32-bit build (any architecture).

```
public static bool HasX32 { get; }
```

Property Value

[bool](#)

## HasX64

Running on a 64-bit build (any architecture).

```
public static bool HasX64 { get; }
```

Property Value

[bool](#) ↗

## HasX86\_32

Running on a 32-bit x86 build.

```
public static bool HasX86_32 { get; }
```

Property Value

[bool](#) ↗

## HasX86\_64

Running on a 64-bit x86 build.

```
public static bool HasX86_64 { get; }
```

Property Value

[bool](#) ↗

# Methods

## HasFeature(string?)

Returns `true` if the feature for the given feature tag is supported in the currently running instance, depending on the platform, build etc. Can be used to check whether you're currently running a debug build, on a certain platform or arch, etc. Refer to the [Feature Tags](#) documentation for more details.

Note: Tag names are case-sensitive.

```
public static bool HasFeature(string? tagName)
```

Parameters

tagName [string](#)

Returns

[bool](#)

# Class GDFile

Namespace: [Cobilas.GodotEngine.Utility](#)

Assembly: com.cobilas.godot.utility.dll

```
public class GDFile : GDFileBase, IDisposable
```

## Inheritance

[object](#) ← [GDFileBase](#) ← GDFile

## Implements

[IDisposable](#)

## Inherited Members

[GDFileBase.Null](#) , [object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Properties

## Attribute

```
public override GDFileAttributes Attribute { get; protected set; }
```

## Property Value

[GDFileAttributes](#)

## Name

```
public override string Name { get; }
```

## Property Value

[string](#)

## NameWithoutExtension

```
public override string NameWithoutExtension { get; }
```

Property Value

[string](#) ↗

## Parent

```
public override GDFFileBase Parent { get; protected set; }
```

Property Value

[GDFFileBase](#)

## Path

```
public override string Path { get; protected set; }
```

Property Value

[string](#) ↗

## Methods

### Dispose()

Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources.

```
public override void Dispose()
```

### Dispose(bool)

```
protected virtual void Dispose(bool disposing)
```

Parameters

disposing [bool](#)

~GDFile()

```
protected ~GDFile()
```

Load()

```
public Resource Load()
```

Returns

Resource

Load<T>()

```
public T Load<T>() where T : class
```

Returns

T

Type Parameters

T

Read()

```
public string Read()
```

Returns

string ↴

## Write(byte[]?)

```
public void Write(byte[]? buffer)
```

Parameters

buffer byte ↴ []

# Enum GDFileAttributes

Namespace: [Cobilas.GodotEngine.Utility](#)

Assembly: com.cobilas.godot.utility.dll

```
public enum GDFileAttributes : byte
```

## Fields

Directory = 0

File = 1

Null = 3

# Class GDFileBase

Namespace: [Cobilas.GodotEngine.Utility](#)

Assembly: com.cobilas.godot.utility.dll

```
public abstract class GDFileBase : IDisposable
```

## Inheritance

[object](#) ← GDFileBase

## Implements

[IDisposable](#)

## Derived

[GDDirectory](#), [GDFile](#), [GDIONull](#)

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Properties

## Attribute

```
public abstract GDFileAttributes Attribute { get; protected set; }
```

## Property Value

[GDFileAttributes](#)

## Name

```
public abstract string Name { get; }
```

## Property Value

[string](#) ↗

## NameWithoutExtension

`public abstract string NameWithoutExtension { get; }`

Property Value

[string](#) ↗

## Null

`public static GDFFileBase Null { get; }`

Property Value

[GDFFileBase](#)

## Parent

`public abstract GDFFileBase Parent { get; protected set; }`

Property Value

[GDFFileBase](#)

## Path

`public abstract string Path { get; protected set; }`

Property Value

[string](#) ↗

# Methods

## Dispose()

Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources.

```
public abstract void Dispose()
```

# Class GDIONull

Namespace: [Cobilas.GodotEngine.Utility](#)

Assembly: com.cobilas.godot.utility.dll

```
public sealed class GDIONull : GDFFileBase, IDisposable
```

## Inheritance

[object](#) ← [GDFFileBase](#) ← GDIONull

## Implements

[IDisposable](#)

## Inherited Members

[GDFFileBase.Null](#) , [object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#)

# Properties

## Attribute

```
public override GDFFileAttributes Attribute { get; protected set; }
```

## Property Value

[GDFFileAttributes](#)

## Name

```
public override string Name { get; }
```

## Property Value

[string](#)

## NameWithoutExtension

```
public override string NameWithoutExtension { get; }
```

Property Value

[string](#) ↗

## Parent

```
public override GDFFileBase Parent { get; protected set; }
```

Property Value

[GDFFileBase](#)

## Path

```
public override string Path { get; protected set; }
```

Property Value

[string](#) ↗

## Methods

### Dispose()

Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources.

```
public override void Dispose()
```

# Class Gizmos

Namespace: [Cobilas.GodotEngine.Utility](#)

Assembly: com.cobilas.godot.utility.dll

```
[RunTimeInitializationClass(Priority.StartLater, "Gizmos")]
public class Gizmos : CanvasLayer, IDisposable
```

## Inheritance

[object](#) ← Object ← Node ← CanvasLayer ← Gizmos

## Implements

[IDisposable](#)

## Inherited Members

[CanvasLayer.SetLayer\(int\)](#) , [CanvasLayer.GetLayer\(\)](#) , [CanvasLayer.SetVisible\(bool\)](#) ,  
[CanvasLayer.IsVisible\(\)](#) , [CanvasLayer.Show\(\)](#) , [CanvasLayer.Hide\(\)](#) ,  
[CanvasLayer.SetTransform\(Transform2D\)](#) , [CanvasLayer.GetTransform\(\)](#) , [CanvasLayer.GetFinalTransform\(\)](#) ,  
[CanvasLayer.SetOffset\(Vector2\)](#) , [CanvasLayer.GetOffset\(\)](#) , [CanvasLayer.SetRotation\(float\)](#) ,  
[CanvasLayer.GetRotation\(\)](#) , [CanvasLayer.SetRotationDegrees\(float\)](#) ,  
[CanvasLayer.GetRotationDegrees\(\)](#) , [CanvasLayer.setScale\(Vector2\)](#) , [CanvasLayer.GetScale\(\)](#) ,  
[CanvasLayer.SetFollowViewport\(bool\)](#) , [CanvasLayer.IsFollowingViewport\(\)](#) ,  
[CanvasLayer.SetFollowViewportScale\(float\)](#) , [CanvasLayer.GetFollowViewportScale\(\)](#) ,  
[CanvasLayer.SetCustomViewport\(Node\)](#) , [CanvasLayer.GetCustomViewport\(\)](#) , [CanvasLayer.GetCanvas\(\)](#) ,  
[CanvasLayer.Layer](#) , [CanvasLayer.Visible](#) , [CanvasLayer.Offset](#) , [CanvasLayer.RotationDegrees](#) ,  
[CanvasLayer.Rotation](#) , [CanvasLayer.Scale](#) , [CanvasLayer.Transform](#) , [CanvasLayer.CustomViewport](#) ,  
[CanvasLayer.FollowViewportEnable](#) , [CanvasLayer.FollowViewportScale](#) , [Node.NotificationEnterTree](#) ,  
[Node.NotificationExitTree](#) , [Node.NotificationMovedInParent](#) , [Node.NotificationReady](#) ,  
[Node.NotificationPaused](#) , [Node.NotificationUnpaused](#) , [Node.NotificationPhysicsProcess](#) ,  
[Node.NotificationProcess](#) , [Node.NotificationParented](#) , [Node.NotificationUnparented](#) ,  
[Node.NotificationInstanced](#) , [Node.NotificationDragBegin](#) , [Node.NotificationDragEnd](#) ,  
[Node.NotificationPathChanged](#) , [Node.NotificationInternalProcess](#) ,  
[Node.NotificationInternalPhysicsProcess](#) , [Node.NotificationPostEnterTree](#) ,  
[Node.NotificationResetPhysicsInterpolation](#) , [Node.NotificationWmMouseEnter](#) ,  
[Node.NotificationWmMouseExit](#) , [Node.NotificationWmFocusIn](#) , [Node.NotificationWmFocusOut](#) ,  
[Node.NotificationWmQuitRequest](#) , [Node.NotificationWmGoBackRequest](#) ,  
[Node.NotificationWmUnfocusRequest](#) , [Node.NotificationOsMemoryWarning](#) ,  
[Node.NotificationTranslationChanged](#) , [Node.NotificationWmAbout](#) , [Node.NotificationCrash](#) ,  
[Node.NotificationOslmeUpdate](#) , [Node.NotificationAppResumed](#) , [Node.NotificationAppPaused](#) ,

Node.GetNode<T>(NodePath) , Node.GetNodeOrNull<T>(NodePath) , [Node.GetChild<T>\(int\)](#) ,  
[Node.GetChildOrNull<T>\(int\)](#) , Node.GetOwner<T>() , Node.GetOwnerOrNull<T>() ,  
Node.GetParent<T>() , Node.GetParentOrNull<T>() , Node.\_EnterTree() , Node.\_ExitTree() ,  
Node.\_GetConfigurationWarning() , Node.\_Input(InputEvent) , [Node\\_PhysicsProcess\(float\)](#) ,  
Node.\_UnhandledInput(InputEvent) , Node.\_UnhandledKeyInput(InputEventKey) ,  
[Node.AddChildBelowNode\(Node, Node, bool\)](#) , [Node.SetName\(string\)](#) , Node.GetName() ,  
[Node.AddChild\(Node, bool\)](#) , Node.RemoveChild(Node) , Node.GetChildCount() , Node.GetChildren() ,  
[Node.GetChild\(int\)](#) , Node.HasNode(NodePath) , Node.GetNode(NodePath) ,  
Node.GetNodeOrNull(NodePath) , Node.GetParent() , [Node.FindNode\(string, bool, bool\)](#) ,  
[Node.FindParent\(string\)](#) , Node.HasNodeAndResource(NodePath) ,  
Node.GetNodeAndResource(NodePath) , Node.IsInsideTree() , Node.IsAParentOf(Node) ,  
Node.IsGreater Than(Node) , Node.GetPath() , Node.GetPathTo(Node) ,  
[Node.AddToGroup\(string, bool\)](#) , [Node.RemoveFromGroup\(string\)](#) , [Node.IsInGroup\(string\)](#) ,  
[Node.MoveChild\(Node, int\)](#) , Node.GetGroups() , Node.Raise() , Node.SetOwner(Node) ,  
Node.GetOwner() , Node.RemoveAndSkip() , Node.GetIndex() , Node.PrintTree() , Node.PrintTreePretty() ,  
[Node.SetFilename\(string\)](#) , Node.GetFilename() , [Node.PropagateNotification\(int\)](#) ,  
[Node.PropagateCall\(string, Array, bool\)](#) , [Node.SetPhysicsProcess\(bool\)](#) ,  
Node.GetPhysicsProcessDelta Time() , Node.IsPhysicsProcessing() , Node.GetProcessDelta Time() ,  
[Node.SetProcess\(bool\)](#) , [Node.SetProcessPriority\(int\)](#) , Node.GetProcessPriority() ,  
Node.IsProcessing() , [Node.SetProcessInput\(bool\)](#) , Node.IsProcessingInput() ,  
[Node.SetProcessUnhandledInput\(bool\)](#) , Node.IsProcessingUnhandledInput() ,  
[Node.SetProcessUnhandledKeyInput\(bool\)](#) , Node.IsProcessingUnhandledKeyInput() ,  
Node.SetPauseMode(Node.PauseModeEnum) , Node.GetPauseMode() , Node.CanProcess() ,  
Node.PrintStrayNodes() , NodeGetPositionInParent() , [Node.SetDisplayFolded\(bool\)](#) ,  
Node.IsDisplayedFolded() , [Node.SetProcessInternal\(bool\)](#) , Node.IsProcessingInternal() ,  
[Node.SetPhysicsProcessInternal\(bool\)](#) , Node.IsPhysicsProcessingInternal() ,  
Node.SetPhysicsInterpolationMode(Node.PhysicsInterpolationModeEnum) ,  
Node.GetPhysicsInterpolationMode() , Node.IsPhysicsInterpolated() ,  
Node.IsPhysicsInterpolatedAndEnabled() , Node.ResetPhysicsInterpolation() , Node.GetTree() ,  
Node.CreateTween() , [Node.Duplicate\(int\)](#) , [Node.ReplaceBy\(Node, bool\)](#) ,  
[Node.setSceneInstanceLoadPlaceholder\(bool\)](#) , Node.GetSceneInstanceLoadPlaceholder() ,  
Node.GetViewport() , Node.QueueFree() , Node.RequestReady() , [Node.SetNetworkMaster\(int, bool\)](#) ,  
Node.GetNetworkMaster() , Node.IsNetworkMaster() , Node.GetMultiplayer() ,  
Node.GetCustomMultiplayer() , Node.SetCustomMultiplayer(MultiplayerAPI) ,  
[Node.RpcConfig\(string, MultiplayerAPI.RPCMode\)](#) ,  
[Node.RsetConfig\(string, MultiplayerAPI.RPCMode\)](#) , [Node.SetUniqueNameInOwner\(bool\)](#) ,  
Node.IsUniqueNameInOwner() , [Node.Rpc\(string, params object\[\]\)](#) ,  
[Node.RpcUnreliable\(string, params object\[\]\)](#) , [Node.Rpcld\(int, string, params object\[\]\)](#) ,  
[Node.RpcUnreliableId\(int, string, params object\[\]\)](#) , [Node.Rset\(string, object\)](#) ,  
[Node.RsetId\(int, string, object\)](#) , [Node.RsetUnreliable\(string, object\)](#) ,

[Node.RsetUnreliableId\(int, string, object\)](#) , Node.UpdateConfigurationWarning() ,  
Node.EditorDescription , Node.\_ImportPath , Node.PauseMode , Node.PhysicsInterpolationMode ,  
Node.Name , Node.UniqueNameInOwner , Node.Filename , Node.Owner , Node.Multiplayer ,  
Node.CustomMultiplayer , Node.ProcessPriority , Object.NotificationPostinitialize ,  
Object.NotificationPredelete , Object.IsInstanceValid(Object) , Object.WeakRef(Object) , Object.Dispose() ,  
[Object.Dispose\(bool\)](#) , Object.ToString() , [Object.ToSignal\(Object, string\)](#) , [Object.Get\(string\)](#) ,  
Object.\_GetPropertyList() , [Object.Notification\(int\)](#) , [Object.Set\(string, object\)](#) , Object.Free() ,  
Object.GetClass() , [Object.IsClass\(string\)](#) , [Object.Set\(string, object\)](#) , [Object.Get\(string\)](#) ,  
[Object.SetIndexed\(NodePath, object\)](#) , Object.GetIndexed(NodePath) , Object.GetPropertyList() ,  
Object.GetMethodList() , [Object.Notification\(int, bool\)](#) , Object.GetInstanceId() ,  
Object.SetScript(Reference) , Object.GetScript() , [Object.SetMeta\(string, object\)](#) ,  
[Object.RemoveMeta\(string\)](#) , [Object.GetMeta\(string, object\)](#) , [Object.HasMeta\(string\)](#) ,  
Object.GetMetaList() , [Object.AddUserSignal\(string, Array\)](#) , [Object.HasUserSignal\(string\)](#) ,  
[Object.EmitSignal\(string, params object\[\]\)](#) , [Object.Call\(string, params object\[\]\)](#) ,  
[Object.CallDeferred\(string, params object\[\]\)](#) , [Object.SetDeferred\(string, object\)](#) ,  
[Object.Cally\(string, Array\)](#) , [Object.HasMethod\(string\)](#) , [Object.HasSignal\(string\)](#) ,  
Object.GetSignalList() , [Object.GetSignalConnectionList\(string\)](#) , Object.GetIncomingConnections() ,  
[Object.Connect\(string, Object, string, Array, uint\)](#) , [Object.Disconnect\(string, Object, string\)](#) ,  
[Object.IsConnected\(string, Object, string\)](#) , [Object.SetBlockSignals\(bool\)](#) , Object.IsBlockingSignals() ,  
Object.PropertyListChangedNotify() , [Object.SetMessageTranslation\(bool\)](#) ,  
Object.CanTranslateMessages() , [Object.Tr\(string\)](#) , Object.IsQueuedForDeletion() ,  
Object.NativeInstance , Object.DynamicObject , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Extension Methods

[Node GD CB Extension.FindNodeByName\(Node, string\)](#) ,  
[Node GD CB Extension.FindNodeByName\(Node, string, bool\)](#) ,  
[Node GD CB Extension.FindNodeByName\(Node, string, Type, bool\)](#) ,  
[Node GD CB Extension.FindNodeByName<T>\(Node, string\)](#) ,  
[Node GD CB Extension.FindNodeByName<T>\(Node, string, bool\)](#) ,  
[Node GD CB Extension.FindNodes\(Node, Type\)](#) , [Node GD CB Extension.FindNodes\(Node, Type, bool\)](#) ,  
[Node GD CB Extension.FindNodes<T>\(Node\)](#) , [Node GD CB Extension.FindNodes<T>\(Node, bool\)](#) ,  
[Node GD CB Extension.GetNodePosition\(Node\)](#) , [Node GD CB Extension.GetNodeRotation\(Node\)](#) ,  
[Node GD CB Extension.GetNodeScale\(Node\)](#) , [Node GD CB Extension.Print\(Node, params object\[\]\)](#) ,  
[Node GD CB Extension.SetNodePosition\(Node, Vector3D\)](#) ,  
[Node GD CB Extension.SetNodeRotation\(Node, Vector3D\)](#) ,  
[Node GD CB Extension.SetNodeScale\(Node, Vector3D\)](#)

# Properties

## Color

```
public static Color Color { get; set; }
```

## Property Value

Color

# Methods

## DrawArc(Vector2, float, float, float, int)

```
public static void DrawArc(Vector2 center, float radius, float startAngle, float endAngle,  
int pointCount)
```

## Parameters

center Vector2

radius [float](#)

startAngle [float](#)

endAngle [float](#)

pointCount [int](#)

## DrawArc(Vector2, float, float, float, int, float)

```
public static void DrawArc(Vector2 center, float radius, float startAngle, float endAngle,  
int pointCount, float width)
```

## Parameters

center Vector2

**radius** `float`

**startAngle** `float`

**endAngle** `float`

**pointCount** `int`

**width** `float`

## DrawCircle(Vector2, float)

```
public static void DrawCircle(Vector2 position, float radius)
```

### Parameters

**position** Vector2

**radius** `float`

## DrawLine(Vector2, Vector2)

```
public static void DrawLine(Vector2 start, Vector2 end)
```

### Parameters

**start** Vector2

**end** Vector2

## DrawLine(Vector2, Vector2, float)

```
public static void DrawLine(Vector2 start, Vector2 end, float width)
```

### Parameters

**start** Vector2

```
end Vector2
```

```
width float ↴
```

## DrawMesh(Mesh, Texture, Texture?, Transform2D?)

```
public static void DrawMesh(Mesh mesh, Texture texture, Texture? normalMap = null,  
Transform2D? transform = null)
```

### Parameters

**mesh** Mesh

**texture** Texture

**normalMap** Texture

**transform** Transform2D?

## DrawMultiline(Vector2[])

```
public static void DrawMultiline(Vector2[] points)
```

### Parameters

**points** Vector2[]

## DrawMultiline(Vector2[], float)

```
public static void DrawMultiline(Vector2[] points, float width)
```

### Parameters

**points** Vector2[]

```
width float ↴
```

## DrawMultiline(List<Vector2>)

```
public static void DrawMultiline(List<Vector2> points)
```

### Parameters

points [List](#)<Vector2>

## DrawMultiline(List<Vector2>, float)

```
public static void DrawMultiline(List<Vector2> points, float width)
```

### Parameters

points [List](#)<Vector2>

width [float](#)

## DrawRect(Rect2)

```
public static void DrawRect(Rect2 rect)
```

### Parameters

rect Rect2

## DrawTexture(Texture, Vector2, Texture?)

```
public static void DrawTexture(Texture texture, Vector2 position, Texture? normalMap = null)
```

### Parameters

texture Texture

position Vector2

`normalMap` Texture

## DrawTextureRect(Texture, Rect2, bool, bool, Texture?)

```
public static void DrawTextureRect(Texture texture, Rect2 rect, bool tile, bool transpose =  
false, Texture? normalMap = null)
```

### Parameters

`texture` Texture

`rect` Rect2

`tile` [bool](#)

`transpose` [bool](#)

`normalMap` Texture

## DrawWireRect(Rect2)

```
public static void DrawWireRect(Rect2 rect)
```

### Parameters

`rect` Rect2

## DrawWireRect(Rect2, float)

```
public static void DrawWireRect(Rect2 rect, float width)
```

### Parameters

`rect` Rect2

`width` [float](#)

## \_Process(float)

Called during the processing step of the main loop. Processing happens at every frame and as fast as possible, so the `delta` time since the previous frame is not constant. `delta` is in seconds.

It is only called if processing is enabled, which is done automatically if this method is overridden, and can be toggled with [SetProcess\(bool\)](#).

Corresponds to the Godot.Node.NotificationProcess notification in [Notification\(int\)](#).

Note: This method is only called if the node is present in the scene tree (i.e. if it's not an orphan).

```
public override void _Process(float delta)
```

## Parameters

`delta` [float](#)

## \_Ready()

Called when the node is "ready", i.e. when both the node and its children have entered the scene tree. If the node has children, their Godot.Node.\_Ready() callbacks get triggered first, and the parent node will receive the ready notification afterwards.

Corresponds to the Godot.Node.NotificationReady notification in [Notification\(int\)](#). See also the `onready` keyword for variables.

Usually used for initialization. For even earlier initialization, may be used. See also Godot.Node.\_EnterTree().

Note: Godot.Node.\_Ready() may be called only once for each node. After removing a node from the scene tree and adding it again, `_ready` will not be called a second time. This can be bypassed by requesting another call with Godot.Node.RequestReady(), which may be called anywhere before adding the node again.

```
public override void _Ready()
```

# Interface IYieldCoroutine

Namespace: [Cobilas.GodotEngine.Utility](#)

Assembly: com.cobilas.godot.utility.dll

```
public interface IYieldCoroutine
```

## Properties

### Delay

```
 TimeSpan Delay { get; }
```

Property Value

[TimeSpan](#) ↗

# Interface IYieldFixedUpdate

Namespace: [Cobilas.GodotEngine.Utility](#)

Assembly: com.cobilas.godot.utility.dll

```
public interface IYieldFixedUpdate : IYieldCoroutine
```

## Inherited Members

[IYieldCoroutine.Delay](#)

# Interface IYieldUpdate

Namespace: [Cobilas.GodotEngine.Utility](#)

Assembly: com.cobilas.godot.utility.dll

```
public interface IYieldUpdate : IYieldCoroutine
```

## Inherited Members

[IYieldCoroutine.Delay](#)

# Interface IYieldVolatile

Namespace: [Cobilas.GodotEngine.Utility](#)

Assembly: com.cobilas.godot.utility.dll

```
public interface IYieldVolatile : IYieldCoroutine
```

## Inherited Members

[IYieldCoroutine.Delay](#)

## Properties

### IsPhysicsProcess

```
bool IsPhysicsProcess { get; }
```

Property Value

[bool](#) ↗

# Class Randomico

Namespace: [Cobilas.GodotEngine.Utility](#)

Assembly: com.cobilas.godot.utility.dll

```
public static class Randomico
```

## Inheritance

[object](#) ← Randomico

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Properties

## BooleanRandom

Less than `0.5f` is false, greater than `0.5f` is true. ([Randomico.value > 0.5f](#))

```
public static bool BooleanRandom { get; }
```

## Property Value

[bool](#)

## value

Returns a random number between 0.0 [inclusive] and 1.0 [inclusive] (Read Only).

```
public static double value { get; }
```

## Property Value

[double](#)

# Methods

## ByteList(byte[])

Fills the elements of a specified array of bytes with random numbers.

```
public static void ByteList(byte[] buffer)
```

### Parameters

buffer [byte\[\]](#)

An array of bytes to contain random numbers.

### Exceptions

[ArgumentNullException](#)

buffer is null.

## ByteRange()

Return a random integer number between min [0] and max [255] (ReadOnly).

```
public static byte ByteRange()
```

### Returns

[byte](#)

## ByteRange(byte)

Return a random integer number between min [0] and max [exclusive] (ReadOnly).

```
public static byte ByteRange(byte max)
```

Parameters

max [byte](#)

Returns

[byte](#)

## ByteRange(byte, byte)

Return a random integer number between min [inclusive] and max [exclusive] (ReadOnly).

```
public static byte ByteRange(byte min, byte max)
```

Parameters

min [byte](#)

max [byte](#)

Returns

[byte](#)

## DecimalRange()

Return a random float number between min [-79228162514264337593543950335M] and max [79228162514264337593543950335M] (ReadOnly).

```
public static decimal DecimalRange()
```

Returns

[decimal](#)

## DecimalRange(decimal)

Return a random float number between min [-79228162514264337593543950335M] and max [exclusive] (ReadOnly).

```
public static decimal DecimalRange(decimal max)
```

Parameters

max [decimal](#)

Returns

[decimal](#)

## DecimalRange(decimal, decimal)

Return a random float number between min [inclusive] and max [exclusive] (ReadOnly).

```
public static decimal DecimalRange(decimal min, decimal max)
```

Parameters

min [decimal](#)

max [decimal](#)

Returns

[decimal](#)

## DoubleRange()

Return a random float number between min [-1.7976931348623157E+308] and max [1.7976931348623157E+308] (ReadOnly).

```
public static double DoubleRange()
```

Returns

[double](#)

## DoubleRange(double)

Return a random float number between min [-1.7976931348623157E+308] and max [exclusive] (ReadOnly).

```
public static double DoubleRange(double max)
```

Parameters

max [double](#)

Returns

[double](#)

## DoubleRange(double, double)

Return a random float number between min [inclusive] and max [exclusive] (ReadOnly).

```
public static double DoubleRange(double min, double max)
```

Parameters

min [double](#)

max [double](#)

Returns

[double](#)

## FloatRange()

Return a random float number between min [-3.4028235E+38F] and max [3.4028235E+38F] (ReadOnly).

```
public static float FloatRange()
```

Returns

[float](#)

## FloatRange(float)

Return a random float number between min [-3.4028235E+38F] and max [exclusive] (ReadOnly).

```
public static float FloatRange(float max)
```

Parameters

max [float](#)

Returns

[float](#)

## FloatRange(float, float)

Return a random float number between min [inclusive] and max [exclusive] (ReadOnly).

```
public static float FloatRange(float min, float max)
```

Parameters

min [float](#)

max [float](#)

Returns

[float](#)

## InitSeed(in int)

```
public static void InitSeed(in int seed)
```

### Parameters

seed [int↗](#)

## IntRange()

Return a random integer number between min [-2147483648] and max [2147483647] (ReadOnly).

```
public static int IntRange()
```

### Returns

[int↗](#)

## IntRange(int)

Return a random integer number between min [-2147483648] and max [exclusive] (ReadOnly).

```
public static int IntRange(int max)
```

### Parameters

max [int↗](#)

### Returns

[int↗](#)

## IntRange(int, int)

Return a random integer number between min [inclusive] and max [exclusive] (ReadOnly).

```
public static int IntRange(int min, int max)
```

Parameters

min [int ↗](#)

max [int ↗](#)

Returns

[int ↗](#)

## LongRange()

Return a random integer number between min [-9223372036854775808] and max [9223372036854775807] (ReadOnly).

```
public static long LongRange()
```

Returns

[long ↗](#)

## LongRange(long)

Return a random integer number between min [-9223372036854775808] and max [exclusive] (ReadOnly).

```
public static long LongRange(long max)
```

Parameters

max [long ↗](#)

Returns

[long ↗](#)

## LongRange(long, long)

Return a random integer number between min [inclusive] and max [exclusive] (ReadOnly).

```
public static long LongRange(long min, long max)
```

Parameters

min [long ↗](#)

max [long ↗](#)

Returns

[long ↗](#)

## SByteRange()

Return a random integer number between min [-128] and max [127] (ReadOnly).

```
public static sbyte SByteRange()
```

Returns

[sbyte ↗](#)

## SByteRange(sbyte)

Return a random integer number between min [-128] and max [exclusive] (ReadOnly).

```
public static sbyte SByteRange(sbyte max)
```

Parameters

max [sbyte ↗](#)

Returns

## [sbyte](#)

### SByteRange(sbyte, sbyte)

Return a random integer number between min [inclusive] and max [exclusive] (ReadOnly).

```
public static sbyte SByteRange(sbyte min, sbyte max)
```

#### Parameters

min [sbyte](#)

max [sbyte](#)

#### Returns

[sbyte](#)

### ShortRange()

Return a random integer number between min [-32768] and max [32767] (ReadOnly).

```
public static short ShortRange()
```

#### Returns

[short](#)

### ShortRange(short)

Return a random integer number between min [-32768] and max [exclusive] (ReadOnly).

```
public static short ShortRange(short max)
```

#### Parameters

max [short](#)

Returns

[short ↴](#)

## ShortRange(short, short)

Return a random integer number between min [inclusive] and max [exclusive] (ReadOnly).

```
public static short ShortRange(short min, short max)
```

Parameters

[min short ↴](#)

[max short ↴](#)

Returns

[short ↴](#)

## ULongRange()

Return a random integer number between min [0] and max [18446744073709551615] (ReadOnly).

```
public static ulong ULongRange()
```

Returns

[ulong ↴](#)

## ULongRange(ulong)

Return a random integer number between min [0] and max [exclusive] (ReadOnly).

```
public static ulong ULongRange(ulong max)
```

Parameters

**max** [ulong](#) ↗

Returns

[ulong](#) ↗

## ULongRange(ulong, ulong)

Return a random integer number between min [inclusive] and max [exclusive] (ReadOnly).

```
public static ulong ULongRange(ulong min, ulong max)
```

Parameters

**min** [ulong](#) ↗

**max** [ulong](#) ↗

Returns

[ulong](#) ↗

## UShortRange()

Return a random integer number between min [0] and max [65535] (ReadOnly).

```
public static ushort UShortRange()
```

Returns

[ushort](#) ↗

## UShortRange(ushort)

Return a random integer number between min [0] and max [exclusive] (ReadOnly).

```
public static ushort UShortRange(ushort max)
```

Parameters

max [ushort](#)

Returns

[ushort](#)

## UShortRange(ushort, ushort)

Return a random integer number between min [inclusive] and max [exclusive] (ReadOnly).

```
public static ushort UShortRange(ushort min, ushort max)
```

Parameters

min [ushort](#)

max [ushort](#)

Returns

[ushort](#)

# Struct RunTimeSecond

Namespace: [Cobilas.GodotEngine.Utility](#)

Assembly: com.cobilas.godot.utility.dll

```
public readonly struct RunTimeSecond : IYieldUpdate, IYieldCoroutine
```

Implements

[IYieldUpdate](#), [IYieldCoroutine](#)

Inherited Members

[ValueType.Equals\(object\)](#) , [ValueType.GetHashCode\(\)](#) , [ValueType.ToString\(\)](#) ,  
[object.Equals\(object, object\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.GetType\(\)](#)

## Constructors

### RunTimeSecond(double)

```
public RunTimeSecond(double second)
```

Parameters

second [double](#)

# Class Screen

Namespace: [Cobilas.GodotEngine.Utility](#)

Assembly: com.cobilas.godot.utility.dll

```
public static class Screen
```

## Inheritance

[object](#) ← Screen

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Properties

## CurrentResolution

```
public static Vector2 CurrentResolution { get; }
```

### Property Value

Vector2

## Mode

```
public static ScreenMode Mode { get; set; }
```

### Property Value

[ScreenMode](#)

## Resolutions

```
public static Vector2[] Resolutions { get; }
```

Property Value

Vector2[]

## Methods

AddResolution(int, int)

```
public static void AddResolution(int width, int height)
```

Parameters

width [int](#)

height [int](#)

SetResolution(Vector2)

```
public static void SetResolution(Vector2 size)
```

Parameters

size Vector2

SetResolution(Vector2, ScreenMode)

```
public static void SetResolution(Vector2 size, ScreenMode mode)
```

Parameters

size Vector2

mode [ScreenMode](#)

## SetResolution(int, int)

```
public static void SetResolution(int width, int height)
```

### Parameters

width [int](#)

height [int](#)

## SetResolution(int, int, ScreenMode)

```
public static void SetResolution(int width, int height, ScreenMode mode)
```

### Parameters

width [int](#)

height [int](#)

mode [ScreenMode](#)

# Enum ScreenMode

Namespace: [Cobilas.GodotEngine.Utility](#)

Assembly: com.cobilas.godot.utility.dll

```
public enum ScreenMode : byte
```

## Fields

Borderless = 1

Fullscreen = 2

Resizable = 0

# Namespace Cobilas.GodotEngine.Utility.Input

## Classes

[InputKeyBoard](#)

## Structs

[KeyItem](#)

[MouseItem](#)

## Enums

[KeyStatus](#)

[MouseButton](#)

# Class InputKeyBoard

Namespace: [Cobilas.GodotEngine.Utility.Input](#)

Assembly: com.cobilas.godot.utility.dll

```
[RunTimeInitializationClass("InputKeyBoard")]
public class InputKeyBoard : Node, IDisposable
```

## Inheritance

[object](#) ← Object ← Node ← InputKeyBoard

## Implements

[IDisposable](#)

## Inherited Members

Node.NotificationEnterTree , Node.NotificationExitTree , Node.NotificationMovedInParent ,  
Node.NotificationReady , Node.NotificationPaused , Node.NotificationUnpaused ,  
Node.NotificationPhysicsProcess , Node.NotificationProcess , Node.NotificationParented ,  
Node.NotificationUnparented , Node.NotificationInstanced , Node.NotificationDragBegin ,  
Node.NotificationDragEnd , Node.NotificationPathChanged , Node.NotificationInternalProcess ,  
Node.NotificationInternalPhysicsProcess , Node.NotificationPostEnterTree ,  
Node.NotificationResetPhysicsInterpolation , Node.NotificationWmMouseEnter ,  
Node.NotificationWmMouseExit , Node.NotificationWmFocusIn , Node.NotificationWmFocusOut ,  
Node.NotificationWmQuitRequest , Node.NotificationWmGoBackRequest ,  
Node.NotificationWmUnfocusRequest , Node.NotificationOsMemoryWarning ,  
Node.NotificationTranslationChanged , Node.NotificationWmAbout , Node.NotificationCrash ,  
Node.NotificationOslmeUpdate , Node.NotificationAppResumed , Node.NotificationAppPaused ,  
Node.GetNode<T>(NodePath) , Node.GetNodeOrNull<T>(NodePath) , [Node.GetChild<T>\(int\)](#) ,  
[Node.GetChildOrNull<T>\(int\)](#) , Node.GetOwner<T>() , Node.GetOwnerOrNull<T>() ,  
Node.GetParent<T>() , Node.GetParentOrNull<T>() , Node.\_EnterTree() , Node.\_ExitTree() ,  
Node.\_GetConfigurationWarning() , [Node.Process\(float\)](#) , Node.\_UnhandledInput(InputEvent) ,  
Node.\_UnhandledKeyInput(InputEventKey) , [Node.AddChildBelowNode\(Node, Node, bool\)](#) ,  
[NodeSetName\(string\)](#) , Node.GetName() , [Node.AddChild\(Node, bool\)](#) , Node.RemoveChild(Node) ,  
Node.GetChildCount() , Node.GetChildren() , [Node.GetChild\(int\)](#) , Node.HasNode(NodePath) ,  
Node.GetNode(NodePath) , Node.GetNodeOrNull(NodePath) , Node.GetParent() ,  
[Node.FindNode\(string, bool, bool\)](#) , [Node.FindParent\(string\)](#) ,  
Node.HasNodeAndResource(NodePath) , Node.GetNodeAndResource(NodePath) , Node.IsInsideTree() ,  
Node.IsAParentOf(Node) , Node.IsGreaterThanOrEqual(Node) , Node.GetPath() , Node.GetPathTo(Node) ,  
[Node.AddToGroup\(string, bool\)](#) , [Node.RemoveFromGroup\(string\)](#) , [Node.IsInGroup\(string\)](#) ,

[Node.MoveChild\(Node, int\)](#) , Node.GetGroups() , Node.Raise() , Node.SetOwner(Node) ,  
Node.GetOwner() , Node.RemoveAndSkip() , Node.GetIndex() , Node.PrintTree() , Node.PrintTreePretty() ,  
[Node.SetFilename\(string\)](#) , Node.GetFilename() , [Node.PropagateNotification\(int\)](#) ,  
[Node.PropagateCall\(string, Array, bool\)](#) , [Node.SetPhysicsProcess\(bool\)](#) ,  
Node.GetPhysicsProcessDeltaTime() , Node.IsPhysicsProcessing() , Node.GetProcessDeltaTime() ,  
[Node.SetProcess\(bool\)](#) , [Node.SetProcessPriority\(int\)](#) , Node.GetProcessPriority() ,  
Node.IsProcessing() , [Node.SetProcessInput\(bool\)](#) , Node.IsProcessingInput() ,  
[Node.SetProcessUnhandledInput\(bool\)](#) , Node.IsProcessingUnhandledInput() ,  
[Node.SetProcessUnhandledKeyInput\(bool\)](#) , Node.IsProcessingUnhandledKeyInput() ,  
Node.SetPauseMode(Node.PauseModeEnum) , Node.GetPauseMode() , Node.CanProcess() ,  
Node.PrintStrayNodes() , NodeGetPositionInParent() , [Node.SetDisplayFolded\(bool\)](#) ,  
Node.IsDisplayedFolded() , [Node.SetProcessInternal\(bool\)](#) , Node.IsProcessingInternal() ,  
[Node.SetPhysicsProcessInternal\(bool\)](#) , Node.IsPhysicsProcessingInternal() ,  
Node.SetPhysicsInterpolationMode(Node.PhysicsInterpolationModeEnum) ,  
Node.GetPhysicsInterpolationMode() , Node.IsPhysicsInterpolated() ,  
Node.IsPhysicsInterpolatedAndEnabled() , Node.ResetPhysicsInterpolation() , Node.GetTree() ,  
Node.CreateTween() , [Node.Duplicate\(int\)](#) , [Node.ReplaceBy\(Node, bool\)](#) ,  
[Node.setSceneInstanceLoadPlaceholder\(bool\)](#) , Node.GetSceneInstanceLoadPlaceholder() ,  
Node.GetViewport() , Node.QueueFree() , Node.RequestReady() , [Node.SetNetworkMaster\(int, bool\)](#) ,  
Node.GetNetworkMaster() , Node.IsNetworkMaster() , Node.GetMultiplayer() ,  
Node.GetCustomMultiplayer() , Node.SetCustomMultiplayer(MultiplayerAPI) ,  
[Node.RpcConfig\(string, MultiplayerAPI.RPCMode\)](#) ,  
[Node.RsetConfig\(string, MultiplayerAPI.RPCMode\)](#) , [Node.SetUniqueNameInOwner\(bool\)](#) ,  
Node.IsUniqueNameInOwner() , [Node.Rpc\(string, params object\[\]\)](#) ,  
[Node.RpcUnreliable\(string, params object\[\]\)](#) , [Node.Rpcld\(int, string, params object\[\]\)](#) ,  
[Node.RpcUnreliableId\(int, string, params object\[\]\)](#) , [Node.Rset\(string, object\)](#) ,  
[Node.RsetId\(int, string, object\)](#) , [Node.RsetUnreliable\(string, object\)](#) ,  
[Node.RsetUnreliableId\(int, string, object\)](#) , Node.UpdateConfigurationWarning() ,  
Node.EditorDescription , Node.\_ImportPath , Node.PauseMode , Node.PhysicsInterpolationMode ,  
Node.Name , Node.UniqueNameInOwner , Node.Filename , Node.Owner , Node.Multiplayer ,  
Node.CustomMultiplayer , Node.ProcessPriority , Object.NotificationPostInitialize ,  
Object.NotificationPredelete , Object.IsInstanceValid(Object) , Object.WeakRef(Object) , Object.Dispose() ,  
[Object.Dispose\(bool\)](#) , Object.ToString() , [Object.ToSignal\(Object, string\)](#) , [Object.Get\(string\)](#) ,  
Object.\_GetPropertyList() , [Object.Notification\(int\)](#) , [Object.Set\(string, object\)](#) , Object.Free() ,  
Object.GetClass() , [Object.IsClass\(string\)](#) , [Object.Set\(string, object\)](#) , [Object.Get\(string\)](#) ,  
[Object.SetIndexed\(NodePath, object\)](#) , Object.GetIndexed(NodePath) , Object.GetPropertyList() ,  
Object.GetMethodList() , [Object.Notification\(int, bool\)](#) , Object.GetInstanceId() ,  
Object.SetScript(Reference) , Object.GetScript() , [Object.SetMeta\(string, object\)](#) ,  
[Object.RemoveMeta\(string\)](#) , [Object.GetMeta\(string, object\)](#) , [Object.HasMeta\(string\)](#) ,  
Object.GetMetaList() , [Object.AddUserSignal\(string, Array\)](#) , [Object.HasUserSignal\(string\)](#) ,

[Object.EmitSignal\(string, params object\[\]\)](#) , [Object.Call\(string, params object\[\]\)](#) ,  
[Object.CallDeferred\(string, params object\[\]\)](#) , [Object.SetDeferred\(string, object\)](#) ,  
[Object.Cally\(string, Array\)](#) , [Object.HasMethod\(string\)](#) , [Object.HasSignal\(string\)](#) ,  
Object.GetSignalList() , [Object.GetSignalConnectionList\(string\)](#) , Object.GetIncomingConnections() ,  
[Object.Connect\(string, Object, string, Array, uint\)](#) , [Object.Disconnect\(string, Object, string\)](#) ,  
[Object.IsConnected\(string, Object, string\)](#) , [Object.SetBlockSignals\(bool\)](#) , Object.IsBlockingSignals() ,  
Object.PropertyListChangedNotify() , [Object.SetMessageTranslation\(bool\)](#) ,  
Object.CanTranslateMessages() , [Object.Tr\(string\)](#) , Object.IsQueuedForDeletion() ,  
Object.NativeInstance , Object.DynamicObject , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Extension Methods

[Node GD CB Extension.FindNodeByName\(Node, string\)](#) ,  
[Node GD CB Extension.FindNodeByName\(Node, string, bool\)](#) ,  
[Node GD CB Extension.FindNodeByName\(Node, string, Type, bool\)](#) ,  
[Node GD CB Extension.FindNodeByName<T>\(Node, string\)](#) ,  
[Node GD CB Extension.FindNodeByName<T>\(Node, string, bool\)](#) ,  
[Node GD CB Extension.FindNodes\(Node, Type\)](#) , [Node GD CB Extension.FindNodes\(Node, Type, bool\)](#) ,  
[Node GD CB Extension.FindNodes<T>\(Node\)](#) , [Node GD CB Extension.FindNodes<T>\(Node, bool\)](#) ,  
[Node GD CB Extension.GetNodePosition\(Node\)](#) , [Node GD CB Extension.GetNodeRotation\(Node\)](#) ,  
[Node GD CB Extension.GetNodeScale\(Node\)](#) , [Node GD CB Extension.Print\(Node, params object\[\]\)](#) ,  
[Node GD CB Extension.SetNodePosition\(Node, Vector3D\)](#) ,  
[Node GD CB Extension.SetNodeRotation\(Node, Vector3D\)](#) ,  
[Node GD CB Extension.SetNodeScale\(Node, Vector3D\)](#)

# Properties

## DeltaScroll

```
public static float DeltaScroll { get; }
```

## Property Value

[float](#)

## DoubleClick

```
public static bool DoubleClick { get; }
```

Property Value

[bool](#)

## MouseGlobalPosition

```
public static Vector2 MouseGlobalPosition { get; }
```

Property Value

Vector2

## MouseIndex

```
public static int MouseIndex { get; }
```

Property Value

[int](#)

## MousePosition

```
public static Vector2 MousePosition { get; }
```

Property Value

Vector2

## Methods

GetKeyDown(KeyList)

```
public static bool GetKeyDown(KeyList key)
```

Parameters

**key** KeyList

Returns

bool ↗

## GetKeyPress(KeyList)

```
public static bool GetKeyPress(KeyList key)
```

Parameters

**key** KeyList

Returns

bool ↗

## GetKeyUp(KeyList)

```
public static bool GetKeyUp(KeyList key)
```

Parameters

**key** KeyList

Returns

bool ↗

## GetMouseDown(MouseButton)

```
public static bool GetMouseDown(MouseButton button)
```

Parameters

button [MouseButton](#)

Returns

[bool](#)

## GetMouseDown(int)

```
public static bool GetMouseDown(int buttonIndex)
```

Parameters

buttonIndex [int](#)

Returns

[bool](#)

## GetMousePress(MouseButton)

```
public static bool GetMousePress(MouseButton button)
```

Parameters

button [MouseButton](#)

Returns

[bool](#)

## GetMousePress(int)

```
public static bool GetMousePress(int buttonIndex)
```

Parameters

buttonIndex [int](#)

Returns

[bool](#)

## GetMouseUp(MouseButton)

```
public static bool GetMouseUp(MouseButton button)
```

Parameters

button [MouseButton](#)

Returns

[bool](#)

## GetMouseUp(int)

```
public static bool GetMouseUp(int buttonIndex)
```

Parameters

buttonIndex [int](#)

Returns

[bool](#)

## \_Input(InputEvent)

Called when there is an input event. The input event propagates up through the node tree until a node consumes it.

It is only called if input processing is enabled, which is done automatically if this method is overridden, and can be toggled with [SetProcessInput\(bool\)](#).

To consume the input event and stop it propagating further to other nodes, Godot.SceneTree.SetInputAsHandled() can be called.

For gameplay input, Godot.Node.\_UnhandledInput(Godot.InputEvent) and Godot.Node.\_UnhandledKeyInput(Godot.InputEventKey) are usually a better fit as they allow the GUI to intercept the events first.

Note: This method is only called if the node is present in the scene tree (i.e. if it's not an orphan).

```
public override void _Input(InputEvent @event)
```

## Parameters

**event** InputEvent

## \_PhysicsProcess(float)

Called during the physics processing step of the main loop. Physics processing means that the frame rate is synced to the physics, i.e. the **delta** variable should be constant. **delta** is in seconds.

It is only called if physics processing is enabled, which is done automatically if this method is overridden, and can be toggled with [SetPhysicsProcess\(bool\)](#).

Corresponds to the Godot.Node.NotificationPhysicsProcess notification in [\\_Notification\(int\)](#).

Note: This method is only called if the node is present in the scene tree (i.e. if it's not an orphan).

```
public override void _PhysicsProcess(float delta)
```

## Parameters

**delta** float

## \_Ready()

Called when the node is "ready", i.e. when both the node and its children have entered the scene tree. If the node has children, their Godot.Node.\_Ready() callbacks get triggered first, and the parent node will receive the ready notification afterwards.

Corresponds to the Godot.Node.NotificationReady notification in [\\_Notification\(int\)](#). See also the [onready](#) keyword for variables.

Usually used for initialization. For even earlier initialization, [onready](#) may be used. See also Godot.Node.\_EnterTree().

Note: Godot.Node.\_Ready() may be called only once for each node. After removing a node from the scene tree and adding it again, [\\_ready](#) will not be called a second time. This can be bypassed by requesting another call with Godot.Node.RequestReady(), which may be called anywhere before adding the node again.

```
public override void _Ready()
```

# Struct KeyItem

Namespace: [Cobilas.GodotEngine.Utility.Input](#)

Assembly: com.cobilas.godot.utility.dll

```
public struct KeyItem : IEquatable<KeyItem>
```

## Implements

[IEquatable](#) <[KeyItem](#)>

## Inherited Members

[ValueType.ToString\(\)](#) , [object.Equals\(object, object\)](#) , [object.ReferenceEquals\(object, object\)](#) ,  
[object.GetType\(\)](#)

## Fields

### key

```
public KeyList key
```

#### Field Value

KeyList

### onDestroy

```
public bool onDestroy
```

#### Field Value

[bool](#)

### pressDelay

```
public bool pressDelay
```

Field Value

[bool](#)

status

```
public KeyStatus status
```

Field Value

[KeyStatus](#)

## Properties

Empty

```
public static KeyItem Empty { get; }
```

Property Value

[KeyItem](#)

## Methods

Equals(KeyItem)

Indicates whether the current object is equal to another object of the same type.

```
public bool Equals(KeyItem other)
```

Parameters

## [other](#) [KeyItem](#)

An object to compare with this object.

Returns

[bool](#)

[true](#) if the current object is equal to the [other](#) parameter; otherwise, [false](#).

## Equals(object)

Indicates whether this instance and a specified object are equal.

```
public override bool Equals(object obj)
```

Parameters

[obj](#) [object](#)

The object to compare with the current instance.

Returns

[bool](#)

[true](#) if [obj](#) and this instance are the same type and represent the same value; otherwise, [false](#).

## GetHashCode()

Returns the hash code for this instance.

```
public override int GetHashCode()
```

Returns

[int](#)

A 32-bit signed integer that is the hash code for this instance.

# Operators

## operator ==(KeyItem, KeyItem)

```
public static bool operator ==(KeyItem A, KeyItem B)
```

### Parameters

A [KeyItem](#)

B [KeyItem](#)

### Returns

[bool](#) ↗

## operator !=(KeyItem, KeyItem)

```
public static bool operator !=(KeyItem A, KeyItem B)
```

### Parameters

A [KeyItem](#)

B [KeyItem](#)

### Returns

[bool](#) ↗

# Enum KeyStatus

Namespace: [Cobilas.GodotEngine.Utility.Input](#)

Assembly: com.cobilas.godot.utility.dll

```
public enum KeyStatus
```

## Fields

Down = 3

None = 0

Press = 2

Up = 1

# Enum MouseButton

Namespace: [Cobilas.GodotEngine.Utility.Input](#)

Assembly: com.cobilas.godot.utility.dll

```
public enum MouseButton
```

## Fields

Left = 2

Middle = 3

Right = 1

Unknown = 0

# Struct MouseItem

Namespace: [Cobilas.GodotEngine.Utility.Input](#)

Assembly: com.cobilas.godot.utility.dll

```
public struct MouseItem : IEquatable<MouseItem>
```

## Implements

[IEquatable](#) <[MouseItem](#)>

## Inherited Members

[ValueType.ToString\(\)](#) , [object.Equals\(object, object\)](#) , [object.ReferenceEquals\(object, object\)](#) ,  
[object.GetType\(\)](#)

## Fields

### Index

```
public int Index
```

#### Field Value

[int](#)

### onDestroy

```
public bool onDestroy
```

#### Field Value

[bool](#)

### pressDelay

```
public bool pressDelay
```

Field Value

[bool](#)

status

```
public KeyStatus status
```

Field Value

[KeyStatus](#)

## Properties

Empty

```
public static MouseItem Empty { get; }
```

Property Value

[MouseItem](#)

## Methods

Equals(MouseItem)

Indicates whether the current object is equal to another object of the same type.

```
public bool Equals(MouseItem other)
```

Parameters

## `other` [MouseItem](#)

An object to compare with this object.

Returns

[bool](#)

[true](#) if the current object is equal to the `other` parameter; otherwise, [false](#).

## `Equals(object)`

Indicates whether this instance and a specified object are equal.

```
public override bool Equals(object obj)
```

Parameters

`obj` [object](#)

The object to compare with the current instance.

Returns

[bool](#)

[true](#) if `obj` and this instance are the same type and represent the same value; otherwise, [false](#).

## `GetHashCode()`

Returns the hash code for this instance.

```
public override int GetHashCode()
```

Returns

[int](#)

A 32-bit signed integer that is the hash code for this instance.

# Operators

## operator ==(MouseItem, MouseItem)

```
public static bool operator ==(MouseItem A, MouseItem B)
```

### Parameters

A [MouseItem](#)

B [MouseItem](#)

### Returns

[bool](#) ↗

## operator !=(MouseItem, MouseItem)

```
public static bool operator !=(MouseItem A, MouseItem B)
```

### Parameters

A [MouseItem](#)

B [MouseItem](#)

### Returns

[bool](#) ↗

# Namespace Cobilas.GodotEngine.Utility.Numerics

## Structs

[Quaternion](#)

[Vector2D](#)

[Vector3D](#)

[Vector4D](#)

[VectorEqualityComparer](#)

## Interfaces

[IVector](#)

[IVector<TVector>](#)

# Interface IVector

Namespace: [Cobilas.GodotEngine.Utility.Numerics](#)

Assembly: com.cobilas.godot.utility.dll

```
public interface IVector : IFormattable
```

## Inherited Members

[IFormattable.ToString\(string, IFormatProvider\)](#) ↗

## Properties

### AxisCount

```
int AxisCount { get; }
```

#### Property Value

[int](#) ↗

### this[int]

```
float this[int index] { get; set; }
```

#### Parameters

index [int](#) ↗

#### Property Value

[float](#) ↗

## Normalized

```
IVector Normalized { get; }
```

Property Value

[IVector](#)

aspect

```
float aspect { get; }
```

Property Value

[float](#)

ceil

```
IVector ceil { get; }
```

Property Value

[IVector](#)

floor

```
IVector floor { get; }
```

Property Value

[IVector](#)

magnitude

```
float magnitude { get; }
```

Property Value

[float](#)

sqrMagnitude

```
float sqrMagnitude { get; }
```

Property Value

[float](#)

## Methods

ToString(string)

```
string ToString(string format)
```

Parameters

[format](#) [string](#)

Returns

[string](#)

# Interface IVector<TVector>

Namespace: [Cobilas.GodotEngine.Utility.Numerics](#)

Assembly: com.cobilas.godot.utility.dll

```
public interface IVector<TVector> : IEquatable<TVector>, IVector, IFormattable where TVector : IVector
```

## Type Parameters

TVector

### Inherited Members

[IEquatable<TVector>.Equals\(TVector\)](#) , [IVector.magnitude](#) , [IVector.sqrMagnitude](#) , [IVector.aspect](#) ,  
[IVector.AxisCount](#) , [IVector.this\[int\]](#) , [IVector.ToString\(string\)](#) ,  
[IFormattable.ToString\(string, IFormatProvider\)](#)

## Properties

### Normalized

```
TVector Normalized { get; }
```

### Property Value

TVector

### ceil

```
TVector ceil { get; }
```

### Property Value

TVector

## floor

```
TVector floor { get; }
```

Property Value

TVector

# Struct Quaternion

Namespace: [Cobilas.GodotEngine.Utility.Numerics](#)

Assembly: com.cobilas.godot.utility.dll

```
[Serializable]
public struct Quaternion : IEquatable<Quaternion>, IFormattable
```

Implements

[IEquatable](#) <[Quaternion](#)>, [IFormattable](#)

Inherited Members

[object.Equals\(object, object\)](#), [object.ReferenceEquals\(object, object\)](#), [object.GetType\(\)](#)

## Constructors

### Quaternion(Quaternion)

```
public Quaternion(Quaternion vector)
```

Parameters

vector [Quaternion](#)

### Quaternion(Vector4D)

```
public Quaternion(Vector4D vector)
```

Parameters

vector [Vector4D](#)

### Quaternion(float, float)

```
public Quaternion(float x, float y)
```

Parameters

x [float](#)

y [float](#)

## Quaternion(float, float, float)

```
public Quaternion(float x, float y, float z)
```

Parameters

x [float](#)

y [float](#)

z [float](#)

## Quaternion(float, float, float, float)

```
public Quaternion(float x, float y, float z, float w)
```

Parameters

x [float](#)

y [float](#)

z [float](#)

w [float](#)

# Fields

## Deg2Rad

```
public const double Deg2Rad = 0.017453292519943295
```

Field Value

[double](#)

## KEpsilon

```
public const float KEpsilon = 1E-06
```

Field Value

[float](#)

## Rad2Deg

```
public const double Rad2Deg = 57.29577951308232
```

Field Value

[double](#)

## W

```
public float w
```

Field Value

[float](#)

## X

```
public float x
```

Field Value

[float](#)

y

```
public float y
```

Field Value

[float](#)

z

```
public float z
```

Field Value

[float](#)

## Properties

Euler

```
public readonly Vector3D Euler { get; }
```

Property Value

[Vector3D](#)

Identity

```
public static Quaternion Identity { get; }
```

Property Value

[Quaternion](#)

Normalized

```
public readonly Quaternion Normalized { get; }
```

Property Value

[Quaternion](#)

## Methods

Angle(Quaternion, Quaternion)

```
public static float Angle(Quaternion a, Quaternion b)
```

Parameters

a [Quaternion](#)

b [Quaternion](#)

Returns

[float](#) ↗

Dot(Quaternion, Quaternion)

```
public static float Dot(Quaternion a, Quaternion b)
```

Parameters

a [Quaternion](#)

b [Quaternion](#)

Returns

[float](#)

## Equals(Quaternion)

Indicates whether the current object is equal to another object of the same type.

```
public readonly bool Equals(Quaternion other)
```

Parameters

other [Quaternion](#)

An object to compare with this object.

Returns

[bool](#)

[true](#) if the current object is equal to the other parameter; otherwise, [false](#).

## Equals(object)

Indicates whether this instance and a specified object are equal.

```
public override readonly bool Equals(object obj)
```

Parameters

obj [object](#)

The object to compare with the current instance.

Returns

[bool](#)

[true](#) if [obj](#) and this instance are the same type and represent the same value; otherwise, [false](#).

## GetHashCode()

Returns the hash code for this instance.

```
public override readonly int GetHashCode()
```

Returns

[int](#)

A 32-bit signed integer that is the hash code for this instance.

## Normalize(Quaternion)

```
public static Quaternion Normalize(Quaternion q)
```

Parameters

[q](#) [Quaternion](#)

Returns

[Quaternion](#)

## ToEuler(Quaternion)

```
public static Vector3D ToEuler(Quaternion quaternion)
```

Parameters

## `quaternion` [Quaternion](#)

Returns

[Vector3D](#)

### ToQuaternion(Vector3D)

```
public static Quaternion ToQuaternion(Vector3D vector)
```

Parameters

`vector` [Vector3D](#)

Returns

[Quaternion](#)

### ToString()

Returns the fully qualified type name of this instance.

```
public override readonly string ToString()
```

Returns

[string](#) ↗

The fully qualified type name.

### ToString(string)

```
public readonly string ToString(string format)
```

Parameters

**format** [string](#)

Returns

[string](#)

## ToString(string, IFormatProvider)

Formats the value of the current instance using the specified format.

```
public readonly string ToString(string format, IFormatProvider formatProvider)
```

Parameters

**format** [string](#)

The format to use.-or- A null reference ([Nothing](#) in Visual Basic) to use the default format defined for the type of the [IFormattable](#) implementation.

**formatProvider** [IFormatProvider](#)

The provider to use to format the value.-or- A null reference ([Nothing](#) in Visual Basic) to obtain the numeric format information from the current locale setting of the operating system.

Returns

[string](#)

The value of the current instance in the specified format.

## Operators

### operator ==(Quaternion, Quaternion)

```
public static bool operator ==(Quaternion lhs, Quaternion rhs)
```

Parameters

**lhs** [Quaternion](#)

[rhs](#) [Quaternion](#)

Returns

[bool](#) ↗

## operator !=(Quaternion, Quaternion)

```
public static bool operator !=(Quaternion lhs, Quaternion rhs)
```

Parameters

[lhs](#) [Quaternion](#)

[rhs](#) [Quaternion](#)

Returns

[bool](#) ↗

## operator \*(Quaternion, Quaternion)

```
public static Quaternion operator *(Quaternion lhs, Quaternion rhs)
```

Parameters

[lhs](#) [Quaternion](#)

[rhs](#) [Quaternion](#)

Returns

[Quaternion](#)

## operator \*(Quaternion, Vector3D)

```
public static Vector3D operator *(Quaternion rotation, Vector3D point)
```

Parameters

rotation [Quaternion](#)

point [Vector3D](#)

Returns

[Vector3D](#)

# Struct Vector2D

Namespace: [Cobilas.GodotEngine.Utility.Numerics](#)

Assembly: com.cobilas.godot.utility.dll

```
[Serializable]
public struct Vector2D : IVector<Vector2D>, IEquatable<Vector2D>, IVector, IFormattable
```

## Implements

[IVector<Vector2D>](#), [IEquatable<Vector2D>](#), [IVector](#), [IFormattable](#)

## Inherited Members

[object.Equals\(object, object\)](#), [object.ReferenceEquals\(object, object\)](#), [object.GetType\(\)](#)

## Constructors

### Vector2D(Vector2D)

```
public Vector2D(Vector2D vector)
```

#### Parameters

**vector** [Vector2D](#)

### Vector2D(Vector2)

```
public Vector2D(Vector2 vector)
```

#### Parameters

**vector** [Vector2](#)

### Vector2D(float, float)

```
public Vector2D(float x, float y)
```

## Parameters

x [float](#)

y [float](#)

## Fields

x

```
public float x
```

Field Value

[float](#)

y

```
public float y
```

Field Value

[float](#)

## Properties

AxisCount

```
public readonly int AxisCount { get; }
```

Property Value

[int](#)

## Down

```
public static Vector2D Down { get; }
```

### Property Value

[Vector2D](#)

## this[int]

```
public float this[int index] { readonly get; set; }
```

### Parameters

[index](#) [int](#)

### Property Value

[float](#)

## Left

```
public static Vector2D Left { get; }
```

### Property Value

[Vector2D](#)

## Normalized

```
public readonly Vector2D Normalized { get; }
```

Property Value

[Vector2D](#)

One

```
public static Vector2D One { get; }
```

Property Value

[Vector2D](#)

Right

```
public static Vector2D Right { get; }
```

Property Value

[Vector2D](#)

Up

```
public static Vector2D Up { get; }
```

Property Value

[Vector2D](#)

Zero

```
public static Vector2D Zero { get; }
```

Property Value

## [Vector2D](#)

### aspect

```
public readonly float aspect { get; }
```

#### Property Value

[float](#) ↗

### ceil

```
public readonly Vector2D ceil { get; }
```

#### Property Value

[Vector2D](#)

### floor

```
public readonly Vector2D floor { get; }
```

#### Property Value

[Vector2D](#)

### magnitude

```
public readonly float magnitude { get; }
```

#### Property Value

[float](#) ↗

# sqrMagnitude

```
public readonly float sqrMagnitude { get; }
```

Property Value

[float](#)

## Methods

### Abs(in Vector2D)

```
public static Vector2D Abs(in Vector2D a)
```

Parameters

a [Vector2D](#)

Returns

[Vector2D](#)

### Abs(bool, bool)

```
public readonly Vector2D Abs(bool absX = true, bool absY = true)
```

Parameters

absX [bool](#)

absY [bool](#)

Returns

[Vector2D](#)

## AngleTo(in Vector2D, in Vector2D)

```
public static float AngleTo(in Vector2D lhs, in Vector2D rhs)
```

Parameters

lhs [Vector2D](#)

rhs [Vector2D](#)

Returns

[float](#) ↗

## AngleToPoint(in Vector2D, in Vector2D)

```
public static float AngleToPoint(in Vector2D lhs, in Vector2D rhs)
```

Parameters

lhs [Vector2D](#)

rhs [Vector2D](#)

Returns

[float](#) ↗

## Aspect(in Vector2D)

```
public static float Aspect(in Vector2D a)
```

Parameters

a [Vector2D](#)

Returns

[float](#)

## Ceil(in Vector2D)

```
public static Vector2D Ceil(in Vector2D a)
```

Parameters

a [Vector2D](#)

Returns

[Vector2D](#)

## Cross(in Vector2D, in Vector2D)

```
public static float Cross(in Vector2D lhs, in Vector2D rhs)
```

Parameters

lhs [Vector2D](#)

rhs [Vector2D](#)

Returns

[float](#)

## Distance(in Vector2D, in Vector2D)

```
public static float Distance(in Vector2D a, in Vector2D b)
```

Parameters

a [Vector2D](#)

b [Vector2D](#)

Returns

[float](#)

## Dot(in Vector2D, in Vector2D)

```
public static float Dot(in Vector2D lhs, in Vector2D rhs)
```

Parameters

lhs [Vector2D](#)

rhs [Vector2D](#)

Returns

[float](#)

## Equals(Vector2D)

Indicates whether the current object is equal to another object of the same type.

```
public readonly bool Equals(Vector2D other)
```

Parameters

other [Vector2D](#)

An object to compare with this object.

Returns

[bool](#)

[true](#) if the current object is equal to the **other** parameter; otherwise, [false](#).

## Equals(object)

Indicates whether this instance and a specified object are equal.

```
public override readonly bool Equals(object obj)
```

Parameters

**obj** [object](#)

The object to compare with the current instance.

Returns

[bool](#)

[true](#) if **obj** and this instance are the same type and represent the same value; otherwise, [false](#).

## Floor(in Vector2D)

```
public static Vector2D Floor(in Vector2D a)
```

Parameters

**a** [Vector2D](#)

Returns

[Vector2D](#)

## GetHashCode()

Returns the hash code for this instance.

```
public override readonly int GetHashCode()
```

Returns

[int](#)

A 32-bit signed integer that is the hash code for this instance.

## Magnitude(in Vector2D)

```
public static float Magnitude(in Vector2D a)
```

Parameters

a [Vector2D](#)

Returns

[float](#)

## Max(Vector2D, Vector2D)

```
public static Vector2D Max(Vector2D lhs, Vector2D rhs)
```

Parameters

lhs [Vector2D](#)

rhs [Vector2D](#)

Returns

[Vector2D](#)

## Min(Vector2D, Vector2D)

```
public static Vector2D Min(Vector2D lhs, Vector2D rhs)
```

Parameters

`lhs` [Vector2D](#)

`rhs` [Vector2D](#)

Returns

[Vector2D](#)

## Normalize(in Vector2D)

```
public static Vector2D Normalize(in Vector2D a)
```

Parameters

`a` [Vector2D](#)

Returns

[Vector2D](#)

## SqrMagnitude(in Vector2D)

```
public static float SqrMagnitude(in Vector2D a)
```

Parameters

`a` [Vector2D](#)

Returns

[float](#) ↗

## ToString()

Returns the fully qualified type name of this instance.

```
public override readonly string ToString()
```

Returns

[string](#)

The fully qualified type name.

## ToString(string)

```
public readonly string ToString(string format)
```

Parameters

[format](#) [string](#)

Returns

[string](#)

## ToString(string, IFormatProvider)

Formats the value of the current instance using the specified format.

```
public readonly string ToString(string format, IFormatProvider formatProvider)
```

Parameters

[format](#) [string](#)

The format to use.-or- A null reference ([Nothing](#) in Visual Basic) to use the default format defined for the type of the [IFormattable](#) implementation.

[formatProvider](#) [IFormatProvider](#)

The provider to use to format the value.-or- A null reference ([Nothing](#) in Visual Basic) to obtain the numeric format information from the current locale setting of the operating system.

Returns

[string](#)

The value of the current instance in the specified format.

## Operators

operator +(Vector2D, Vector2D)

```
public static Vector2D operator +(Vector2D a, Vector2D b)
```

Parameters

a [Vector2D](#)

b [Vector2D](#)

Returns

[Vector2D](#)

operator /(Vector2D, Vector2D)

```
public static Vector2D operator /(Vector2D a, Vector2D b)
```

Parameters

a [Vector2D](#)

b [Vector2D](#)

Returns

[Vector2D](#)

operator /(Vector2D, float)

```
public static Vector2D operator /(Vector2D a, float b)
```

Parameters

a [Vector2D](#)

b [float](#)

Returns

[Vector2D](#)

## operator ==(in Vector2D, in Vector2D)

```
public static bool operator ==(in Vector2D lhs, in Vector2D rhs)
```

Parameters

lhs [Vector2D](#)

rhs [Vector2D](#)

Returns

[bool](#)

## implicit operator Vector3D(Vector2D)

```
public static implicit operator Vector3D(Vector2D v)
```

Parameters

v [Vector2D](#)

Returns

[Vector3D](#)

## implicit operator Vector4D(Vector2D)

```
public static implicit operator Vector4D(Vector2D v)
```

Parameters

v [Vector2D](#)

Returns

[Vector4D](#)

## implicit operator Vector2(Vector2D)

```
public static implicit operator Vector2(Vector2D v)
```

Parameters

v [Vector2D](#)

Returns

[Vector2](#)

## implicit operator Vector3(Vector2D)

```
public static implicit operator Vector3(Vector2D v)
```

Parameters

v [Vector2D](#)

Returns

Vector3

## implicit operator Vector2D(Vector2)

```
public static implicit operator Vector2D(Vector2 v)
```

Parameters

v Vector2

Returns

[Vector2D](#)

## implicit operator Vector2D(Vector3)

```
public static implicit operator Vector2D(Vector3 v)
```

Parameters

v Vector3

Returns

[Vector2D](#)

## operator !=(in Vector2D, in Vector2D)

```
public static bool operator !=(in Vector2D lhs, in Vector2D rhs)
```

Parameters

lhs [Vector2D](#)

rhs [Vector2D](#)

Returns

[bool](#)

## operator \*(Vector2D, Vector2D)

```
public static Vector2D operator *(Vector2D a, Vector2D b)
```

Parameters

a [Vector2D](#)

b [Vector2D](#)

Returns

[Vector2D](#)

## operator \*(Vector2D, float)

```
public static Vector2D operator *(Vector2D a, float b)
```

Parameters

a [Vector2D](#)

b [float](#)

Returns

[Vector2D](#)

## operator -(Vector2D, Vector2D)

```
public static Vector2D operator -(Vector2D a, Vector2D b)
```

## Parameters

a [Vector2D](#)

b [Vector2D](#)

## Returns

[Vector2D](#)

# Struct Vector3D

Namespace: [Cobilas.GodotEngine.Utility.Numerics](#)

Assembly: com.cobilas.godot.utility.dll

```
[Serializable]
public struct Vector3D : IVector<Vector3D>, IEquatable<Vector3D>, IVector, IFormattable
```

## Implements

[IVector<Vector3D>](#), [IEquatable<Vector3D>](#), [IVector](#), [IFormattable](#)

## Inherited Members

[object.Equals\(object, object\)](#), [object.ReferenceEquals\(object, object\)](#), [object.GetType\(\)](#)

## Constructors

### Vector3D(Vector3D)

```
public Vector3D(Vector3D vector)
```

#### Parameters

**vector** [Vector3D](#)

### Vector3D(Vector3)

```
public Vector3D(Vector3 vector)
```

#### Parameters

**vector** [Vector3](#)

### Vector3D(float, float)

```
public Vector3D(float x, float y)
```

## Parameters

x [float](#)

y [float](#)

## Vector3D(float, float, float)

```
public Vector3D(float x, float y, float z)
```

## Parameters

x [float](#)

y [float](#)

z [float](#)

## Fields

### X

```
public float x
```

### Field Value

[float](#)

### y

```
public float y
```

### Field Value

[float](#) ↴

Z

```
public float z
```

Field Value

[float](#) ↴

## Properties

AxisCount

```
public readonly int AxisCount { get; }
```

Property Value

[int](#) ↴

Back

```
public static Vector3D Back { get; }
```

Property Value

[Vector3D](#)

Down

```
public static Vector3D Down { get; }
```

Property Value

[Vector3D](#)

## Forward

```
public static Vector3D Forward { get; }
```

Property Value

[Vector3D](#)

## this[int]

```
public float this[int index] { readonly get; set; }
```

Parameters

index [int](#)

Property Value

[float](#)

## Left

```
public static Vector3D Left { get; }
```

Property Value

[Vector3D](#)

## Normalized

```
public readonly Vector3D Normalized { get; }
```

Property Value

[Vector3D](#)

One

```
public static Vector3D One { get; }
```

Property Value

[Vector3D](#)

Right

```
public static Vector3D Right { get; }
```

Property Value

[Vector3D](#)

Up

```
public static Vector3D Up { get; }
```

Property Value

[Vector3D](#)

Zero

```
public static Vector3D Zero { get; }
```

Property Value

## [Vector3D](#)

### ceil

```
public readonly Vector3D ceil { get; }
```

Property Value

## [Vector3D](#)

### floor

```
public readonly Vector3D floor { get; }
```

Property Value

## [Vector3D](#)

### magnitude

```
public readonly float magnitude { get; }
```

Property Value

## [float](#) ↗

### sqrMagnitude

```
public readonly float sqrMagnitude { get; }
```

Property Value

## [float](#) ↗

# Methods

## Abs(in Vector3D)

```
public static Vector3D Abs(in Vector3D a)
```

### Parameters

a [Vector3D](#)

### Returns

[Vector3D](#)

## Abs(bool, bool, bool)

```
public readonly Vector3D Abs(bool absX = true, bool absY = true, bool absZ = true)
```

### Parameters

absX [bool](#)

absY [bool](#)

absZ [bool](#)

### Returns

[Vector3D](#)

## AngleTo(in Vector2D, in Vector2D)

```
public static float AngleTo(in Vector2D lhs, in Vector2D rhs)
```

### Parameters

lhs [Vector2D](#)

`rhs` [Vector2D](#)

Returns

[float](#) ↗

## Ceil(in Vector3D)

```
public static Vector3D Ceil(in Vector3D a)
```

Parameters

`a` [Vector3D](#)

Returns

[Vector3D](#)

## Cross(in Vector3D, in Vector3D)

```
public static Vector3D Cross(in Vector3D lhs, in Vector3D rhs)
```

Parameters

`lhs` [Vector3D](#)

`rhs` [Vector3D](#)

Returns

[Vector3D](#)

## Distance(in Vector3D, in Vector3D)

```
public static float Distance(in Vector3D a, in Vector3D b)
```

Parameters

a [Vector3D](#)

b [Vector3D](#)

Returns

[float](#) ↗

## Dot(in Vector3D, in Vector3D)

```
public static float Dot(in Vector3D lhs, in Vector3D rhs)
```

Parameters

lhs [Vector3D](#)

rhs [Vector3D](#)

Returns

[float](#) ↗

## Equals(Vector3D)

Indicates whether the current object is equal to another object of the same type.

```
public readonly bool Equals(Vector3D other)
```

Parameters

other [Vector3D](#)

An object to compare with this object.

Returns

[bool](#) ↗

`true` if the current object is equal to the `other` parameter; otherwise, `false`.

## Equals(object)

Indicates whether this instance and a specified object are equal.

```
public override readonly bool Equals(object obj)
```

Parameters

`obj` [object](#)

The object to compare with the current instance.

Returns

`bool`

`true` if `obj` and this instance are the same type and represent the same value; otherwise, `false`.

## Floor(in Vector3D)

```
public static Vector3D Floor(in Vector3D a)
```

Parameters

`a` [Vector3D](#)

Returns

[Vector3D](#)

## GetHashCode()

Returns the hash code for this instance.

```
public override readonly int GetHashCode()
```

Returns

[int](#)

A 32-bit signed integer that is the hash code for this instance.

## Magnitude(in Vector3D)

```
public static float Magnitude(in Vector3D a)
```

Parameters

a [Vector3D](#)

Returns

[float](#)

## Max(Vector3D, Vector3D)

```
public static Vector3D Max(Vector3D lhs, Vector3D rhs)
```

Parameters

lhs [Vector3D](#)

rhs [Vector3D](#)

Returns

[Vector3D](#)

## Min(Vector3D, Vector3D)

```
public static Vector3D Min(Vector3D lhs, Vector3D rhs)
```

Parameters

lhs [Vector3D](#)

rhs [Vector3D](#)

Returns

[Vector3D](#)

## Normalize(in Vector3D)

```
public static Vector3D Normalize(in Vector3D a)
```

Parameters

a [Vector3D](#)

Returns

[Vector3D](#)

## SqrMagnitude(in Vector3D)

```
public static float SqrMagnitude(in Vector3D a)
```

Parameters

a [Vector3D](#)

Returns

[float](#) ↗

## ToString()

Returns the fully qualified type name of this instance.

```
public override readonly string ToString()
```

Returns

[string](#)

The fully qualified type name.

## ToString(string)

```
public readonly string ToString(string format)
```

Parameters

[format](#) [string](#)

Returns

[string](#)

## ToString(string, IFormatProvider)

Formats the value of the current instance using the specified format.

```
public readonly string ToString(string format, IFormatProvider formatProvider)
```

Parameters

[format](#) [string](#)

The format to use.-or- A null reference ([Nothing](#) in Visual Basic) to use the default format defined for the type of the [IFormattable](#) implementation.

[formatProvider](#) [IFormatProvider](#)

The provider to use to format the value.-or- A null reference ([Nothing](#) in Visual Basic) to obtain the numeric format information from the current locale setting of the operating system.

Returns

[string](#)

The value of the current instance in the specified format.

## Operators

operator +(Vector3D, Vector3D)

```
public static Vector3D operator +(Vector3D a, Vector3D b)
```

Parameters

a [Vector3D](#)

b [Vector3D](#)

Returns

[Vector3D](#)

operator /(Vector3D, Vector3D)

```
public static Vector3D operator /(Vector3D a, Vector3D b)
```

Parameters

a [Vector3D](#)

b [Vector3D](#)

Returns

[Vector3D](#)

operator /(Vector3D, float)

```
public static Vector3D operator /(Vector3D a, float b)
```

Parameters

a [Vector3D](#)

b [float](#)

Returns

[Vector3D](#)

## operator ==(in Vector3D, in Vector3D)

```
public static bool operator ==(in Vector3D lhs, in Vector3D rhs)
```

Parameters

lhs [Vector3D](#)

rhs [Vector3D](#)

Returns

[bool](#)

## implicit operator Vector2D(Vector3D)

```
public static implicit operator Vector2D(Vector3D v)
```

Parameters

v [Vector3D](#)

Returns

[Vector2D](#)

## implicit operator Vector4D(Vector3D)

```
public static implicit operator Vector4D(Vector3D v)
```

Parameters

v [Vector3D](#)

Returns

[Vector4D](#)

## implicit operator Vector2(Vector3D)

```
public static implicit operator Vector2(Vector3D v)
```

Parameters

v [Vector3D](#)

Returns

Vector2

## implicit operator Vector3(Vector3D)

```
public static implicit operator Vector3(Vector3D v)
```

Parameters

v [Vector3D](#)

Returns

## implicit operator Vector3D(Vector2)

```
public static implicit operator Vector3D(Vector2 v)
```

### Parameters

v Vector2

### Returns

[Vector3D](#)

## implicit operator Vector3D(Vector3)

```
public static implicit operator Vector3D(Vector3 v)
```

### Parameters

v Vector3

### Returns

[Vector3D](#)

## operator !=(in Vector3D, in Vector3D)

```
public static bool operator !=(in Vector3D lhs, in Vector3D rhs)
```

### Parameters

lhs [Vector3D](#)

rhs [Vector3D](#)

Returns

[bool](#)

## operator \*(Vector3D, Vector3D)

```
public static Vector3D operator *(Vector3D a, Vector3D b)
```

Parameters

a [Vector3D](#)

b [Vector3D](#)

Returns

[Vector3D](#)

## operator \*(Vector3D, float)

```
public static Vector3D operator *(Vector3D a, float b)
```

Parameters

a [Vector3D](#)

b [float](#)

Returns

[Vector3D](#)

## operator -(Vector3D, Vector3D)

```
public static Vector3D operator -(Vector3D a, Vector3D b)
```

## Parameters

a [Vector3D](#)

b [Vector3D](#)

## Returns

[Vector3D](#)

# Struct Vector4D

Namespace: [Cobilas.GodotEngine.Utility.Numerics](#)

Assembly: com.cobilas.godot.utility.dll

```
[Serializable]
public struct Vector4D : IVector<Vector4D>, IEquatable<Vector4D>, IVector, IFormattable
```

## Implements

[IVector<Vector4D>](#), [IEquatable<Vector4D>](#), [IVector](#), [IFormattable](#)

## Inherited Members

[object.Equals\(object, object\)](#), [object.ReferenceEquals\(object, object\)](#), [object.GetType\(\)](#)

## Constructors

### Vector4D(Quaternion)

```
public Vector4D(Quaternion vector)
```

#### Parameters

vector [Quaternion](#)

### Vector4D(Vector4D)

```
public Vector4D(Vector4D vector)
```

#### Parameters

vector [Vector4D](#)

### Vector4D(float, float)

```
public Vector4D(float x, float y)
```

Parameters

x [float](#)

y [float](#)

## Vector4D(float, float, float)

```
public Vector4D(float x, float y, float z)
```

Parameters

x [float](#)

y [float](#)

z [float](#)

## Vector4D(float, float, float, float)

```
public Vector4D(float x, float y, float z, float w)
```

Parameters

x [float](#)

y [float](#)

z [float](#)

w [float](#)

# Fields

w

```
public float w
```

Field Value

[float ↗](#)

x

```
public float x
```

Field Value

[float ↗](#)

y

```
public float y
```

Field Value

[float ↗](#)

z

```
public float z
```

Field Value

[float ↗](#)

## Properties

## AxisCount

```
public readonly int AxisCount { get; }
```

### Property Value

[int](#)

## this[int]

```
public float this[int index] { readonly get; set; }
```

### Parameters

[index](#) [int](#)

### Property Value

[float](#)

## Normalized

```
public readonly Vector4D Normalized { get; }
```

### Property Value

[Vector4D](#)

## One

```
public static Vector4D One { get; }
```

### Property Value

[Vector4D](#)

## Zero

```
public static Vector4D Zero { get; }
```

### Property Value

[Vector4D](#)

## ceil

```
public readonly Vector4D ceil { get; }
```

### Property Value

[Vector4D](#)

## floor

```
public readonly Vector4D floor { get; }
```

### Property Value

[Vector4D](#)

## magnitude

```
public readonly float magnitude { get; }
```

### Property Value

[float](#) ↗

# sqrMagnitude

```
public readonly float sqrMagnitude { get; }
```

Property Value

[float](#)

## Methods

### Abs(Vector4D)

```
public static Vector4D Abs(Vector4D a)
```

Parameters

a [Vector4D](#)

Returns

[Vector4D](#)

### Abs(bool, bool, bool, bool)

```
public readonly Vector3D Abs(bool absX = true, bool absY = true, bool absZ = true, bool absW = true)
```

Parameters

absX [bool](#)

absY [bool](#)

absZ [bool](#)

absW [bool](#)

Returns

[Vector3D](#)

## Ceil(in Vector4D)

```
public static Vector4D Ceil(in Vector4D a)
```

Parameters

a [Vector4D](#)

Returns

[Vector4D](#)

## Distance(Vector4D, Vector4D)

```
public static float Distance(Vector4D a, Vector4D b)
```

Parameters

a [Vector4D](#)

b [Vector4D](#)

Returns

[float](#)

## Dot(Vector4D, Vector4D)

```
public static float Dot(Vector4D a, Vector4D b)
```

Parameters

a [Vector4D](#)

b [Vector4D](#)

Returns

[float](#)

## Equals(Vector4D)

Indicates whether the current object is equal to another object of the same type.

```
public readonly bool Equals(Vector4D other)
```

Parameters

other [Vector4D](#)

An object to compare with this object.

Returns

[bool](#)

[true](#) if the current object is equal to the other parameter; otherwise, [false](#).

## Equals(object)

Indicates whether this instance and a specified object are equal.

```
public override readonly bool Equals(object obj)
```

Parameters

obj [object](#)

The object to compare with the current instance.

Returns

[bool](#)

[true](#) if [obj](#) and this instance are the same type and represent the same value; otherwise, [false](#).

## Floor(in Vector3D)

```
public static Vector3D Floor(in Vector3D a)
```

Parameters

a [Vector3D](#)

Returns

[Vector3D](#)

## GetHashCode()

Returns the hash code for this instance.

```
public override readonly int GetHashCode()
```

Returns

[int](#)

A 32-bit signed integer that is the hash code for this instance.

## IsNormalized(IVector)

```
public static bool IsNormalized(IVector a)
```

Parameters

a [IVector](#)

Returns

[bool](#)

## Magnitude(Vector4D)

```
public static float Magnitude(Vector4D a)
```

Parameters

a [Vector4D](#)

Returns

[float](#)

## Max(Vector4D, Vector4D)

```
public static Vector4D Max(Vector4D lhs, Vector4D rhs)
```

Parameters

lhs [Vector4D](#)

rhs [Vector4D](#)

Returns

[Vector4D](#)

## Min(Vector4D, Vector4D)

```
public static Vector4D Min(Vector4D lhs, Vector4D rhs)
```

Parameters

`lhs` [Vector4D](#)

`rhs` [Vector4D](#)

Returns

[Vector4D](#)

## Normalize(Vector4D)

```
public static Vector4D Normalize(Vector4D a)
```

Parameters

`a` [Vector4D](#)

Returns

[Vector4D](#)

## SqrMagnitude(Vector4D)

```
public static float SqrMagnitude(Vector4D a)
```

Parameters

`a` [Vector4D](#)

Returns

[float](#) ↗

## ToString()

Returns the fully qualified type name of this instance.

```
public override readonly string ToString()
```

Returns

[string](#)

The fully qualified type name.

## ToString(string)

```
public readonly string ToString(string format)
```

Parameters

[format](#) [string](#)

Returns

[string](#)

## ToString(string, IFormatProvider)

Formats the value of the current instance using the specified format.

```
public readonly string ToString(string format, IFormatProvider formatProvider)
```

Parameters

[format](#) [string](#)

The format to use.-or- A null reference ([Nothing](#) in Visual Basic) to use the default format defined for the type of the [IFormattable](#) implementation.

[formatProvider](#) [IFormatProvider](#)

The provider to use to format the value.-or- A null reference ([Nothing](#) in Visual Basic) to obtain the numeric format information from the current locale setting of the operating system.

Returns

[string](#)

The value of the current instance in the specified format.

## Operators

operator +(Vector4D, Vector4D)

```
public static Vector4D operator +(Vector4D a, Vector4D b)
```

Parameters

a [Vector4D](#)

b [Vector4D](#)

Returns

[Vector4D](#)

operator /(Vector4D, Vector4D)

```
public static Vector4D operator /(Vector4D a, Vector4D b)
```

Parameters

a [Vector4D](#)

b [Vector4D](#)

Returns

[Vector4D](#)

operator /(Vector4D, float)

```
public static Vector4D operator /(Vector4D a, float b)
```

Parameters

a [Vector4D](#)

b [float](#)

Returns

[Vector4D](#)

## operator /(float, Vector4D)

```
public static Vector4D operator /(float a, Vector4D b)
```

Parameters

a [float](#)

b [Vector4D](#)

Returns

[Vector4D](#)

## implicit operator Vector2D(Vector4D)

```
public static implicit operator Vector2D(Vector4D v)
```

Parameters

v [Vector4D](#)

Returns

[Vector2D](#)

## implicit operator Vector3D(Vector4D)

```
public static implicit operator Vector3D(Vector4D v)
```

Parameters

v [Vector4D](#)

Returns

[Vector3D](#)

## implicit operator Vector2(Vector4D)

```
public static implicit operator Vector2(Vector4D v)
```

Parameters

v [Vector4D](#)

Returns

[Vector2](#)

## implicit operator Vector3(Vector4D)

```
public static implicit operator Vector3(Vector4D v)
```

Parameters

v [Vector4D](#)

Returns

Vector3

## implicit operator Vector4D(Vector2)

```
public static implicit operator Vector4D(Vector2 v)
```

Parameters

v Vector2

Returns

[Vector4D](#)

## implicit operator Vector4D(Vector3)

```
public static implicit operator Vector4D(Vector3 v)
```

Parameters

v Vector3

Returns

[Vector4D](#)

## operator \*(Vector4D, Vector4D)

```
public static Vector4D operator *(Vector4D a, Vector4D b)
```

Parameters

a [Vector4D](#)

b [Vector4D](#)

Returns

[Vector4D](#)

## operator \*(Vector4D, float)

```
public static Vector4D operator *(Vector4D a, float b)
```

Parameters

a [Vector4D](#)

b [float](#)

Returns

[Vector4D](#)

## operator \*(float, Vector4D)

```
public static Vector4D operator *(float a, Vector4D b)
```

Parameters

a [float](#)

b [Vector4D](#)

Returns

[Vector4D](#)

## operator -(Vector4D, Vector4D)

```
public static Vector4D operator -(Vector4D a, Vector4D b)
```

## Parameters

a [Vector4D](#)

b [Vector4D](#)

## Returns

[Vector4D](#)

# Struct VectorEqualityComparer

Namespace: [Cobilas.GodotEngine.Utility.Numerics](#)

Assembly: com.cobilas.godot.utility.dll

```
public struct VectorEqualityComparer : IEqualityComparer, IEqualityComparer<Vector2D>,  
IEqualityComparer<Vector3D>, IEqualityComparer<Vector4D>
```

## Implements

[IEqualityComparer](#), [IEqualityComparer<Vector2D>](#), [IEqualityComparer<Vector3D>](#),  
[IEqualityComparer<Vector4D>](#)

## Inherited Members

[ValueType.Equals\(object\)](#), [ValueType.GetHashCode\(\)](#), [ValueType.ToString\(\)](#),  
[object.ReferenceEquals\(object, object\)](#), [object.GetType\(\)](#)

## Methods

### Equals(Vector2D, Vector2D)

Determines whether the specified objects are equal.

```
public readonly bool Equals(Vector2D x, Vector2D y)
```

#### Parameters

x [Vector2D](#)

The first object of type T to compare.

y [Vector2D](#)

The second object of type T to compare.

#### Returns

[bool](#)

[true](#) if the specified objects are equal; otherwise, [false](#).

## Equals(Vector3D, Vector3D)

Determines whether the specified objects are equal.

```
public readonly bool Equals(Vector3D x, Vector3D y)
```

Parameters

x [Vector3D](#)

The first object of type T to compare.

y [Vector3D](#)

The second object of type T to compare.

Returns

[bool](#)

[true](#) if the specified objects are equal; otherwise, [false](#).

## Equals(Vector4D, Vector4D)

Determines whether the specified objects are equal.

```
public readonly bool Equals(Vector4D x, Vector4D y)
```

Parameters

x [Vector4D](#)

The first object of type T to compare.

y [Vector4D](#)

The second object of type T to compare.

Returns

[bool](#)

[true](#) if the specified objects are equal; otherwise, [false](#).

## Equals(object, object)

Determines whether the specified objects are equal.

```
public readonly bool Equals(object x, object y)
```

### Parameters

*x* [object](#)

The first object to compare.

*y* [object](#)

The second object to compare.

### Returns

[bool](#)

[true](#) if the specified objects are equal; otherwise, [false](#).

### Exceptions

[ArgumentException](#)

*x* and *y* are of different types and neither one can handle comparisons with the other.

## GetHashCode(Vector2D)

Returns a hash code for the specified object.

```
public readonly int GetHashCode(Vector2D obj)
```

### Parameters

*obj* [Vector2D](#)

The [object](#) for which a hash code is to be returned.

Returns

[int](#)

A hash code for the specified object.

Exceptions

[ArgumentNullException](#)

The type of [obj](#) is a reference type and [obj](#) is [null](#).

## GetHashCode(Vector3D)

Returns a hash code for the specified object.

```
public readonly int GetHashCode(Vector3D obj)
```

Parameters

[obj](#) [Vector3D](#)

The [object](#) for which a hash code is to be returned.

Returns

[int](#)

A hash code for the specified object.

Exceptions

[ArgumentNullException](#)

The type of [obj](#) is a reference type and [obj](#) is [null](#).

## GetHashCode(Vector4D)

Returns a hash code for the specified object.

```
public readonly int GetHashCode(Vector4D obj)
```

Parameters

**obj** [Vector4D](#)

The [object](#) for which a hash code is to be returned.

Returns

[int](#)

A hash code for the specified object.

Exceptions

[ArgumentNullException](#)

The type of **obj** is a reference type and **obj** is [null](#).

## GetHashCode(object)

Returns a hash code for the specified object.

```
public readonly int GetHashCode(object obj)
```

Parameters

**obj** [object](#)

The [object](#) for which a hash code is to be returned.

Returns

[int](#)

A hash code for the specified object.

## Exceptions

### [ArgumentNullException](#)

The type of `obj` is a reference type and `obj` is [null](#).

# Namespace Cobilas.GodotEngine.Utility.Physics

## Classes

[Physics2D](#)

## Structs

[Hit2D](#)

[RayHit2D](#)

# Struct Hit2D

Namespace: [Cobilas.GodotEngine.Utility.Physics](#)

Assembly: com.cobilas.godot.utility.dll

```
public struct Hit2D
```

## Inherited Members

[ValueType.Equals\(object\)](#) , [ValueType.GetHashCode\(\)](#) , [ValueType.ToString\(\)](#) ,  
[object.Equals\(object, object\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.GetType\(\)](#)

## Properties

### Collision

```
public readonly Node Collision { get; }
```

### Property Value

Node

### ID

```
public readonly int ID { get; }
```

### Property Value

[int](#)

### MetaData

```
public readonly object MetaData { get; }
```

Property Value

[object](#)

Name

```
public readonly string Name { get; }
```

Property Value

[string](#)

RID

```
public readonly RID RID { get; }
```

Property Value

RID

## Operators

explicit operator Hit2D(Dictionary?)

```
public static explicit operator Hit2D(Dictionary? D)
```

Parameters

D Dictionary

Returns

[Hit2D](#)

# Class Physics2D

Namespace: [Cobilas.GodotEngine.Utility.Physics](#)

Assembly: com.cobilas.godot.utility.dll

```
[RunTimeInitializationClass("Physics2D")]
public class Physics2D : Node2D, IDisposable
```

## Inheritance

[object](#) ← Object ← Node ← CanvasItem ← Node2D ← Physics2D

## Implements

[IDisposable](#)

## Inherited Members

Node2D.SetPosition(Vector2) , [Node2D.SetRotation\(float\)](#) , [Node2D.SetRotationDegrees\(float\)](#) ,  
Node2D.setScale(Vector2) , Node2D.GetPosition() , Node2D.GetRotation() ,  
Node2D.GetRotationDegrees() , Node2D.GetScale() , [Node2D.Rotate\(float\)](#) ,  
[Node2D.MoveLocalX\(float, bool\)](#) , [Node2D.MoveLocalY\(float, bool\)](#) , Node2D.Translate(Vector2) ,  
Node2D.GlobalTranslate(Vector2) , Node2D.ApplyScale(Vector2) , Node2D.SetGlobalPosition(Vector2) ,  
Node2D.GetGlobalPosition() , [Node2D.SetGlobalRotation\(float\)](#) , Node2D.GetGlobalRotation() ,  
[Node2D.SetGlobalRotationDegrees\(float\)](#) , Node2D.GetGlobalRotationDegrees() ,  
Node2D.SetGlobalScale(Vector2) , Node2D.GetGlobalScale() , Node2D.SetTransform(Transform2D) ,  
Node2D.SetGlobalTransform(Transform2D) , Node2D.LookAt(Vector2) , Node2D.GetAngleTo(Vector2) ,  
Node2DToLocal(Vector2) , Node2D.ToGlobal(Vector2) , [Node2D.SetZIndex\(int\)](#) , Node2D.GetZIndex() ,  
[Node2D.SetZAsRelative\(bool\)](#) , Node2D.IsZRelative() , Node2D.GetRelativeTransformToParent(Node) ,  
Node2D.Position , Node2D.Rotation , Node2D.RotationDegrees , Node2D.Scale , Node2D.Transform ,  
Node2D.GlobalPosition , Node2D.GlobalRotation , Node2D.GlobalRotationDegrees ,  
Node2D.GlobalScale , Node2D.GlobalTransform , Node2D.ZIndex , Node2D.ZAsRelative ,  
CanvasItem.NotificationTransformChanged , CanvasItem.NotificationLocalTransformChanged ,  
CanvasItem.NotificationDraw , CanvasItem.NotificationVisibilityChanged ,  
CanvasItem.NotificationEnterCanvas , CanvasItem.NotificationExitCanvas , CanvasItem.\_Draw() ,  
CanvasItem.GetCanvasItem() , [CanvasItem.SetVisible\(bool\)](#) , CanvasItem.isVisible() ,  
CanvasItem.isVisibleInTree() , CanvasItem.Show() , CanvasItem.Hide() , CanvasItem.Update() ,  
[CanvasItem.SetAsToplevel\(bool\)](#) , CanvasItem.IsSetAsToplevel() , [CanvasItem.SetLightMask\(int\)](#) ,  
CanvasItem.GetLightMask() , CanvasItem.SetModulate(Color) , CanvasItem.GetModulate() ,  
CanvasItem.SetSelfModulate(Color) , CanvasItem.GetSelfModulate() ,  
[CanvasItem.SetDrawBehindParent\(bool\)](#) , CanvasItem.IsDrawBehindParentEnabled() ,  
[CanvasItem.DrawLine\(Vector2, Vector2, Color, float, bool\)](#) ,

[CanvasItem.DrawPolyline\(Vector2\[\], Color, float, bool\)](#) ,  
[CanvasItem.DrawPolylineColors\(Vector2\[\], Color\[\], float, bool\)](#) ,  
[CanvasItem.DrawArc\(Vector2, float, float, float, int, Color, float, bool\)](#) ,  
[CanvasItem.DrawMultiline\(Vector2\[\], Color, float, bool\)](#) ,  
[CanvasItem.DrawMultilineColors\(Vector2\[\], Color\[\], float, bool\)](#) ,  
[CanvasItem.DrawRect\(Rect2, Color, bool, float, bool\)](#) , [CanvasItem.DrawCircle\(Vector2, float, Color\)](#) ,  
CanvasItem.DrawTexture(Texture, Vector2, Color?, Texture) ,  
[CanvasItem.DrawTextureRect\(Texture, Rect2, bool, Color?, bool, Texture\)](#) ,  
[CanvasItem.DrawTextureRectRegion\(Texture, Rect2, Rect2, Color?, bool, Texture, bool\)](#) ,  
CanvasItem.DrawStyleBox(StyleBox, Rect2) ,  
[CanvasItem.DrawPrimitive\(Vector2\[\], Color\[\], Vector2\[\], Texture, float, Texture\)](#) ,  
[CanvasItem.DrawPolygon\(Vector2\[\], Color\[\], Vector2\[\], Texture, Texture, bool\)](#) ,  
[CanvasItem.DrawColoredPolygon\(Vector2\[\], Color, Vector2\[\], Texture, Texture, bool\)](#) ,  
[CanvasItem.DrawString\(Font, Vector2, string, Color?, int\)](#) ,  
[CanvasItem.DrawChar\(Font, Vector2, string, string, Color?\)](#) ,  
CanvasItem.DrawMesh(Mesh, Texture, Texture, Transform2D?, Color?) ,  
CanvasItem.DrawMultimesh(MultiMesh, Texture, Texture) ,  
[CanvasItem.DrawSetTransform\(Vector2, float, Vector2\)](#) ,  
CanvasItem.DrawSetTransformMatrix(Transform2D) , CanvasItem.GetTransform() ,  
CanvasItem.GetGlobalTransform() , CanvasItem.GetGlobalTransformWithCanvas() ,  
CanvasItem.GetViewportTransform() , CanvasItem.GetViewportRect() , CanvasItem.GetCanvasTransform() ,  
CanvasItem.GetLocalMousePosition() , CanvasItem.GetGlobalMousePosition() , CanvasItem.GetCanvas() ,  
CanvasItem.GetWorld2d() , CanvasItem.SetMaterial(Material) , CanvasItem.GetMaterial() ,  
[CanvasItem.SetUseParentMaterial\(bool\)](#) , CanvasItem.GetUseParentMaterial() ,  
[CanvasItem.SetNotifyLocalTransform\(bool\)](#) , CanvasItem.IsLocalTransformNotificationEnabled() ,  
[CanvasItem.SetNotifyTransform\(bool\)](#) , CanvasItem.IsTransformNotificationEnabled() ,  
CanvasItem.ForceUpdateTransform() , CanvasItem.MakeCanvasPositionLocal(Vector2) ,  
CanvasItem.MakeInputLocal(InputEvent) , CanvasItem.Visible , CanvasItem.Modulate ,  
CanvasItem.SelfModulate , CanvasItem.ShowBehindParent , CanvasItem.ShowOnTop ,  
CanvasItem.LightMask , CanvasItem.Material , CanvasItem.UseParentMaterial ,  
Node.NotificationEnterTree , Node.NotificationExitTree , Node.NotificationMovedInParent ,  
Node.NotificationReady , Node.NotificationPaused , Node.NotificationUnpaused ,  
Node.NotificationPhysicsProcess , Node.NotificationProcess , Node.NotificationParented ,  
Node.NotificationUnparented , Node.NotificationInstanced , Node.NotificationDragBegin ,  
Node.NotificationDragEnd , Node.NotificationPathChanged , Node.NotificationInternalProcess ,  
Node.NotificationInternalPhysicsProcess , Node.NotificationPostEnterTree ,  
Node.NotificationResetPhysicsInterpolation , Node.NotificationWmMouseEnter ,  
Node.NotificationWmMouseExit , Node.NotificationWmFocusIn , Node.NotificationWmFocusOut ,  
Node.NotificationWmQuitRequest , Node.NotificationWmGoBackRequest ,  
Node.NotificationWmUnfocusRequest , Node.NotificationOsMemoryWarning ,

Node.NotificationTranslationChanged , Node.NotificationWmAbout , Node.NotificationCrash ,  
Node.NotificationOslmeUpdate , Node.NotificationAppResumed , Node.NotificationAppPaused ,  
Node.GetNode<T>(NodePath) , Node.GetNodeOrNull<T>(NodePath) , [Node.GetChild<T>\(int\)](#) ,  
[Node.GetChildOrNull<T>\(int\)](#) , Node.GetOwner<T>() , Node.GetOwnerOrNull<T>() ,  
Node.GetParent<T>() , Node.GetParentOrNull<T>() , Node.\_EnterTree() , Node.\_ExitTree() ,  
Node.\_GetConfigurationWarning() , Node.\_Input(InputEvent) , [Node. PhysicsProcess\(float\)](#) ,  
[Node. Process\(float\)](#) , Node.\_UnhandledInput(InputEvent) , Node.\_UnhandledKeyInput(InputEventKey) ,  
[Node.AddChildBelowNode\(Node, Node, bool\)](#) , [Node.SetName\(string\)](#) , Node.GetName() ,  
[Node.AddChild\(Node, bool\)](#) , Node.RemoveChild(Node) , Node.GetChildCount() , Node.GetChildren() ,  
[Node.GetChild\(int\)](#) , Node.HasNode(NodePath) , Node.GetNode(NodePath) ,  
Node.GetNodeOrNull(NodePath) , Node.GetParent() , [Node.FindNode\(string, bool, bool\)](#) ,  
[Node.FindParent\(string\)](#) , Node.HasNodeAndResource(NodePath) ,  
Node.GetNodeAndResource(NodePath) , Node.IsInsideTree() , Node.IsAParentOf(Node) ,  
Node.IsGreaterThan(Node) , Node.GetPath() , Node.GetPathTo(Node) ,  
[Node.AddToGroup\(string, bool\)](#) , [Node.RemoveFromGroup\(string\)](#) , [Node.IsInGroup\(string\)](#) ,  
[Node.MoveChild\(Node, int\)](#) , Node.GetGroups() , Node.Raise() , Node.SetOwner(Node) ,  
Node.GetOwner() , Node.RemoveAndSkip() , Node.GetIndex() , Node.PrintTree() , Node.PrintTreePretty() ,  
[Node.SetFilename\(string\)](#) , Node.GetFilename() , [Node.PropagateNotification\(int\)](#) ,  
[Node.PropagateCall\(string, Array, bool\)](#) , [Node.SetPhysicsProcess\(bool\)](#) ,  
Node.GetPhysicsProcessDeltaTime() , Node.IsPhysicsProcessing() , Node.GetProcessDeltaTime() ,  
[Node.SetProcess\(bool\)](#) , [Node.SetProcessPriority\(int\)](#) , Node.GetProcessPriority() ,  
Node.IsProcessing() , [Node.SetProcessInput\(bool\)](#) , Node.IsProcessingInput() ,  
[Node.SetProcessUnhandledInput\(bool\)](#) , Node.IsProcessingUnhandledInput() ,  
[Node.SetProcessUnhandledKeyInput\(bool\)](#) , Node.IsProcessingUnhandledKeyInput() ,  
Node.SetPauseMode(Node.PauseModeEnum) , Node.GetPauseMode() , Node.CanProcess() ,  
Node.PrintStrayNodes() , NodeGetPositionInParent() , [Node.SetDisplayFolded\(bool\)](#) ,  
Node.IsDisplayedFolded() , [Node.SetProcessInternal\(bool\)](#) , Node.IsProcessingInternal() ,  
[Node.SetPhysicsProcessInternal\(bool\)](#) , Node.IsPhysicsProcessingInternal() ,  
Node.SetPhysicsInterpolationMode(Node.PhysicsInterpolationModeEnum) ,  
Node.GetPhysicsInterpolationMode() , Node.IsPhysicsInterpolated() ,  
Node.IsPhysicsInterpolatedAndEnabled() , Node.ResetPhysicsInterpolation() , Node.GetTree() ,  
Node.CreateTween() , [Node.Duplicate\(int\)](#) , [Node.ReplaceBy\(Node, bool\)](#) ,  
[Node.setSceneInstanceLoadPlaceholder\(bool\)](#) , Node.GetSceneInstanceLoadPlaceholder() ,  
Node.GetViewport() , Node.QueueFree() , Node.RequestReady() , [Node.SetNetworkMaster\(int, bool\)](#) ,  
Node.GetNetworkMaster() , Node.IsNetworkMaster() , Node.GetMultiplayer() ,  
Node.GetCustomMultiplayer() , Node.SetCustomMultiplayer(MultiplayerAPI) ,  
[Node.RpcConfig\(string, MultiplayerAPI.RPCMode\)](#) ,  
[Node.RsetConfig\(string, MultiplayerAPI.RPCMode\)](#) , [Node.SetUniqueNameInOwner\(bool\)](#) ,  
Node.IsUniqueNameInOwner() , [Node.Rpc\(string, params object\[\]\)](#) ,  
[Node.RpcUnreliable\(string, params object\[\]\)](#) , [Node.RpIcd\(int, string, params object\[\]\)](#) ,

[Node.RpcUnreliableId\(int, string, params object\[\]\)](#) , [Node.Rset\(string, object\)](#) ,  
[Node.RsetId\(int, string, object\)](#) , [Node.RsetUnreliable\(string, object\)](#) ,  
[Node.RsetUnreliableId\(int, string, object\)](#) , Node.UpdateConfigurationWarning() ,  
Node.EditorDescription , Node.\_ImportPath , Node.PauseMode , Node.PhysicsInterpolationMode ,  
Node.Name , Node.UniqueNameInOwner , Node.Filename , Node.Owner , Node.Multiplayer ,  
Node.CustomMultiplayer , Node.ProcessPriority , Object.NotificationPostinitialize ,  
Object.NotificationPredelete , Object.IsInstanceValid(Object) , Object.WeakRef(Object) , Object.Dispose() ,  
[Object.Dispose\(bool\)](#) , Object.ToString() , [Object.ToSignal\(Object, string\)](#) , [Object.Get\(string\)](#) ,  
Object.\_GetPropertyList() , [Object.Notification\(int\)](#) , [Object.Set\(string, object\)](#) , Object.Free() ,  
Object.GetClass() , [Object.IsClass\(string\)](#) , [Object.Set\(string, object\)](#) , [Object.Get\(string\)](#) ,  
[Object.SetIndexed\(NodePath, object\)](#) , Object.GetIndexed(NodePath) , Object.GetPropertyList() ,  
Object.GetMethodList() , [Object.Notification\(int, bool\)](#) , Object.GetInstanceId() ,  
Object.SetScript(Reference) , Object.GetScript() , [Object.SetMeta\(string, object\)](#) ,  
[Object.RemoveMeta\(string\)](#) , [Object.GetMeta\(string, object\)](#) , [Object.HasMeta\(string\)](#) ,  
Object.GetMetaList() , [Object.AddUserSignal\(string, Array\)](#) , [Object.HasUserSignal\(string\)](#) ,  
[Object.EmitSignal\(string, params object\[\]\)](#) , [Object.Call\(string, params object\[\]\)](#) ,  
[Object.CallDeferred\(string, params object\[\]\)](#) , [Object.SetDeferred\(string, object\)](#) ,  
[Object.Cally\(string, Array\)](#) , [Object.HasMethod\(string\)](#) , [Object.HasSignal\(string\)](#) ,  
Object.GetSignalList() , [Object.GetSignalConnectionList\(string\)](#) , Object.GetIncomingConnections() ,  
[Object.Connect\(string, Object, string, Array, uint\)](#) , [Object.Disconnect\(string, Object, string\)](#) ,  
[Object.IsConnected\(string, Object, string\)](#) , [Object.SetBlockSignals\(bool\)](#) , Object.IsBlockingSignals() ,  
Object.PropertyListChangedNotify() , [Object.SetMessageTranslation\(bool\)](#) ,  
Object.CanTranslateMessages() , [Object.Tr\(string\)](#) , Object.IsQueuedForDeletion() ,  
Object.NativeInstance , Object.DynamicObject , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Extension Methods

[Node\\_GD\\_CB\\_Extension.FindNodeByName\(Node, string\)](#) ,  
[Node\\_GD\\_CB\\_Extension.FindNodeByName\(Node, string, bool\)](#) ,  
[Node\\_GD\\_CB\\_Extension.FindNodeByName\(Node, string, Type, bool\)](#) ,  
[Node\\_GD\\_CB\\_Extension.FindNodeByName<T>\(Node, string\)](#) ,  
[Node\\_GD\\_CB\\_Extension.FindNodeByName<T>\(Node, string, bool\)](#) ,  
[Node\\_GD\\_CB\\_Extension.FindNodes\(Node, Type\)](#) , [Node\\_GD\\_CB\\_Extension.FindNodes\(Node, Type, bool\)](#) ,  
[Node\\_GD\\_CB\\_Extension.FindNodes<T>\(Node\)](#) , [Node\\_GD\\_CB\\_Extension.FindNodes<T>\(Node, bool\)](#) ,  
[Node\\_GD\\_CB\\_Extension.GetNodePosition\(Node\)](#) , [Node\\_GD\\_CB\\_Extension.GetNodeRotation\(Node\)](#) ,  
[Node\\_GD\\_CB\\_Extension.GetNodeScale\(Node\)](#) , [Node\\_GD\\_CB\\_Extension.Print\(Node, params object\[\]\)](#) ,  
[Node\\_GD\\_CB\\_Extension.SetNodePosition\(Node, Vector3D\)](#) ,  
[Node\\_GD\\_CB\\_Extension.SetNodeRotation\(Node, Vector3D\)](#) ,  
[Node\\_GD\\_CB\\_Extension.SetNodeScale\(Node, Vector3D\)](#)

## Methods

### RayCast(Camera2D, Vector2, Vector2, out RayHit2D)

```
public static bool RayCast(Camera2D camera, Vector2 from, Vector2 to, out RayHit2D hit)
```

#### Parameters

**camera** Camera2D

**from** Vector2

**to** Vector2

**hit** [RayHit2D](#)

#### Returns

[bool](#)

### RayCast(Camera2D, Vector2, Vector2, CollisionObject2D[], out RayHit2D)

```
public static bool RayCast(Camera2D camera, Vector2 from, Vector2 to, CollisionObject2D[] exclude, out RayHit2D hit)
```

#### Parameters

**camera** Camera2D

**from** Vector2

**to** Vector2

**exclude** CollisionObject2D[]

**hit** [RayHit2D](#)

#### Returns

[bool](#)

RayCast(Camera2D?, Vector2, Vector2, CollisionObject2D[]?, uint, out RayHit2D)

```
public static bool RayCast(Camera2D? camera, Vector2 from, Vector2 to, CollisionObject2D[]? exclude, uint collisionLayer, out RayHit2D hit)
```

Parameters

**camera** Camera2D

**from** Vector2

**to** Vector2

**exclude** CollisionObject2D[]

**collisionLayer** [uint](#)

**hit** [RayHit2D](#)

Returns

[bool](#)

RayCast(Camera2D, Vector2, Vector2, uint, out RayHit2D)

```
public static bool RayCast(Camera2D camera, Vector2 from, Vector2 to, uint collisionLayer, out RayHit2D hit)
```

Parameters

**camera** Camera2D

**from** Vector2

**to** Vector2

`collisionLayer` `uint`

`hit` `RayHit2D`

Returns

`bool`

`RayCastAllBox(Camera2D, Vector2, Vector2, CollisionObject2D[], List<Hit2D>)`

```
public static bool RayCastAllBox(Camera2D camera, Vector2 mousePosition, Vector2 size,  
CollisionObject2D[] exclude, List<Hit2D> list)
```

Parameters

`camera` Camera2D

`mousePosition` Vector2

`size` Vector2

`exclude` CollisionObject2D[]

`list` `List`<`Hit2D`>

Returns

`bool`

`RayCastAllBox(Camera2D?, Vector2, Vector2,  
CollisionObject2D[]?, uint, List<Hit2D>)`

```
public static bool RayCastAllBox(Camera2D? camera, Vector2 mousePosition, Vector2 size,  
CollisionObject2D[]? exclude, uint collisionLayer, List<Hit2D> list)
```

Parameters

```
camera Camera2D  
  
mousePosition Vector2  
  
size Vector2  
  
exclude CollisionObject2D[]  
  
collisionLayer uint  
  
list List<Hit2D>
```

Returns

[bool](#)

## RayCastAllBox(Camera2D, Vector2, Vector2, List<Hit2D>)

```
public static bool RayCastAllBox(Camera2D camera, Vector2 mousePosition, Vector2 size,  
List<Hit2D> list)
```

Parameters

```
camera Camera2D  
  
mousePosition Vector2  
  
size Vector2  
  
list List<Hit2D>
```

Returns

[bool](#)

## RayCastAllBox(Camera2D, Vector2, Vector2, uint, List<Hit2D>)

```
public static bool RayCastAllBox(Camera2D camera, Vector2 mousePosition, Vector2 size, uint  
collisionLayer, List<Hit2D> list)
```

Parameters

**camera** Camera2D

**mousePosition** Vector2

**size** Vector2

**collisionLayer** uint

**list** List<Hit2D>

Returns

[bool](#)

RayCastAllCircle(Camera2D, Vector2, float, CollisionObject2D[], List<Hit2D>)

```
public static bool RayCastAllCircle(Camera2D camera, Vector2 mousePosition, float radius,
CollisionObject2D[] exclude, List<Hit2D> list)
```

Parameters

**camera** Camera2D

**mousePosition** Vector2

**radius** float

**exclude** CollisionObject2D[]

**list** List<Hit2D>

Returns

[bool](#)

RayCastAllCircle(Camera2D?, Vector2, float,

## CollisionObject2D[]?, uint, List<Hit2D>)

```
public static bool RayCastAllCircle(Camera2D? camera, Vector2 mousePosition, float radius,  
CollisionObject2D[]? exclude, uint collisionLayer, List<Hit2D> list)
```

### Parameters

**camera** Camera2D

**mousePosition** Vector2

**radius** [float](#)

**exclude** CollisionObject2D[]

**collisionLayer** [uint](#)

**list** [List](#)<Hit2D>

### Returns

[bool](#)

## RayCastAllCircle(Camera2D, Vector2, float, List<Hit2D>)

```
public static bool RayCastAllCircle(Camera2D camera, Vector2 mousePosition, float radius,  
List<Hit2D> list)
```

### Parameters

**camera** Camera2D

**mousePosition** Vector2

**radius** [float](#)

**list** [List](#)<Hit2D>

### Returns

[bool](#)

## RayCastAllCircle(Camera2D, Vector2, float, uint, List<Hit2D>)

```
public static bool RayCastAllCircle(Camera2D camera, Vector2 mousePosition, float radius,  
uint collisionLayer, List<Hit2D> list)
```

### Parameters

**camera** Camera2D

**mousePosition** Vector2

**radius** [float](#)

**collisionLayer** [uint](#)

**list** [List](#)<Hit2D>

### Returns

[bool](#)

## RayCastBox(Camera2D, Vector2, Vector2, out Hit2D)

```
public static bool RayCastBox(Camera2D camera, Vector2 mousePosition, Vector2 size, out  
Hit2D hit)
```

### Parameters

**camera** Camera2D

**mousePosition** Vector2

**size** Vector2

**hit** [Hit2D](#)

### Returns

[bool](#)

RayCastBox(Camera2D, Vector2, Vector2, CollisionObject2D[],  
out Hit2D)

```
public static bool RayCastBox(Camera2D camera, Vector2 mousePosition, Vector2 size,  
CollisionObject2D[] exclude, out Hit2D hit)
```

Parameters

**camera** Camera2D

**mousePosition** Vector2

**size** Vector2

**exclude** CollisionObject2D[]

**hit** [Hit2D](#)

Returns

[bool](#)

RayCastBox(Camera2D, Vector2, Vector2, CollisionObject2D[]?,  
uint, out Hit2D)

```
public static bool RayCastBox(Camera2D camera, Vector2 mousePosition, Vector2 size,  
CollisionObject2D[]? exclude, uint collisionLayer, out Hit2D hit)
```

Parameters

**camera** Camera2D

**mousePosition** Vector2

**size** Vector2

**exclude** CollisionObject2D[]

**collisionLayer** [uint](#)

**hit** [Hit2D](#)

Returns

[bool](#)

## RayCastBox(Camera2D, Vector2, Vector2, uint, out Hit2D)

```
public static bool RayCastBox(Camera2D camera, Vector2 mousePosition, Vector2 size, uint  
collisionLayer, out Hit2D hit)
```

Parameters

**camera** Camera2D

**mousePosition** Vector2

**size** Vector2

**collisionLayer** [uint](#)

**hit** [Hit2D](#)

Returns

[bool](#)

## RayCastCircle(Camera2D, Vector2, float, out Hit2D)

```
public static bool RayCastCircle(Camera2D camera, Vector2 mousePosition, float radius, out  
Hit2D hit)
```

Parameters

**camera** Camera2D

**mousePosition** Vector2

`radius` [float](#)

`hit` [Hit2D](#)

Returns

[bool](#)

**RayCastCircle(Camera2D, Vector2, float, CollisionObject2D[], out Hit2D)**

```
public static bool RayCastCircle(Camera2D camera, Vector2 mousePosition, float radius,
CollisionObject2D[] exclude, out Hit2D hit)
```

Parameters

`camera` Camera2D

`mousePosition` Vector2

`radius` [float](#)

`exclude` CollisionObject2D[]

`hit` [Hit2D](#)

Returns

[bool](#)

**RayCastCircle(Camera2D, Vector2, float, CollisionObject2D[]?, uint, out Hit2D)**

```
public static bool RayCastCircle(Camera2D camera, Vector2 mousePosition, float radius,
CollisionObject2D[]? exclude, uint collisionLayer, out Hit2D hit)
```

Parameters

`camera` Camera2D

`mousePosition` Vector2

`radius` [float](#)

`exclude` CollisionObject2D[]

`collisionLayer` [uint](#)

`hit` [Hit2D](#)

Returns

[bool](#)

## RayCastCircle(Camera2D, Vector2, float, uint, out Hit2D)

```
public static bool RayCastCircle(Camera2D camera, Vector2 mousePosition, float radius, uint collisionLayer, out Hit2D hit)
```

Parameters

`camera` Camera2D

`mousePosition` Vector2

`radius` [float](#)

`collisionLayer` [uint](#)

`hit` [Hit2D](#)

Returns

[bool](#)

## RayCastHit(Camera2D, Vector2, out Hit2D)

```
public static bool RayCastHit(Camera2D camera, Vector2 mousePosition, out Hit2D hit)
```

Parameters

**camera** Camera2D

**mousePosition** Vector2

**hit** [Hit2D](#)

Returns

[bool](#) ↗

**RayCastHit(Camera2D, Vector2, CollisionObject2D[], out Hit2D)**

```
public static bool RayCastHit(Camera2D camera, Vector2 mousePosition, CollisionObject2D[] exclude, out Hit2D hit)
```

Parameters

**camera** Camera2D

**mousePosition** Vector2

**exclude** CollisionObject2D[]

**hit** [Hit2D](#)

Returns

[bool](#) ↗

**RayCastHit(Camera2D, Vector2, CollisionObject2D[]?, uint, out Hit2D)**

```
public static bool RayCastHit(Camera2D camera, Vector2 mousePosition, CollisionObject2D[]? exclude, uint collisionLayer, out Hit2D hit)
```

Parameters

`camera` Camera2D

`mousePosition` Vector2

`exclude` CollisionObject2D[]

`collisionLayer` [uint](#)

`hit` [Hit2D](#)

Returns

[bool](#)

## RayCastHit(Camera2D, Vector2, uint, out Hit2D)

```
public static bool RayCastHit(Camera2D camera, Vector2 mousePosition, uint collisionLayer,  
out Hit2D hit)
```

Parameters

`camera` Camera2D

`mousePosition` Vector2

`collisionLayer` [uint](#)

`hit` [Hit2D](#)

Returns

[bool](#)

## RayCastHitAll(Camera2D, Vector2, CollisionObject2D[], List<Hit2D>)

```
public static bool RayCastHitAll(Camera2D camera, Vector2 mousePosition, CollisionObject2D[]  
exclude, List<Hit2D> list)
```

Parameters

`camera Camera2D`

`mousePosition Vector2`

`exclude CollisionObject2D[]`

`list List<Hit2D>`

Returns

`bool`

**RayCastHitAll(Camera2D?, Vector2, CollisionObject2D[], uint, List<Hit2D>)**

```
public static bool RayCastHitAll(Camera2D? camera, Vector2 mousePosition,  
CollisionObject2D[]? exclude, uint collisionLayer, List<Hit2D> list)
```

Parameters

`camera Camera2D`

`mousePosition Vector2`

`exclude CollisionObject2D[]`

`collisionLayer uint`

`list List<Hit2D>`

Returns

`bool`

**RayCastHitAll(Camera2D, Vector2, List<Hit2D>)**

```
public static bool RayCastHitAll(Camera2D camera, Vector2 mousePosition, List<Hit2D> list)
```

## Parameters

`camera` Camera2D

`mousePosition` Vector2

`list` [List<Hit2D>](#)

## Returns

`bool`

## RayCastHitAll(Camera2D, Vector2, uint, List<Hit2D>)

```
public static bool RayCastHitAll(Camera2D camera, Vector2 mousePosition, uint  
collisionLayer, List<Hit2D> list)
```

## Parameters

`camera` Camera2D

`mousePosition` Vector2

`collisionLayer` [uint](#)

`list` [List<Hit2D>](#)

## Returns

`bool`

## \_Ready()

Called when the node is "ready", i.e. when both the node and its children have entered the scene tree. If the node has children, their Godot.Node.\_Ready() callbacks get triggered first, and the parent node will receive the ready notification afterwards.

Corresponds to the Godot.Node.NotificationReady notification in [Notification\(int\)](#). See also the `onready` keyword for variables.

Usually used for initialization. For even earlier initialization, may be used. See also Godot.Node.\_EnterTree().

Note: Godot.Node.\_Ready() may be called only once for each node. After removing a node from the scene tree and adding it again, `_ready` will not be called a second time. This can be bypassed by requesting another call with Godot.Node.RequestReady(), which may be called anywhere before adding the node again.

```
public override void _Ready()
```

# Struct RayHit2D

Namespace: [Cobilas.GodotEngine.Utility.Physics](#)

Assembly: com.cobilas.godot.utility.dll

```
public struct RayHit2D
```

## Inherited Members

[ValueType.Equals\(object\)](#) , [ValueType.GetHashCode\(\)](#) , [ValueType.ToString\(\)](#) ,  
[object.Equals\(object, object\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.GetType\(\)](#)

## Properties

### Collision

```
public readonly Node Collision { get; }
```

#### Property Value

Node

### ID

```
public readonly int ID { get; }
```

#### Property Value

[int](#)

### MetaData

```
public readonly object MetaData { get; }
```

Property Value

object ↗

Name

```
public readonly string Name { get; }
```

Property Value

string ↗

Normal

```
public readonly Vector2 Normal { get; }
```

Property Value

Vector2

Position

```
public readonly Vector2 Position { get; }
```

Property Value

Vector2

RID

```
public readonly RID RID { get; }
```

Property Value

## Operators

### explicit operator RayHit2D(Dictionary?)

```
public static explicit operator RayHit2D(Dictionary? D)
```

#### Parameters

D Dictionary

#### Returns

[RayHit2D](#)

# Namespace Cobilas.GodotEngine.Utility.Runtime

## Classes

[RunTimeInitialization](#)

[RunTimeInitializationClassAttribute](#)

## Structs

[RunTimeInitialization.PriorityList](#)

## Enums

[Priority](#)

# Enum Priority

Namespace: [Cobilas.GodotEngine.Utility.Runtime](#)

Assembly: com.cobilas.godot.utility.dll

```
public enum Priority
```

## Fields

StartBefore = 0

StartLater = 1

# Class RunTimeInitialization

Namespace: [Cobilas.GodotEngine.Utility.Runtime](#)

Assembly: com.cobilas.godot.utility.dll

```
public class RunTimeInitialization : Node, IDisposable
```

## Inheritance

[object](#) ← Object ← Node ← RunTimeInitialization

## Implements

[IDisposable](#)

## Inherited Members

Node.NotificationEnterTree , Node.NotificationExitTree , Node.NotificationMovedInParent ,  
Node.NotificationReady , Node.NotificationPaused , Node.NotificationUnpaused ,  
Node.NotificationPhysicsProcess , Node.NotificationProcess , Node.NotificationParented ,  
Node.NotificationUnparented , Node.NotificationInstanced , Node.NotificationDragBegin ,  
Node.NotificationDragEnd , Node.NotificationPathChanged , Node.NotificationInternalProcess ,  
Node.NotificationInternalPhysicsProcess , Node.NotificationPostEnterTree ,  
Node.NotificationResetPhysicsInterpolation , Node.NotificationWmMouseEnter ,  
Node.NotificationWmMouseExit , Node.NotificationWmFocusIn , Node.NotificationWmFocusOut ,  
Node.NotificationWmQuitRequest , Node.NotificationWmGoBackRequest ,  
Node.NotificationWmUnfocusRequest , Node.NotificationOsMemoryWarning ,  
Node.NotificationTranslationChanged , Node.NotificationWmAbout , Node.NotificationCrash ,  
Node.NotificationOslmeUpdate , Node.NotificationAppResumed , Node.NotificationAppPaused ,  
Node.GetNode<T>(NodePath) , Node.GetNodeOrNull<T>(NodePath) , [Node.GetChild<T>\(int\)](#) ,  
[Node.GetChildOrNull<T>\(int\)](#) , Node.GetOwner<T>() , Node.GetOwnerOrNull<T>() ,  
Node.GetParent<T>() , Node.GetParentOrNull<T>() , Node.\_EnterTree() , Node.\_ExitTree() ,  
Node.\_GetConfigurationWarning() , Node.\_Input(InputEvent) , [Node\\_PhysicsProcess\(float\)](#) ,  
[Node\\_Process\(float\)](#) , Node.\_UnhandledInput(InputEvent) , Node.\_UnhandledKeyInput(InputEventKey) ,  
[Node.AddChildBelowNode\(Node, Node, bool\)](#) , [Node.SetName\(string\)](#) , Node.GetName() ,  
[Node.AddChild\(Node, bool\)](#) , Node.RemoveChild(Node) , Node.GetChildCount() , Node.GetChildren() ,  
[Node.GetChild\(int\)](#) , Node.HasNode(NodePath) , Node.GetNode(NodePath) ,  
Node.GetNodeOrNull(NodePath) , Node.GetParent() , [Node.FindNode\(string, bool, bool\)](#) ,  
[Node.FindParent\(string\)](#) , Node.HasNodeAndResource(NodePath) ,  
Node.GetNodeAndResource(NodePath) , Node.IsInsideTree() , Node.IsAParentOf(Node) ,  
Node.IsGreater Than(Node) , Node.GetPath() , Node.GetPathTo(Node) ,  
[Node.AddToGroup\(string, bool\)](#) , [Node.RemoveFromGroup\(string\)](#) , [Node.IsInGroup\(string\)](#) ,

[Node.MoveChild\(Node, int\)](#) , Node.GetGroups() , Node.Raise() , Node.SetOwner(Node) ,  
Node.GetOwner() , Node.RemoveAndSkip() , Node.GetIndex() , Node.PrintTree() , Node.PrintTreePretty() ,  
[Node.SetFilename\(string\)](#) , Node.GetFilename() , [Node.PropagateNotification\(int\)](#) ,  
[Node.PropagateCall\(string, Array, bool\)](#) , [Node.SetPhysicsProcess\(bool\)](#) ,  
Node.GetPhysicsProcessDeltaTime() , Node.IsPhysicsProcessing() , Node.GetProcessDeltaTime() ,  
[Node.SetProcess\(bool\)](#) , [Node.SetProcessPriority\(int\)](#) , Node.GetProcessPriority() ,  
Node.IsProcessing() , [Node.SetProcessInput\(bool\)](#) , Node.IsProcessingInput() ,  
[Node.SetProcessUnhandledInput\(bool\)](#) , Node.IsProcessingUnhandledInput() ,  
[Node.SetProcessUnhandledKeyInput\(bool\)](#) , Node.IsProcessingUnhandledKeyInput() ,  
Node.SetPauseMode(Node.PauseModeEnum) , Node.GetPauseMode() , Node.CanProcess() ,  
Node.PrintStrayNodes() , NodeGetPositionInParent() , [Node.SetDisplayFolded\(bool\)](#) ,  
Node.IsDisplayedFolded() , [Node.SetProcessInternal\(bool\)](#) , Node.IsProcessingInternal() ,  
[Node.SetPhysicsProcessInternal\(bool\)](#) , Node.IsPhysicsProcessingInternal() ,  
Node.SetPhysicsInterpolationMode(Node.PhysicsInterpolationModeEnum) ,  
Node.GetPhysicsInterpolationMode() , Node.IsPhysicsInterpolated() ,  
Node.IsPhysicsInterpolatedAndEnabled() , Node.ResetPhysicsInterpolation() , Node.GetTree() ,  
Node.CreateTween() , [Node.Duplicate\(int\)](#) , [Node.ReplaceBy\(Node, bool\)](#) ,  
[Node.setSceneInstanceLoadPlaceholder\(bool\)](#) , Node.GetSceneInstanceLoadPlaceholder() ,  
Node.GetViewport() , Node.QueueFree() , Node.RequestReady() , [Node.SetNetworkMaster\(int, bool\)](#) ,  
Node.GetNetworkMaster() , Node.IsNetworkMaster() , Node.GetMultiplayer() ,  
Node.GetCustomMultiplayer() , Node.SetCustomMultiplayer(MultiplayerAPI) ,  
[Node.RpcConfig\(string, MultiplayerAPI.RPCMode\)](#) ,  
[Node.RsetConfig\(string, MultiplayerAPI.RPCMode\)](#) , [Node.SetUniqueNameInOwner\(bool\)](#) ,  
Node.IsUniqueNameInOwner() , [Node.Rpc\(string, params object\[\]\)](#) ,  
[Node.RpcUnreliable\(string, params object\[\]\)](#) , [Node.Rpcld\(int, string, params object\[\]\)](#) ,  
[Node.RpcUnreliableId\(int, string, params object\[\]\)](#) , [Node.Rset\(string, object\)](#) ,  
[Node.RsetId\(int, string, object\)](#) , [Node.RsetUnreliable\(string, object\)](#) ,  
[Node.RsetUnreliableId\(int, string, object\)](#) , Node.UpdateConfigurationWarning() ,  
Node.EditorDescription , Node.\_ImportPath , Node.PauseMode , Node.PhysicsInterpolationMode ,  
Node.Name , Node.UniqueNameInOwner , Node.Filename , Node.Owner , Node.Multiplayer ,  
Node.CustomMultiplayer , Node.ProcessPriority , Object.NotificationPostInitialize ,  
Object.NotificationPredelete , Object.IsInstanceValid(Object) , Object.WeakRef(Object) , Object.Dispose() ,  
[Object.Dispose\(bool\)](#) , Object.ToString() , [Object.ToSignal\(Object, string\)](#) , [Object.Get\(string\)](#) ,  
Object.\_GetPropertyList() , [Object.Notification\(int\)](#) , [Object.Set\(string, object\)](#) , Object.Free() ,  
Object.GetClass() , [Object.IsClass\(string\)](#) , [Object.Set\(string, object\)](#) , [Object.Get\(string\)](#) ,  
[Object.SetIndexed\(NodePath, object\)](#) , Object.GetIndexed(NodePath) , Object.GetPropertyList() ,  
Object.GetMethodList() , [Object.Notification\(int, bool\)](#) , Object.GetInstanceId() ,  
Object.SetScript(Reference) , Object.GetScript() , [Object.SetMeta\(string, object\)](#) ,  
[Object.RemoveMeta\(string\)](#) , [Object.GetMeta\(string, object\)](#) , [Object.HasMeta\(string\)](#) ,  
Object.GetMetaList() , [Object.AddUserSignal\(string, Array\)](#) , [Object.HasUserSignal\(string\)](#) ,

[Object.EmitSignal\(string, params object\[\]\)](#) , [Object.Call\(string, params object\[\]\)](#) ,  
[Object.CallDeferred\(string, params object\[\]\)](#) , [Object.SetDeferred\(string, object\)](#) ,  
[Object.Cally\(string, Array\)](#) , [Object.HasMethod\(string\)](#) , [Object.HasSignal\(string\)](#) ,  
Object.GetSignalList() , [Object.GetSignalConnectionList\(string\)](#) , Object.GetIncomingConnections() ,  
[Object.Connect\(string, Object, string, Array, uint\)](#) , [Object.Disconnect\(string, Object, string\)](#) ,  
[Object.IsConnected\(string, Object, string\)](#) , [Object.SetBlockSignals\(bool\)](#) , Object.IsBlockingSignals() ,  
Object.PropertyListChangedNotify() , [Object.SetMessageTranslation\(bool\)](#) ,  
Object.CanTranslateMessages() , [Object.Tr\(string\)](#) , Object.IsQueuedForDeletion() ,  
Object.NativeInstance , Object.DynamicObject , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Extension Methods

[Node GD CB Extension.FindNodeByName\(Node, string\)](#) ,  
[Node GD CB Extension.FindNodeByName\(Node, string, bool\)](#) ,  
[Node GD CB Extension.FindNodeByName\(Node, string, Type, bool\)](#) ,  
[Node GD CB Extension.FindNodeByName<T>\(Node, string\)](#) ,  
[Node GD CB Extension.FindNodeByName<T>\(Node, string, bool\)](#) ,  
[Node GD CB Extension.FindNodes\(Node, Type\)](#) , [Node GD CB Extension.FindNodes\(Node, Type, bool\)](#) ,  
[Node GD CB Extension.FindNodes<T>\(Node\)](#) , [Node GD CB Extension.FindNodes<T>\(Node, bool\)](#) ,  
[Node GD CB Extension.GetNodePosition\(Node\)](#) , [Node GD CB Extension.GetNodeRotation\(Node\)](#) ,  
[Node GD CB Extension.GetNodeScale\(Node\)](#) , [Node GD CB Extension.Print\(Node, params object\[\]\)](#) ,  
[Node GD CB Extension.SetNodePosition\(Node, Vector3D\)](#) ,  
[Node GD CB Extension.SetNodeRotation\(Node, Vector3D\)](#) ,  
[Node GD CB Extension.SetNodeScale\(Node, Vector3D\)](#)

## Fields

### DeltaTime

```
public const float DeltaTime = 0.33333334
```

#### Field Value

[float](#)

### FixedDeltaTime

```
public const float FixedDeltaTime = 0.02
```

## Field Value

[float](#) ↗

## Methods

### \_Ready()

Called when the node is "ready", i.e. when both the node and its children have entered the scene tree. If the node has children, their Godot.Node.\_Ready() callbacks get triggered first, and the parent node will receive the ready notification afterwards.

Corresponds to the Godot.Node.NotificationReady notification in [Notification\(int\)](#) ↗. See also the [onready](#) keyword for variables.

Usually used for initialization. For even earlier initialization, may be used. See also Godot.Node.\_EnterTree().

Note: Godot.Node.\_Ready() may be called only once for each node. After removing a node from the scene tree and adding it again, [\\_ready](#) will not be called a second time. This can be bypassed by requesting another call with Godot.Node.RequestReady(), which may be called anywhere before adding the node again.

```
public override void _Ready()
```

# Struct RunTimeInitialization.PriorityList

Namespace: [Cobilas.GodotEngine.Utility.Runtime](#)

Assembly: com.cobilas.godot.utility.dll

```
public struct RunTimeInitialization.PriorityList : IDisposable
```

Implements

[IDisposable](#)

Inherited Members

[ValueType.Equals\(object\)](#) , [ValueType.GetHashCode\(\)](#) , [ValueType.ToString\(\)](#) ,  
[object.Equals\(object, object\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.GetType\(\)](#)

## Methods

Add(int, Node)

```
public RunTimeInitialization.PriorityList Add(int index, Node node)
```

Parameters

index [int](#)

node [Node](#)

Returns

[RunTimeInitialization.PriorityList](#)

Dispose()

Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources.

```
public void Dispose()
```

## Run(Node)

```
public readonly void Run(Node root)
```

### Parameters

**root** Node

# Class RunTimeInitializationClassAttribute

Namespace: [Cobilas.GodotEngine.Utility.Runtime](#)

Assembly: com.cobilas.godot.utility.dll

```
[AttributeUsage(AttributeTargets.Class, Inherited = false, AllowMultiple = false)]
public sealed class RunTimeInitializationClassAttribute : Attribute, _Attribute
```

## Inheritance

[object](#) ← [Attribute](#) ← RunTimeInitializationClassAttribute

## Implements

[Attribute](#)

## Inherited Members

[Attribute.GetCustomAttributes\(MemberInfo, Type\)](#) ,  
[Attribute.GetCustomAttributes\(MemberInfo, Type, bool\)](#) ,  
[Attribute.GetCustomAttributes\(MemberInfo\)](#) , [Attribute.GetCustomAttributes\(MemberInfo, bool\)](#) ,  
[Attribute.IsDefined\(MemberInfo, Type\)](#) , [Attribute.IsDefined\(MemberInfo, Type, bool\)](#) ,  
[Attribute.GetCustomAttribute\(MemberInfo, Type\)](#) ,  
[Attribute.GetCustomAttribute\(MemberInfo, Type, bool\)](#) ,  
[Attribute.GetCustomAttributes\(ParameterInfo\)](#) , [Attribute.GetCustomAttributes\(ParameterInfo, Type\)](#) ,  
[Attribute.GetCustomAttributes\(ParameterInfo, Type, bool\)](#) ,  
[Attribute.GetCustomAttributes\(ParameterInfo, bool\)](#) , [Attribute.IsDefined\(ParameterInfo, Type\)](#) ,  
[Attribute.IsDefined\(ParameterInfo, Type, bool\)](#) , [Attribute.GetCustomAttribute\(ParameterInfo, Type\)](#) ,  
[Attribute.GetCustomAttribute\(ParameterInfo, Type, bool\)](#) ,  
[Attribute.GetCustomAttributes\(Module, Type\)](#) , [Attribute.GetCustomAttributes\(Module\)](#) ,  
[Attribute.GetCustomAttributes\(Module, bool\)](#) , [Attribute.GetCustomAttributes\(Module, Type, bool\)](#) ,  
[Attribute.IsDefined\(Module, Type\)](#) , [Attribute.IsDefined\(Module, Type, bool\)](#) ,  
[Attribute.GetCustomAttribute\(Module, Type\)](#) , [Attribute.GetCustomAttribute\(Module, Type, bool\)](#) ,  
[Attribute.GetCustomAttributes\(Assembly, Type\)](#) ,  
[Attribute.GetCustomAttributes\(Assembly, Type, bool\)](#) , [Attribute.GetCustomAttributes\(Assembly\)](#) ,  
[Attribute.GetCustomAttributes\(Assembly, bool\)](#) , [Attribute.IsDefined\(Assembly, Type\)](#) ,  
[Attribute.IsDefined\(Assembly, Type, bool\)](#) , [Attribute.GetCustomAttribute\(Assembly, Type\)](#) ,  
[Attribute.GetCustomAttribute\(Assembly, Type, bool\)](#) , [Attribute.Equals\(object\)](#) ,  
[Attribute.GetHashCode\(\)](#) , [Attribute.Match\(object\)](#) , [Attribute.IsDefaultAttribute\(\)](#) ,  
[Attribute.TypeId](#) , [object.ToString\(\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetType\(\)](#)

## Constructors

### RunTimeInitializationClassAttribute()

```
public RunTimeInitializationClassAttribute()
```

### RunTimeInitializationClassAttribute(Priority)

```
public RunTimeInitializationClassAttribute(Priority bootPriority)
```

#### Parameters

bootPriority [Priority](#)

### RunTimeInitializationClassAttribute(Priority, string?)

```
public RunTimeInitializationClassAttribute(Priority bootPriority, string? name)
```

#### Parameters

bootPriority [Priority](#)

name [string](#) ↗

### RunTimeInitializationClassAttribute(Priority, string?, int)

```
public RunTimeInitializationClassAttribute(Priority bootPriority, string? name,  
int subPriority)
```

#### Parameters

bootPriority [Priority](#)

name [string](#) ↗

`subPriority` [int](#)

## RunTimeInitializationClassAttribute(string)

`public RunTimeInitializationClassAttribute(string name)`

### Parameters

`name` [string](#)

## RunTimeInitializationClassAttribute(string, int)

`public RunTimeInitializationClassAttribute(string name, int subPriority)`

### Parameters

`name` [string](#)

`subPriority` [int](#)

## Properties

### BootPriority

`public Priority BootPriority { get; }`

### Property Value

[Priority](#)

### ClassName

`public string ClassName { get; }`

Property Value

[string ↗](#)

SubPriority

```
public int SubPriority { get; }
```

Property Value

[int ↗](#)

# Namespace Cobilas.GodotEngine.Utility.Scene

## Classes

[SceneManager](#)

## Structs

[Scene](#)

# Struct Scene

Namespace: [Cobilas.GodotEngine.Utility.Scene](#)

Assembly: com.cobilas.godot.utility.dll

```
public struct Scene
```

## Inherited Members

[ValueType.Equals\(object\)](#) , [ValueType.GetHashCode\(\)](#) , [ValueType.ToString\(\)](#) ,  
[object.Equals\(object, object\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.GetType\(\)](#)

## Constructors

### Scene(string, int)

```
public Scene(string scenePath, int index)
```

#### Parameters

scenePath [string](#)

index [int](#)

## Properties

### Index

```
public readonly int Index { get; }
```

#### Property Value

[int](#)

### Name

```
public readonly string Name { get; }
```

Property Value

[string](#) ↗

NameWithoutExtension

```
public readonly string NameWithoutExtension { get; }
```

Property Value

[string](#) ↗

ScenePath

```
public readonly string ScenePath { get; }
```

Property Value

[string](#) ↗

# Class SceneManager

Namespace: [Cobilas.GodotEngine.Utility.Scene](#)

Assembly: com.cobilas.godot.utility.dll

```
[RunTimeInitializationClass("SceneManager")]
public class SceneManager : Node, IDisposable
```

## Inheritance

[object](#) ← Object ← Node ← SceneManager

## Implements

[IDisposable](#)

## Inherited Members

Node.NotificationEnterTree , Node.NotificationExitTree , Node.NotificationMovedInParent ,  
Node.NotificationReady , Node.NotificationPaused , Node.NotificationUnpaused ,  
Node.NotificationPhysicsProcess , Node.NotificationProcess , Node.NotificationParented ,  
Node.NotificationUnparented , Node.NotificationInstanced , Node.NotificationDragBegin ,  
Node.NotificationDragEnd , Node.NotificationPathChanged , Node.NotificationInternalProcess ,  
Node.NotificationInternalPhysicsProcess , Node.NotificationPostEnterTree ,  
Node.NotificationResetPhysicsInterpolation , Node.NotificationWmMouseEnter ,  
Node.NotificationWmMouseExit , Node.NotificationWmFocusIn , Node.NotificationWmFocusOut ,  
Node.NotificationWmQuitRequest , Node.NotificationWmGoBackRequest ,  
Node.NotificationWmUnfocusRequest , Node.NotificationOsMemoryWarning ,  
Node.NotificationTranslationChanged , Node.NotificationWmAbout , Node.NotificationCrash ,  
Node.NotificationOslmeUpdate , Node.NotificationAppResumed , Node.NotificationAppPaused ,  
Node.GetNode<T>(NodePath) , Node.GetNodeOrNull<T>(NodePath) , [Node.GetChild<T>\(int\)](#) ,  
[Node.GetChildOrNull<T>\(int\)](#) , Node.GetOwner<T>() , Node.GetOwnerOrNull<T>() ,  
Node.GetParent<T>() , Node.GetParentOrNull<T>() , Node.\_EnterTree() , Node.\_ExitTree() ,  
Node.\_GetConfigurationWarning() , Node.\_Input(InputEvent) , [Node\\_PhysicsProcess\(float\)](#) ,  
[Node\\_Process\(float\)](#) , Node.\_UnhandledInput(InputEvent) , Node.\_UnhandledKeyInput(InputEventKey) ,  
[Node.AddChildBelowNode\(Node, Node, bool\)](#) , [NodeSetName\(string\)](#) , Node.GetName() ,  
[Node.AddChild\(Node, bool\)](#) , Node.RemoveChild(Node) , Node.GetChildCount() , Node.GetChildren() ,  
[Node.GetChild\(int\)](#) , Node.HasNode(NodePath) , Node.GetNode(NodePath) ,  
Node.GetNodeOrNull(NodePath) , Node.GetParent() , [Node.FindNode\(string, bool, bool\)](#) ,  
[Node.FindParent\(string\)](#) , Node.HasNodeAndResource(NodePath) ,  
Node.GetNodeAndResource(NodePath) , Node.IsInsideTree() , Node.IsAParentOf(Node) ,  
Node.IsGreater Than(Node) , Node.GetPath() , Node.GetPathTo(Node) ,

[Node.AddToGroup\(string, bool\)](#) , [Node.RemoveFromGroup\(string\)](#) , [Node.IsInGroup\(string\)](#) ,  
[Node.MoveChild\(Node, int\)](#) , Node.GetGroups() , Node.Raise() , Node.SetOwner(Node) ,  
Node.GetOwner() , Node.RemoveAndSkip() , Node.GetIndex() , Node.PrintTree() , Node.PrintTreePretty() ,  
[Node.SetFilename\(string\)](#) , Node.GetFilename() , [Node.PropagateNotification\(int\)](#) ,  
[Node.PropagateCall\(string, Array, bool\)](#) , [Node.SetPhysicsProcess\(bool\)](#) ,  
Node.GetPhysicsProcessDeltaTime() , Node.IsPhysicsProcessing() , Node.GetProcessDeltaTime() ,  
[Node.SetProcess\(bool\)](#) , [Node.SetProcessPriority\(int\)](#) , Node.GetProcessPriority() ,  
Node.IsProcessing() , [Node.SetProcessInput\(bool\)](#) , Node.IsProcessingInput() ,  
[Node.SetProcessUnhandledInput\(bool\)](#) , Node.IsProcessingUnhandledInput() ,  
[Node.SetProcessUnhandledKeyInput\(bool\)](#) , Node.IsProcessingUnhandledKeyInput() ,  
Node.SetPauseMode(Node.PauseModeEnum) , Node.GetPauseMode() , Node.CanProcess() ,  
Node.PrintStrayNodes() , NodeGetPositionInParent() , [Node.SetDisplayFolded\(bool\)](#) ,  
Node.IsDisplayedFolded() , [Node.SetProcessInternal\(bool\)](#) , Node.IsProcessingInternal() ,  
[Node.SetPhysicsProcessInternal\(bool\)](#) , Node.IsPhysicsProcessingInternal() ,  
Node.SetPhysicsInterpolationMode(Node.PhysicsInterpolationModeEnum) ,  
Node.GetPhysicsInterpolationMode() , Node.IsPhysicsInterpolated() ,  
Node.IsPhysicsInterpolatedAndEnabled() , Node.ResetPhysicsInterpolation() , Node.GetTree() ,  
Node.CreateTween() , [Node.Duplicate\(int\)](#) , [Node.ReplaceBy\(Node, bool\)](#) ,  
[Node.SetSceneInstanceLoadPlaceholder\(bool\)](#) , Node.GetSceneInstanceLoadPlaceholder() ,  
Node.GetViewport() , Node.QueueFree() , Node.RequestReady() , [Node.SetNetworkMaster\(int, bool\)](#) ,  
Node.GetNetworkMaster() , Node.IsNetworkMaster() , Node.GetMultiplayer() ,  
Node.GetCustomMultiplayer() , Node.SetCustomMultiplayer(MultiplayerAPI) ,  
[Node.RpcConfig\(string, MultiplayerAPI.RPCMode\)](#) ,  
[Node.RsetConfig\(string, MultiplayerAPI.RPCMode\)](#) , [Node.SetUniqueNameInOwner\(bool\)](#) ,  
Node.IsUniqueNameInOwner() , [Node.Rpc\(string, params object\[\]\)](#) ,  
[Node.RpcUnreliable\(string, params object\[\]\)](#) , [Node.Rpcld\(int, string, params object\[\]\)](#) ,  
[Node.RpcUnreliableId\(int, string, params object\[\]\)](#) , [Node.Rset\(string, object\)](#) ,  
[Node.RsetId\(int, string, object\)](#) , [Node.RsetUnreliable\(string, object\)](#) ,  
[Node.RsetUnreliableId\(int, string, object\)](#) , Node.UpdateConfigurationWarning() ,  
Node.EditorDescription , Node.\_ImportPath , Node.PauseMode , Node.PhysicsInterpolationMode ,  
Node.Name , Node.UniqueNameInOwner , Node.Filename , Node.Owner , Node.Multiplayer ,  
Node.CustomMultiplayer , Node.ProcessPriority , Object.NotificationPostInitialize ,  
Object.NotificationPreDelete , Object.IsValid(Object) , Object.WeakRef(Object) , Object.Dispose() ,  
[Object.Dispose\(bool\)](#) , Object.ToString() , [Object.ToSignal\(Object, string\)](#) , [Object.Get\(string\)](#) ,  
Object.\_GetPropertyList() , [Object.Notification\(int\)](#) , [Object.Set\(string, object\)](#) , Object.Free() ,  
Object.GetClass() , [Object.IsClass\(string\)](#) , [Object.Set\(string, object\)](#) , [Object.Get\(string\)](#) ,  
[Object.SetIndexed\(NodePath, object\)](#) , Object.GetIndexed(NodePath) , Object.GetPropertyList() ,  
Object.GetMethodList() , [Object.Notification\(int, bool\)](#) , Object.GetInstanceId() ,  
Object.SetScript(Reference) , Object.GetScript() , [Object.SetMeta\(string, object\)](#) ,  
[Object.RemoveMeta\(string\)](#) , [Object.GetMeta\(string, object\)](#) , [Object.HasMeta\(string\)](#) ,

[Object.GetMetaList\(\)](#) , [Object.AddUserSignal\(string, Array\)](#) , [Object.HasUserSignal\(string\)](#) ,  
[Object.EmitSignal\(string, params object\[\]\)](#) , [Object.Call\(string, params object\[\]\)](#) ,  
[Object.CallDeferred\(string, params object\[\]\)](#) , [Object.SetDeferred\(string, object\)](#) ,  
[Object.Cally\(string, Array\)](#) , [Object.HasMethod\(string\)](#) , [Object.HasSignal\(string\)](#) ,  
Object.GetSignalList() , [Object.GetSignalConnectionList\(string\)](#) , Object.GetIncomingConnections() ,  
[Object.Connect\(string, Object, string, Array, uint\)](#) , [Object.Disconnect\(string, Object, string\)](#) ,  
[Object.IsConnected\(string, Object, string\)](#) , [Object.SetBlockSignals\(bool\)](#) , Object.IsBlockingSignals() ,  
Object.PropertyListChangedNotify() , [Object.SetMessageTranslation\(bool\)](#) ,  
Object.CanTranslateMessages() , [Object.Tr\(string\)](#) , Object.IsQueuedForDeletion() ,  
Object.NativeInstance , Object.DynamicObject , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Extension Methods

[Node GD CB Extension.FindNodeByName\(Node, string\)](#) ,  
[Node GD CB Extension.FindNodeByName\(Node, string, bool\)](#) ,  
[Node GD CB Extension.FindNodeByName\(Node, string, Type, bool\)](#) ,  
[Node GD CB Extension.FindNodeByName<T>\(Node, string\)](#) ,  
[Node GD CB Extension.FindNodeByName<T>\(Node, string, bool\)](#) ,  
[Node GD CB Extension.FindNodes\(Node, Type\)](#) , [Node GD CB Extension.FindNodes\(Node, Type, bool\)](#) ,  
[Node GD CB Extension.FindNodes<T>\(Node\)](#) , [Node GD CB Extension.FindNodes<T>\(Node, bool\)](#) ,  
[Node GD CB Extension.GetNodePosition\(Node\)](#) , [Node GD CB Extension.GetNodeRotation\(Node\)](#) ,  
[Node GD CB Extension.GetNodeScale\(Node\)](#) , [Node GD CB Extension.Print\(Node, params object\[\]\)](#) ,  
[Node GD CB Extension.SetNodePosition\(Node, Vector3D\)](#) ,  
[Node GD CB Extension.SetNodeRotation\(Node, Vector3D\)](#) ,  
[Node GD CB Extension.SetNodeScale\(Node, Vector3D\)](#)

## Fields

### LoadedScene

```
public static Action<Scene>? LoadedScene
```

### Field Value

[Action<Scene>](#)

# UnloadedScene

```
public static Action<Scene>? UnloadedScene
```

Field Value

[Action](#) <[Scene](#)>

## Properties

### CurrentScene

```
public static Scene CurrentScene { get; }
```

Property Value

[Scene](#)

### CurrentSceneNode

```
public static Node? CurrentSceneNode { get; }
```

Property Value

Node

## Methods

### DontDestroyOnLoad(Node)

```
public static void DontDestroyOnLoad(Node obj)
```

Parameters

**obj** Node

## LoadScene(int)

```
public static bool LoadScene(int index)
```

Parameters

index [int](#)

Returns

[bool](#)

## LoadScene(string)

```
public static bool LoadScene(string name)
```

Parameters

name [string](#)

Returns

[bool](#)

## \_Ready()

Called when the node is "ready", i.e. when both the node and its children have entered the scene tree. If the node has children, their Godot.Node.\_Ready() callbacks get triggered first, and the parent node will receive the ready notification afterwards.

Corresponds to the Godot.Node.NotificationReady notification in [Notification\(int\)](#). See also the **onready** keyword for variables.

Usually used for initialization. For even earlier initialization, may be used. See also Godot.Node.\_EnterTree().

Note: Godot.Node.\_Ready() may be called only once for each node. After removing a node from the scene tree and adding it again, `_ready` will not be called a second time. This can be bypassed by requesting another call with Godot.Node.RequestReady(), which may be called anywhere before adding the node again.

```
public override void _Ready()
```

# Namespace Godot

## Classes

[Camera2D\\_GD\\_CB\\_Extension](#)

[Node\\_GD\\_CB\\_Extension](#)

[Rect2\\_GD\\_CB\\_Extension](#)

# Class Camera2D\_GD\_CB\_Extension

Namespace: [Godot](#)

Assembly: com.cobilas.godot.utility.dll

```
public static class Camera2D_GD_CB_Extension
```

## Inheritance

[object](#) ← Camera2D\_GD\_CB\_Extension

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Methods

### ScreenToWorldPoint(Camera2D, Vector2)

```
public static Vector2 ScreenToWorldPoint(this Camera2D C, Vector2 mousePosition)
```

#### Parameters

C Camera2D

mousePosition Vector2

#### Returns

Vector2

### WorldToScreenPoint(Camera2D, Vector2)

```
public static Vector2 WorldToScreenPoint(this Camera2D C, Vector2 position)
```

## Parameters

C Camera2D

position Vector2

## Returns

Vector2

# Class Node\_GD\_CB\_Extension

Namespace: [Godot](#)

Assembly: com.cobilas.godot.utility.dll

```
public static class Node_GD_CB_Extension
```

## Inheritance

[object](#) ← Node\_GD\_CB\_Extension

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Methods

### FindNodeByName(Node, string)

Get a node from name.

By default, the method looks for a node of type node.([Type](#) typeNode = typeof(Godot.Node))

By default, the method searches recursively.([bool](#) recursive = true)

```
public static Node? FindNodeByName(this Node N, string name)
```

## Parameters

N Node

name [string](#)

The node name

## Returns

Node

## FindNodeByName(Node, string, bool)

Get a node from name.

By default, the method looks for a node of type node.([Type](#) `typeNode = typeof(Godot.Node)`)

```
public static Node? FindNodeByName(this Node N, string name, bool recursive)
```

### Parameters

**N** Node

**name** [string](#)

The node name

**recusive** [bool](#)

Also look for your children.

### Returns

Node

## FindNodeByName(Node, string, Type, bool)

Get a node from name.

```
public static Node? FindNodeByName(this Node N, string name, Type typeNode, bool recursive)
```

### Parameters

**N** Node

**name** [string](#)

The node name

**typeNode** [Type](#)

The type to look for.

`recusive` `bool`

Also look for your children.

Returns

Node

## FindNodeByName<T>(Node, string)

Get a node from name.

By default, the method searches recursively.(`bool` `recusive = true`)

```
public static T FindNodeByName<T>(this Node N, string name) where T : Node
```

Parameters

`N` Node

`name` `string`

The node name

Returns

T

Type Parameters

`T`

## FindNodeByName<T>(Node, string, bool)

Get a node from name.

```
public static T FindNodeByName<T>(this Node N, string name, bool recursive) where T : Node
```

Parameters

N Node

name [string](#)

The node name

recusive [bool](#)

Also look for your children.

Returns

T

Type Parameters

T

The type to look for.

## FindNodes(Node, Type)

Get the nodes from a type.

By default, the method searches recursively. ([bool](#) recusive = true)

```
public static Node[] FindNodes(this Node N, Type typeNode)
```

Parameters

N Node

typeNode [Type](#)

The type to look for.

Returns

Node[]

Returns a list of nodes.

## FindNodes(Node, Type, bool)

Get the nodes from a type.

```
public static Node[] FindNodes(this Node N, Type typeNode, bool recursive)
```

Parameters

**N** Node

**typeNode** [Type](#)

The type to look for.

**recusive** [bool](#)

Also look for your children.

Returns

Node[]

Returns a list of nodes.

## FindNodes<T>(Node)

Get the nodes from a type.

By default, the method searches recursively.([bool](#) recursive = true)

```
public static T[] FindNodes<T>(this Node N) where T : Node
```

Parameters

**N** Node

Returns

T[]

Returns a list of nodes.

## Type Parameters

T

The type to look for.

## FindNodes<T>(Node, bool)

Get the nodes from a type.

```
public static T[] FindNodes<T>(this Node N, bool recursive) where T : Node
```

### Parameters

N Node

recursive [bool](#)

Also look for your children.

### Returns

T[]

Returns a list of nodes.

## Type Parameters

T

The type to look for.

## GetNodePosition(Node)

```
public static Vector3D GetNodePosition(this Node N)
```

### Parameters

N Node

Returns

[Vector3D](#)

## GetNodeRotation(Node)

```
public static Vector3D GetNodeRotation(this Node N)
```

Parameters

N Node

Returns

[Vector3D](#)

## GetNodeScale(Node)

```
public static Vector3D GetNodeScale(this Node N)
```

Parameters

N Node

Returns

[Vector3D](#)

## Print(Node, params object[])

```
public static void Print(this Node N, params object[] args)
```

Parameters

N Node

args [object](#)[]

## SetNodePosition(Node, Vector3D)

```
public static void SetNodePosition(this Node N, Vector3D position)
```

### Parameters

N Node

position [Vector3D](#)

## SetNodeRotation(Node, Vector3D)

```
public static void SetNodeRotation(this Node N, Vector3D rotation)
```

### Parameters

N Node

rotation [Vector3D](#)

## SetNodeScale(Node, Vector3D)

```
public static void SetNodeScale(this Node N, Vector3D scale)
```

### Parameters

N Node

scale [Vector3D](#)

# Class Rect2\_GD\_CB\_Extension

Namespace: [Godot](#)

Assembly: com.cobilas.godot.utility.dll

```
public static class Rect2_GD_CB_Extension
```

## Inheritance

[object](#) ← Rect2\_GD\_CB\_Extension

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Methods

### Bottom(Rect2)

```
public static float Bottom(this Rect2 R)
```

#### Parameters

R Rect2

#### Returns

[float](#)

### Center(Rect2)

```
[Obsolete("Use Rect2.GetCenter()")]
public static Vector2 Center(this Rect2 R)
```

#### Parameters

R Rect2

Returns

Vector2

## Left(Rect2)

```
public static float Left(this Rect2 R)
```

Parameters

R Rect2

Returns

[float](#)

## Right(Rect2)

```
public static float Right(this Rect2 R)
```

Parameters

R Rect2

Returns

[float](#)

## Top(Rect2)

```
public static float Top(this Rect2 R)
```

Parameters

R Rect2

Returns

[float](#)