Namespace DG.Tweening

Classes

DOTweenCYInstruction

DOTweenCYInstruction.WaitForCompletion

DOTweenCYInstruction.WaitForElapsedLoops

DOTweenCYInstruction.WaitForKill

DOTweenCYInstruction.WaitForPosition

DOTweenCYInstruction.WaitForRewind

DOTweenCYInstruction.WaitForStart

DOTweenModuleAudio

DOTweenModulePhysics2D

DOTweenModuleSprite

DOTweenModuleUI

DOTweenModuleUI.Utils

DOTweenModuleUnityVersion

Shortcuts/functions that are not strictly related to specific Modules but are available only on some Unity versions

DOTweenModuleUtils

Utility functions that deal with available Modules. Modules defines:

- DOTAUDIO
- DOTPHYSICS
- DOTPHYSICS2D
- DOTSPRITE
- DOTUI Extra defines set and used for implementation of external assets:
- DOTWEEN_TMP ► TextMesh Pro
- DOTWEEN_TK2D ► 2D Toolkit

DOTweenModuleUtils.Physics

Class DOTweenCYInstruction

Namespace: DG.Tweening

Assembly: MasterMot.dll

```
public static class DOTweenCYInstruction
```

Inheritance

object ← DOTweenCYInstruction

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Class DOTweenCYInstruction.WaitForCompletion

Namespace: DG.Tweening

Assembly: MasterMot.dll

```
public class DOTweenCYInstruction.WaitForCompletion : CustomYieldInstruction, IEnumerator
```

Inheritance

object ← CustomYieldInstruction ← DOTweenCYInstruction.WaitForCompletion

Implements

IEnumerator

Inherited Members

CustomYieldInstruction.MoveNext() , CustomYieldInstruction.Reset() , CustomYieldInstruction.Current , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

WaitForCompletion(Tween)

```
public WaitForCompletion(Tween tween)
```

Parameters

`tween` Tween

Properties

keepWaiting

Indicates if coroutine should be kept suspended.

```
public override bool keepWaiting { get; }
```

Property Value

bool

Class DOTweenCYInstruction.WaitForElapsedLoops

Namespace: DG.Tweening

Assembly: MasterMot.dll

```
public class DOTweenCYInstruction.WaitForElapsedLoops : CustomYieldInstruction, IEnumerator
```

Inheritance

object ← CustomYieldInstruction ← DOTweenCYInstruction.WaitForElapsedLoops

Implements

IEnumerator

Inherited Members

CustomYieldInstruction.MoveNext() , CustomYieldInstruction.Reset() , CustomYieldInstruction.Current , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

WaitForElapsedLoops(Tween, int)

```
public WaitForElapsedLoops(Tween tween, int elapsedLoops)
```

Parameters

`tween` Tween

`elapsedLoops` int

Properties

keepWaiting

Indicates if coroutine should be kept suspended.

```
public override bool keepWaiting { get; }
```

Property Value

bool

Class DOTweenCYInstruction.WaitForKill

Namespace: DG.Tweening

Assembly: MasterMot.dll

```
public class DOTweenCYInstruction.WaitForKill : CustomYieldInstruction, IEnumerator
```

Inheritance

object ← CustomYieldInstruction ← DOTweenCYInstruction.WaitForKill

Implements

IEnumerator

Inherited Members

CustomYieldInstruction.MoveNext() , CustomYieldInstruction.Reset() , CustomYieldInstruction.Current , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

WaitForKill(Tween)

```
public WaitForKill(Tween tween)
```

Parameters

`tween` Tween

Properties

keepWaiting

Indicates if coroutine should be kept suspended.

```
public override bool keepWaiting { get; }
```

Property Value

bool

Class DOTweenCYInstruction.WaitForPosition

Namespace: DG.Tweening

Assembly: MasterMot.dll

```
public class DOTweenCYInstruction.WaitForPosition : CustomYieldInstruction, IEnumerator
```

Inheritance

object ← CustomYieldInstruction ← DOTweenCYInstruction.WaitForPosition

Implements

IEnumerator

Inherited Members

CustomYieldInstruction.MoveNext() , CustomYieldInstruction.Reset() , CustomYieldInstruction.Current , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

WaitForPosition(Tween, float)

```
public WaitForPosition(Tween tween, float position)
```

Parameters

`tween` Tween

`position` float

Properties

keepWaiting

Indicates if coroutine should be kept suspended.

```
public override bool keepWaiting { get; }
```

Property Value

bool

Class DOTweenCYInstruction.WaitForRewind

Namespace: DG.Tweening

Assembly: MasterMot.dll

```
public class DOTweenCYInstruction.WaitForRewind : CustomYieldInstruction, IEnumerator
```

Inheritance

object ← CustomYieldInstruction ← DOTweenCYInstruction.WaitForRewind

Implements

IEnumerator

Inherited Members

CustomYieldInstruction.MoveNext() , CustomYieldInstruction.Reset() , CustomYieldInstruction.Current , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

WaitForRewind(Tween)

```
public WaitForRewind(Tween tween)
```

Parameters

`tween` Tween

Properties

keepWaiting

Indicates if coroutine should be kept suspended.

```
public override bool keepWaiting { get; }
```

Property Value

bool

Class DOTweenCYInstruction.WaitForStart

Namespace: DG.Tweening

Assembly: MasterMot.dll

```
public class DOTweenCYInstruction.WaitForStart : CustomYieldInstruction, IEnumerator
```

Inheritance

object ← CustomYieldInstruction ← DOTweenCYInstruction.WaitForStart

Implements

IEnumerator

Inherited Members

CustomYieldInstruction.MoveNext() , CustomYieldInstruction.Reset() , CustomYieldInstruction.Current , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

WaitForStart(Tween)

```
public WaitForStart(Tween tween)
```

Parameters

`tween` Tween

Properties

keepWaiting

Indicates if coroutine should be kept suspended.

```
public override bool keepWaiting { get; }
```

Property Value

bool

Class DOTweenModuleAudio

Namespace: DG.Tweening

Assembly: MasterMot.dll

```
public static class DOTweenModuleAudio
```

Inheritance

object ← DOTweenModuleAudio

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

DOComplete(AudioMixer, bool)

Completes all tweens that have this target as a reference (meaning tweens that were started from this target, or that had this target added as an Id) and returns the total number of tweens completed (meaning the tweens that don't have infinite loops and were not already complete)

```
public static int DOComplete(this AudioMixer target, bool withCallbacks = false)
```

Parameters

`target` AudioMixer

`withCallbacks` bool

For Sequences only: if TRUE also internal Sequence callbacks will be fired, otherwise they will be ignored

Returns

int

DOFade(AudioSource, float, float)

Tweens an AudioSource's volume to the given value. Also stores the AudioSource as the tween's target so it can be used for filtered operations

```
public static TweenerCore<float, float, FloatOptions> DOFade(this AudioSource target, float
endValue, float duration)
```

Parameters

`target` AudioSource

`endValue` float

The end value to reach (0 to 1)

`duration` float

The duration of the tween

Returns

TweenerCore<float, float, FloatOptions>

DOFlip(AudioMixer)

Flips the direction (backwards if it was going forward or viceversa) of all tweens that have this target as a reference (meaning tweens that were started from this target, or that had this target added as an Id) and returns the total number of tweens flipped.

```
public static int DOFlip(this AudioMixer target)
```

Parameters

`target` AudioMixer

Returns

int

DOGoto(AudioMixer, float, bool)

Sends to the given position all tweens that have this target as a reference (meaning tweens that were started from this target, or that had this target added as an Id) and returns the total number of tweens involved.

```
public static int DOGoto(this AudioMixer target, float to, bool andPlay = false)
```

Parameters

`target` AudioMixer

`to` float

Time position to reach (if higher than the whole tween duration the tween will simply reach its end)

`andPlay` bool

If TRUE will play the tween after reaching the given position, otherwise it will pause it

Returns

int

DOKill(AudioMixer, bool)

Kills all tweens that have this target as a reference (meaning tweens that were started from this target, or that had this target added as an Id) and returns the total number of tweens killed.

```
public static int DOKill(this AudioMixer target, bool complete = false)
```

Parameters

`target` AudioMixer

`complete` bool

If TRUE completes the tween before killing it

Returns

int

DOPause(AudioMixer)

Pauses all tweens that have this target as a reference (meaning tweens that were started from this target, or that had this target added as an Id) and returns the total number of tweens paused.

```
public static int DOPause(this AudioMixer target)
```

Parameters

`target` AudioMixer

Returns

int

DOPitch(AudioSource, float, float)

Tweens an AudioSource's pitch to the given value. Also stores the AudioSource as the tween's target so it can be used for filtered operations

```
public static TweenerCore<float, float, FloatOptions> DOPitch(this AudioSource target, float
endValue, float duration)
```

Parameters

`target` AudioSource

`endValue` float

The end value to reach

`duration` float

The duration of the tween

Returns

TweenerCore<float, float, FloatOptions>

DOPlay(AudioMixer)

Plays all tweens that have this target as a reference (meaning tweens that were started from this target, or that had this target added as an Id) and returns the total number of tweens played.

```
public static int DOPlay(this AudioMixer target)
```

Parameters

`target` AudioMixer

Returns

int

DOPlayBackwards(AudioMixer)

Plays backwards all tweens that have this target as a reference (meaning tweens that were started from this target, or that had this target added as an Id) and returns the total number of tweens played.

```
public static int DOPlayBackwards(this AudioMixer target)
```

Parameters

`target` AudioMixer

Returns

int

DOPlayForward(AudioMixer)

Plays forward all tweens that have this target as a reference (meaning tweens that were started from this target, or that had this target added as an Id) and returns the total number of tweens played.

```
public static int DOPlayForward(this AudioMixer target)
```

Parameters

`target` AudioMixer

Returns

int

DORestart(AudioMixer)

Restarts all tweens that have this target as a reference (meaning tweens that were started from this target, or that had this target added as an Id) and returns the total number of tweens restarted.

```
public static int DORestart(this AudioMixer target)
```

Parameters

`target` AudioMixer

Returns

int

DORewind(AudioMixer)

Rewinds all tweens that have this target as a reference (meaning tweens that were started from this target, or that had this target added as an Id) and returns the total number of tweens rewinded.

```
public static int DORewind(this AudioMixer target)
```

Parameters

`target` AudioMixer

Returns

int

DOSetFloat(AudioMixer, string, float, float)

Tweens an AudioMixer's exposed float to the given value. Also stores the AudioMixer as the tween's target so it can be used for filtered operations. Note that you need to manually expose a float in an AudioMixerGroup in order to be able to tween it from an AudioMixer.

```
public static TweenerCore<float, float, FloatOptions> DOSetFloat(this AudioMixer target,
string floatName, float endValue, float duration)
```

Parameters

`target` AudioMixer

`floatName` string

Name given to the exposed float to set

`endValue` float

The end value to reach

`duration` float

The duration of the tween

Returns

TweenerCore<float, float, FloatOptions>

DOSmoothRewind(AudioMixer)

Smoothly rewinds all tweens that have this target as a reference (meaning tweens that were started from this target, or that had this target added as an Id) and returns the total number of tweens rewinded.

```
public static int DOSmoothRewind(this AudioMixer target)
```

Parameters

`target` AudioMixer

Returns

int

DOTogglePause(AudioMixer)

Toggles the paused state (plays if it was paused, pauses if it was playing) of all tweens that have this target as a reference (meaning tweens that were started from this target, or that had this target added as an Id) and returns the total number of tweens involved.

```
public static int DOTogglePause(this AudioMixer target)
```

Parameters

`target` AudioMixer

Returns

int

Class DOTweenModulePhysics2D

Namespace: DG.Tweening

Assembly: MasterMot.dll

```
public static class DOTweenModulePhysics2D
```

Inheritance

object ← DOTweenModulePhysics2D

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

DOJump(Rigidbody2D, Vector2, float, int, float, bool)

Tweens a Rigidbody2D's position to the given value, while also applying a jump effect along the Y axis. Returns a Sequence instead of a Tweener. Also stores the Rigidbody2D as the tween's target so it can be used for filtered operations.

IMPORTANT: a rigidbody2D can't be animated in a jump arc using MovePosition, so the tween will directly set the position

```
public static Sequence DOJump(this Rigidbody2D target, Vector2 endValue, float jumpPower,
int numJumps, float duration, bool snapping = false)
```

Parameters

`target` Rigidbody2D

`endValue` Vector2

The end value to reach

`jumpPower` float

Power of the jump (the max height of the jump is represented by this plus the final Y offset) `numJumps` int

Total number of jumps

`duration` float

The duration of the tween

`snapping` bool

If TRUE the tween will smoothly snap all values to integers

Returns

Sequence

DOLocalPath(Rigidbody2D, Vector2[], float, PathType, PathMode, int, Color?)

Tweens a Rigidbody2D's localPosition through the given path waypoints, using the chosen path algorithm. Also stores the Rigidbody2D as the tween's target so it can be used for filtered operations

NOTE: to tween a Rigidbody2D correctly it should be set to kinematic at least while being tweened.

BEWARE: doesn't work on Windows Phone store (waiting for Unity to fix their own bug). If you plan to publish there you should use a regular transform.DOLocalPath.

```
public static TweenerCore<Vector3, Path, PathOptions> DOLocalPath(this Rigidbody2D target,
Vector2[] path, float duration, PathType pathType = PathType.Linear, PathMode pathMode =
PathMode.Full3D, int resolution = 10, Color? gizmoColor = null)
```

Parameters

`target` Rigidbody2D

`path` Vector2[]

The waypoint to go through

`duration` float

The duration of the tween

`pathType` PathType

The type of path: Linear (straight path), CatmullRom (curved CatmullRom path) or CubicBezier (curved with control points)

`pathMode` PathMode

The path mode: 3D, side-scroller 2D, top-down 2D

`resolution` int

The resolution of the path: higher resolutions make for more detailed curved paths but are more expensive. Defaults to 10, but a value of 5 is usually enough if you don't have dramatic long curves between waypoints

`gizmoColor` Color?

The color of the path (shown when gizmos are active in the Play panel and the tween is running)

Returns

TweenerCore<Vector3, Path, PathOptions>

DOMove(Rigidbody2D, Vector2, float, bool)

Tweens a Rigidbody2D's position to the given value. Also stores the Rigidbody2D as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Vector2, Vector2, VectorOptions> DOMove(this Rigidbody2D target,
Vector2 endValue, float duration, bool snapping = false)
```

Parameters

`target` Rigidbody2D

`endValue` Vector2

The end value to reach

`duration` float

The duration of the tween

`snapping` [bool]

If TRUE the tween will smoothly snap all values to integers

Returns

TweenerCore<Vector2, Vector2, VectorOptions>

DOMoveX(Rigidbody2D, float, float, bool)

Tweens a Rigidbody2D's X position to the given value. Also stores the Rigidbody2D as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Vector2, Vector2, VectorOptions> DOMoveX(this Rigidbody2D target,
float endValue, float duration, bool snapping = false)
```

Parameters

`target` Rigidbody2D

`endValue` [float]

The end value to reach

`duration` [float]

The duration of the tween

`snapping` [bool]

If TRUE the tween will smoothly snap all values to integers

Returns

TweenerCore<Vector2, Vector2, VectorOptions>

DOMoveY(Rigidbody2D, float, float, bool)

Tweens a Rigidbody2D's Y position to the given value. Also stores the Rigidbody2D as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Vector2, Vector2, VectorOptions> DOMoveY(this Rigidbody2D target,
float endValue, float duration, bool snapping = false)
```

Parameters

`target` Rigidbody2D

`endValue` float

The end value to reach

`duration` float

The duration of the tween

`snapping` bool

If TRUE the tween will smoothly snap all values to integers

Returns

TweenerCore<Vector2, Vector2, VectorOptions>

DOPath(Rigidbody2D, Vector2[], float, PathType, PathMode, int, Color?)

Tweens a Rigidbody2D's position through the given path waypoints, using the chosen path algorithm. Also stores the Rigidbody2D as the tween's target so it can be used for filtered operations.

NOTE: to tween a Rigidbody2D correctly it should be set to kinematic at least while being tweened.

BEWARE: doesn't work on Windows Phone store (waiting for Unity to fix their own bug). If you plan to publish there you should use a regular transform.DOPath.

```
public static TweenerCore<Vector3, Path, PathOptions> DOPath(this Rigidbody2D target,
Vector2[] path, float duration, PathType pathType = PathType.Linear, PathMode pathMode =
PathMode.Full3D, int resolution = 10, Color? gizmoColor = null)
```

Parameters

`target` Rigidbody2D

`path` Vector2[]

The waypoints to go through

`duration` float

The duration of the tween

`pathType` PathType

The type of path: Linear (straight path), CatmullRom (curved CatmullRom path) or CubicBezier (curved with control points)

`pathMode` PathMode

The path mode: 3D, side-scroller 2D, top-down 2D

`resolution` int

The resolution of the path (useless in case of Linear paths): higher resolutions make for more detailed curved paths but are more expensive. Defaults to 10, but a value of 5 is usually enough if you don't have dramatic long curves between waypoints

`gizmoColor` Color?

The color of the path (shown when gizmos are active in the Play panel and the tween is running)

Returns

TweenerCore<Vector3, Path, PathOptions>

DORotate(Rigidbody2D, float, float)

Tweens a Rigidbody2D's rotation to the given value. Also stores the Rigidbody2D as the tween's target so it can be used for filtered operations

```
public static TweenerCore<float, float, FloatOptions> DORotate(this Rigidbody2D target,
float endValue, float duration)
```

Parameters

`target` Rigidbody2D

`endValue` float

The end value to reach

`duration` float

The duration of the tween

Returns

TweenerCore<float, float, FloatOptions>

Class DOTweenModuleSprite

Namespace: DG.Tweening

Assembly: MasterMot.dll

```
public static class DOTweenModuleSprite
```

Inheritance

object ← DOTweenModuleSprite

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

DOBlendableColor(SpriteRenderer, Color, float)

Tweens a SpriteRenderer's color to the given value, in a way that allows other DOBlendableColor tweens to work together on the same target, instead than fight each other as multiple DOColor would do. Also stores the SpriteRenderer as the tween's target so it can be used for filtered operations

```
public static Tweener DOBlendableColor(this SpriteRenderer target, Color endValue,
float duration)
```

Parameters

`target` SpriteRenderer

`endValue` Color

The value to tween to

`duration` float

The duration of the tween

Returns

Tweener

DOColor(SpriteRenderer, Color, float)

Tweens a SpriteRenderer's color to the given value. Also stores the spriteRenderer as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Color, Color, ColorOptions> DOColor(this SpriteRenderer target,
Color endValue, float duration)
```

Parameters

`target` SpriteRenderer

`endValue` Color

The end value to reach

`duration` float

The duration of the tween

Returns

TweenerCore<Color, Color, ColorOptions>

DOFade(SpriteRenderer, float, float)

Tweens a Material's alpha color to the given value. Also stores the spriteRenderer as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Color, Color, ColorOptions> DOFade(this SpriteRenderer target,
float endValue, float duration)
```

Parameters

`target` SpriteRenderer

`endValue` float

The end value to reach

`duration` float

The duration of the tween

Returns

TweenerCore<Color, Color, ColorOptions>

DOGradientColor(SpriteRenderer, Gradient, float)

Tweens a SpriteRenderer's color using the given gradient (NOTE 1: only uses the colors of the gradient, not the alphas - NOTE 2: creates a Sequence, not a Tweener). Also stores the image as the tween's target so it can be used for filtered operations

```
public static Sequence DOGradientColor(this SpriteRenderer target, Gradient gradient,
float duration)
```

Parameters

`target` SpriteRenderer

`gradient` Gradient

The gradient to use

`duration` float

The duration of the tween

Returns

Sequence

Class DOTweenModuleUI

Namespace: DG.Tweening

Assembly: MasterMot.dll

```
public static class DOTweenModuleUI
```

Inheritance

object ← DOTweenModuleUI

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

DOAnchorMax(RectTransform, Vector2, float, bool)

Tweens a RectTransform's anchorMax to the given value. Also stores the RectTransform as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Vector2, Vector2, VectorOptions> DOAnchorMax(this RectTransform
target, Vector2 endValue, float duration, bool snapping = false)
```

Parameters

`target` RectTransform

`endValue` Vector2

The end value to reach

`duration` float

The duration of the tween

`snapping` bool

If TRUE the tween will smoothly snap all values to integers

Returns

TweenerCore<Vector2, Vector2, VectorOptions>

DOAnchorMin(RectTransform, Vector2, float, bool)

Tweens a RectTransform's anchorMin to the given value. Also stores the RectTransform as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Vector2, Vector2, VectorOptions> DOAnchorMin(this RectTransform
target, Vector2 endValue, float duration, bool snapping = false)
```

Parameters

`target` RectTransform

`endValue` Vector2

The end value to reach

`duration` float

The duration of the tween

`snapping` bool

If TRUE the tween will smoothly snap all values to integers

Returns

TweenerCore<Vector2, Vector2, VectorOptions>

DOAnchorPos(RectTransform, Vector2, float, bool)

Tweens a RectTransform's anchoredPosition to the given value. Also stores the RectTransform as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Vector2, Vector2, VectorOptions> DOAnchorPos(this RectTransform
target, Vector2 endValue, float duration, bool snapping = false)
```

Parameters

`target` RectTransform

`endValue` Vector2

The end value to reach

`duration` float

The duration of the tween

`snapping` bool

If TRUE the tween will smoothly snap all values to integers

Returns

TweenerCore<Vector2, Vector2, VectorOptions>

DOAnchorPos3D(RectTransform, Vector3, float, bool)

Tweens a RectTransform's anchoredPosition3D to the given value. Also stores the RectTransform as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Vector3, Vector3, VectorOptions> DOAnchorPos3D(this RectTransform
target, Vector3 endValue, float duration, bool snapping = false)
```

Parameters

`target` RectTransform

`endValue` Vector3

The end value to reach

`duration` float

The duration of the tween

`snapping` bool

If TRUE the tween will smoothly snap all values to integers

Returns

TweenerCore<Vector3, Vector3, VectorOptions>

DOAnchorPos3DX(RectTransform, float, float, bool)

Tweens a RectTransform's anchoredPosition3D X to the given value. Also stores the RectTransform as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Vector3, Vector3, VectorOptions> DOAnchorPos3DX(this RectTransform
target, float endValue, float duration, bool snapping = false)
```

Parameters

`target` RectTransform

`endValue` float

The end value to reach

`duration` float

The duration of the tween

`snapping` bool

If TRUE the tween will smoothly snap all values to integers

Returns

TweenerCore<Vector3, Vector3, VectorOptions>

DOAnchorPos3DY(RectTransform, float, float, bool)

Tweens a RectTransform's anchoredPosition3D Y to the given value. Also stores the RectTransform as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Vector3, Vector3, VectorOptions> DOAnchorPos3DY(this RectTransform
target, float endValue, float duration, bool snapping = false)
```

Parameters

`target` RectTransform

`endValue` float

The end value to reach

`duration` float

The duration of the tween

`snapping` bool

If TRUE the tween will smoothly snap all values to integers

Returns

TweenerCore<Vector3, Vector3, VectorOptions>

DOAnchorPos3DZ(RectTransform, float, float, bool)

Tweens a RectTransform's anchoredPosition3D Z to the given value. Also stores the RectTransform as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Vector3, Vector3, VectorOptions> DOAnchorPos3DZ(this RectTransform
target, float endValue, float duration, bool snapping = false)
```

Parameters

`target` RectTransform

`endValue` float

The end value to reach

`duration` float

The duration of the tween

`snapping` bool

If TRUE the tween will smoothly snap all values to integers

Returns

TweenerCore<Vector3, Vector3, VectorOptions>

DOAnchorPosX(RectTransform, float, float, bool)

Tweens a RectTransform's anchoredPosition X to the given value. Also stores the RectTransform as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Vector2, Vector2, VectorOptions> DOAnchorPosX(this RectTransform
target, float endValue, float duration, bool snapping = false)
```

Parameters

`target` RectTransform

`endValue` float

The end value to reach

`duration` float

The duration of the tween

`snapping` bool

If TRUE the tween will smoothly snap all values to integers

Returns

TweenerCore<Vector2, Vector2, VectorOptions>

DOAnchorPosY(RectTransform, float, float, bool)

Tweens a RectTransform's anchoredPosition Y to the given value. Also stores the RectTransform as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Vector2, Vector2, VectorOptions> DOAnchorPosY(this RectTransform
target, float endValue, float duration, bool snapping = false)
```

Parameters

`target` RectTransform

`endValue` float

The end value to reach

`duration` float

The duration of the tween

`snapping` bool

If TRUE the tween will smoothly snap all values to integers

Returns

TweenerCore<Vector2, Vector2, VectorOptions>

DOBlendableColor(Graphic, Color, float)

Tweens a Graphic's color to the given value, in a way that allows other DOBlendableColor tweens to work together on the same target, instead than fight each other as multiple DOColor would do. Also stores the Graphic as the tween's target so it can be used for filtered operations

```
public static Tweener DOBlendableColor(this Graphic target, Color endValue, float duration)
```

Parameters

`target` Graphic

`endValue` Color

The value to tween to

`duration` float

The duration of the tween

Returns

Tweener

DOBlendableColor(Image, Color, float)

Tweens a Image's color to the given value, in a way that allows other DOBlendableColor tweens to work together on the same target, instead than fight each other as multiple DOColor would do. Also stores the Image as the tween's target so it can be used for filtered operations

```
public static Tweener DOBlendableColor(this Image target, Color endValue, float duration)
```

Parameters

`target` Image

`endValue` Color

The value to tween to

`duration` float

The duration of the tween

Returns

Tweener

DOBlendableColor(Text, Color, float)

Tweens a Text's color BY the given value, in a way that allows other DOBlendableColor tweens to work together on the same target, instead than fight each other as multiple DOColor would do. Also stores the Text as the tween's target so it can be used for filtered operations

```
public static Tweener DOBlendableColor(this Text target, Color endValue, float duration)
```

Parameters

`target` Text

`endValue` Color

The value to tween to

`duration` float

The duration of the tween

Returns

Tweener

DOColor(Graphic, Color, float)

Tweens an Graphic's color to the given value. Also stores the image as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Color, Color, ColorOptions> DOColor(this Graphic target, Color
endValue, float duration)
```

Parameters

`target` Graphic

`endValue` Color

The end value to reach

`duration` float

The duration of the tween

Returns

TweenerCore<Color, Color, ColorOptions>

DOColor(Image, Color, float)

Tweens an Image's color to the given value. Also stores the image as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Color, Color, ColorOptions> DOColor(this Image target, Color
endValue, float duration)
```

Parameters

`target` Image

`endValue` Color

The end value to reach

`duration` [float](#)

The duration of the tween

Returns

TweenerCore<Color, Color, ColorOptions>

DOColor(Outline, Color, float)

Tweens a Outline's effectColor to the given value. Also stores the Outline as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Color, Color, ColorOptions> DOColor(this Outline target, Color
endValue, float duration)
```

Parameters

`target` Outline

`endValue` Color

The end value to reach

`duration` [float](#)

The duration of the tween

Returns

TweenerCore<Color, Color, ColorOptions>

DOColor(Text, Color, float)

Tweens a Text's color to the given value. Also stores the Text as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Color, Color, ColorOptions> DOColor(this Text target, Color
endValue, float duration)
```

Parameters

`target` Text

`endValue` Color

The end value to reach

`duration` float

The duration of the tween

Returns

TweenerCore<Color, Color, ColorOptions>

DOCounter(Text, int, int, float, bool, CultureInfo)

Tweens a Text's text from one integer to another, with options for thousands separators

```
public static TweenerCore<int, int, NoOptions> DOCounter(this Text target, int fromValue,
int endValue, float duration, bool addThousandsSeparator = true, CultureInfo culture = null)
```

Parameters

`target` Text

`fromValue` int

The value to start from

`endValue` int

The end value to reach

`duration` float

The duration of the tween

`addThousandsSeparator` bool

If TRUE (default) also adds thousands separators

`culture` CultureInfo

The CultureInfo to use (InvariantCulture if NULL)

Returns

TweenerCore<int, int, NoOptions>

DOFade(CanvasGroup, float, float)

Tweens a CanvasGroup's alpha color to the given value. Also stores the canvasGroup as the tween's target so it can be used for filtered operations

```
public static TweenerCore<float, float, FloatOptions> DOFade(this CanvasGroup target, float
endValue, float duration)
```

Parameters

`target` CanvasGroup

`endValue` float

The end value to reach

`duration` float

The duration of the tween

Returns

TweenerCore<float, float, FloatOptions>

DOFade(Graphic, float, float)

Tweens an Graphic's alpha color to the given value. Also stores the image as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Color, Color, ColorOptions> DOFade(this Graphic target, float
endValue, float duration)
```

Parameters

`target` Graphic

`endValue` float

The end value to reach

`duration` float

The duration of the tween

Returns

TweenerCore<Color, Color, ColorOptions>

DOFade(Image, float, float)

Tweens an Image's alpha color to the given value. Also stores the image as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Color, Color, ColorOptions> DOFade(this Image target, float
endValue, float duration)
```

Parameters

`target` Image

`endValue` float

The end value to reach

`duration` float

The duration of the tween

Returns

TweenerCore<Color, Color, ColorOptions>

DOFade(Outline, float, float)

Tweens a Outline's effectColor alpha to the given value. Also stores the Outline as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Color, Color, ColorOptions> DOFade(this Outline target, float
endValue, float duration)
```

Parameters

`target` Outline

`endValue` float

The end value to reach

`duration` float

The duration of the tween

Returns

TweenerCore<Color, Color, ColorOptions>

DOFade(Text, float, float)

Tweens a Text's alpha color to the given value. Also stores the Text as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Color, Color, ColorOptions> DOFade(this Text target, float
endValue, float duration)
```

Parameters

`target` Text

`endValue` float

The end value to reach

`duration` float

The duration of the tween

Returns

TweenerCore<Color, Color, ColorOptions>

DOFillAmount(Image, float, float)

Tweens an Image's fillAmount to the given value. Also stores the image as the tween's target so it can be used for filtered operations

```
public static TweenerCore<float, float, FloatOptions> DOFillAmount(this Image target, float
endValue, float duration)
```

Parameters

`target` Image

`endValue` float

The end value to reach (0 to 1)

`duration` float

The duration of the tween

Returns

TweenerCore<float, float, FloatOptions>

DOFlexibleSize(LayoutElement, Vector2, float, bool)

Tweens an LayoutElement's flexibleWidth/Height to the given value. Also stores the LayoutElement as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Vector2, Vector2, VectorOptions> DOFlexibleSize(this LayoutElement
target, Vector2 endValue, float duration, bool snapping = false)
```

Parameters

`target` LayoutElement

`endValue` Vector2

The end value to reach

`duration` float

The duration of the tween

`snapping` bool

If TRUE the tween will smoothly snap all values to integers

Returns

TweenerCore<Vector2, Vector2, VectorOptions>

DOGradientColor(Image, Gradient, float)

Tweens an Image's colors using the given gradient (NOTE 1: only uses the colors of the gradient, not the alphas - NOTE 2: creates a Sequence, not a Tweener). Also stores the image as the tween's target so it can be used for filtered operations

```
public static Sequence DOGradientColor(this Image target, Gradient gradient, float duration)
```

Parameters

`target` Image

`gradient` Gradient

The gradient to use

`duration` float

The duration of the tween

Returns

Sequence

DOHorizontalNormalizedPos(ScrollRect, float, float, bool)

Tweens a ScrollRect's horizontalNormalizedPosition to the given value. Also stores the ScrollRect as the tween's target so it can be used for filtered operations

```
public static Tweener DOHorizontalNormalizedPos(this ScrollRect target, float endValue,
float duration, bool snapping = false)
```

Parameters

`target` ScrollRect

`endValue` float

The end value to reach

`duration` float

The duration of the tween

`snapping` bool

If TRUE the tween will smoothly snap all values to integers

Returns

Tweener

DOJumpAnchorPos(RectTransform, Vector2, float, int, float, bool)

Tweens a RectTransform's anchoredPosition to the given value, while also applying a jump effect along the Y axis. Returns a Sequence instead of a Tweener. Also stores the RectTransform as the tween's target so it can be used for filtered operations

```
public static Sequence DOJumpAnchorPos(this RectTransform target, Vector2 endValue, float
jumpPower, int numJumps, float duration, bool snapping = false)
```

Parameters

`target` RectTransform

`endValue` Vector2

The end value to reach

`jumpPower` float

Power of the jump (the max height of the jump is represented by this plus the final Y offset)

`numJumps` int

Total number of jumps

`duration` float

The duration of the tween

`snapping` bool

If TRUE the tween will smoothly snap all values to integers

Returns

Sequence

DOMinSize(LayoutElement, Vector2, float, bool)

Tweens an LayoutElement's minWidth/Height to the given value. Also stores the LayoutElement as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Vector2, Vector2, VectorOptions> DOMinSize(this LayoutElement
target, Vector2 endValue, float duration, bool snapping = false)
```

Parameters

`target` LayoutElement

`endValue` Vector2

The end value to reach

`duration` float

The duration of the tween

`snapping` bool

If TRUE the tween will smoothly snap all values to integers

Returns

TweenerCore<Vector2, Vector2, VectorOptions>

DONormalizedPos(ScrollRect, Vector2, float, bool)

Tweens a ScrollRect's horizontal/verticalNormalizedPosition to the given value. Also stores the ScrollRect as the tween's target so it can be used for filtered operations

```
public static Tweener DONormalizedPos(this ScrollRect target, Vector2 endValue, float
duration, bool snapping = false)
```

Parameters

`target` ScrollRect

`endValue` Vector2

The end value to reach

`duration` float

The duration of the tween

`snapping` bool

If TRUE the tween will smoothly snap all values to integers

Returns

Tweener

DOPivot(RectTransform, Vector2, float)

Tweens a RectTransform's pivot to the given value. Also stores the RectTransform as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Vector2, Vector2, VectorOptions> DOPivot(this RectTransform
target, Vector2 endValue, float duration)
```

Parameters

`target` RectTransform

`endValue` Vector2

The end value to reach

`duration` float

The duration of the tween

Returns

TweenerCore<Vector2, Vector2, VectorOptions>

DOPivotX(RectTransform, float, float)

Tweens a RectTransform's pivot X to the given value. Also stores the RectTransform as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Vector2, Vector2, VectorOptions> DOPivotX(this RectTransform
target, float endValue, float duration)
```

Parameters

`target` RectTransform

`endValue` float

The end value to reach

`duration` float

The duration of the tween

Returns

TweenerCore<Vector2, Vector2, VectorOptions>

DOPivotY(RectTransform, float, float)

Tweens a RectTransform's pivot Y to the given value. Also stores the RectTransform as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Vector2, Vector2, VectorOptions> DOPivotY(this RectTransform
target, float endValue, float duration)
```

Parameters

`target` RectTransform

`endValue` float

The end value to reach

`duration` float

The duration of the tween

Returns

TweenerCore<Vector2, Vector2, VectorOptions>

DOPreferredSize(LayoutElement, Vector2, float, bool)

Tweens an LayoutElement's preferredWidth/Height to the given value. Also stores the LayoutElement as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Vector2, Vector2, VectorOptions> DOPreferredSize(this
LayoutElement target, Vector2 endValue, float duration, bool snapping = false)
```

Parameters

`target` LayoutElement

`endValue` Vector2

The end value to reach

`duration` float

The duration of the tween

`snapping` bool

If TRUE the tween will smoothly snap all values to integers

Returns

TweenerCore<Vector2, Vector2, VectorOptions>

DOPunchAnchorPos(RectTransform, Vector2, float, int, float, bool)

Punches a RectTransform's anchoredPosition towards the given direction and then back to the starting one as if it was connected to the starting position via an elastic. Also stores the RectTransform as the tween's target so it can be used for filtered operations

```
public static Tweener DOPunchAnchorPos(this RectTransform target, Vector2 punch, float
duration, int vibrato = 10, float elasticity = 1, bool snapping = false)
```

Parameters

`target` RectTransform

`punch` Vector2

The direction and strength of the punch (added to the RectTransform's current position)

`duration` float

The duration of the tween

`vibrato` int

Indicates how much will the punch vibrate

`elasticity` float

Represents how much (0 to 1) the vector will go beyond the starting position when bouncing backwards. 1 creates a full oscillation between the punch direction and the opposite direction, while 0 oscillates only between the punch and the start position

`snapping` bool

If TRUE the tween will smoothly snap all values to integers

Returns

Tweener

DOScale(Outline, Vector2, float)

Tweens a Outline's effectDistance to the given value. Also stores the Outline as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Vector2, Vector2, VectorOptions> DOScale(this Outline target,
Vector2 endValue, float duration)
```

Parameters

`target` Outline

`endValue` Vector2

The end value to reach

`duration` float

The duration of the tween

Returns

TweenerCore<Vector2, Vector2, VectorOptions>

DOShakeAnchorPos(RectTransform, float, float, int, float, bool, bool, ShakeRandomnessMode)

Shakes a RectTransform's anchoredPosition with the given values. Also stores the RectTransform as the tween's target so it can be used for filtered operations

```
public static Tweener DOShakeAnchorPos(this RectTransform target, float duration, float
strength = 100, int vibrato = 10, float randomness = 90, bool snapping = false, bool fadeOut
= true, ShakeRandomnessMode randomnessMode = ShakeRandomnessMode.Full)
```

Parameters

`target` RectTransform

`duration` float

The duration of the tween

`strength` float

The shake strength

`vibrato` int

Indicates how much will the shake vibrate

`randomness` float

Indicates how much the shake will be random (0 to 180 - values higher than 90 kind of suck, so beware). Setting it to 0 will shake along a single direction.

`snapping` bool

If TRUE the tween will smoothly snap all values to integers

`fadeOut` bool

If TRUE the shake will automatically fadeOut smoothly within the tween's duration, otherwise it will not `randomnessMode` ShakeRandomnessMode

Randomness mode

Returns

Tweener

DOShakeAnchorPos(RectTransform, float, Vector2, int, float, bool, bool, ShakeRandomnessMode)

Shakes a RectTransform's anchoredPosition with the given values. Also stores the RectTransform as the tween's target so it can be used for filtered operations

```
public static Tweener DOShakeAnchorPos(this RectTransform target, float duration, Vector2
strength, int vibrato = 10, float randomness = 90, bool snapping = false, bool fadeOut =
true, ShakeRandomnessMode randomnessMode = ShakeRandomnessMode.Full)
```

Parameters

`target` RectTransform

`duration` float

The duration of the tween

`strength` Vector2

The shake strength on each axis

`vibrato` int

Indicates how much will the shake vibrate

`randomness` float

Indicates how much the shake will be random (0 to 180 - values higher than 90 kind of suck, so beware). Setting it to 0 will shake along a single direction.

`snapping` bool

If TRUE the tween will smoothly snap all values to integers

`fadeOut` bool

If TRUE the shake will automatically fadeOut smoothly within the tween's duration, otherwise it will not

`randomnessMode` ShakeRandomnessMode

Randomness mode

Returns

Tweener

DOShapeCircle(RectTransform, Vector2, float, float, bool, bool)

Tweens a RectTransform's anchoredPosition so that it draws a circle around the given center. Also stores the RectTransform as the tween's target so it can be used for filtered operations.

IMPORTANT: SetFrom(value) requires a UnityEngine.Vector2 instead of a float, where the X property represents the "from degrees value"

```
public static TweenerCore<Vector2, Vector2, CircleOptions> DOShapeCircle(this RectTransform
target, Vector2 center, float endValueDegrees, float duration, bool relativeCenter = false,
bool snapping = false)
```

Parameters

`target` RectTransform

`center` Vector2

Circle-center/pivot around which to rotate (in UI anchoredPosition coordinates)

`endValueDegrees` float

The end value degrees to reach (to rotate counter-clockwise pass a negative value)

`duration` float

The duration of the tween

`relativeCenter` bool

If TRUE the center coordinates will be considered as relative to the target's current anchoredPosition `snapping` bool

If TRUE the tween will smoothly snap all values to integers

Returns

TweenerCore<Vector2, Vector2, CircleOptions>

DOSizeDelta(RectTransform, Vector2, float, bool)

Tweens a RectTransform's sizeDelta to the given value. Also stores the RectTransform as the tween's target so it can be used for filtered operations

```
public static TweenerCore<Vector2, Vector2, VectorOptions> DOSizeDelta(this RectTransform
target, Vector2 endValue, float duration, bool snapping = false)
```

Parameters

`target` RectTransform

`endValue` Vector2

The end value to reach

`duration` float

The duration of the tween

`snapping` bool

If TRUE the tween will smoothly snap all values to integers

Returns

TweenerCore<Vector2, Vector2, VectorOptions>

DOText(Text, string, float, bool, ScrambleMode, string)

Tweens a Text's text to the given value. Also stores the Text as the tween's target so it can be used for filtered operations

```
public static TweenerCore<string, string, StringOptions> DOText(this Text target, string
endValue, float duration, bool richTextEnabled = true, ScrambleMode scrambleMode =
ScrambleMode.None, string scrambleChars = null)
```

Parameters

`target` Text

`endValue` string

The end string to tween to

`duration` float

The duration of the tween

`richTextEnabled` bool

If TRUE (default), rich text will be interpreted correctly while animated, otherwise all tags will be considered as normal text

`scrambleMode` ScrambleMode

The type of scramble mode to use, if any

`scrambleChars` string

A string containing the characters to use for scrambling. Use as many characters as possible (minimum 10) because DOTween uses a fast scramble mode which gives better results with more characters. Leave it to NULL (default) to use default ones

Returns

TweenerCore<string, string, StringOptions>

DOValue(Slider, float, float, bool)

Tweens a Slider's value to the given value. Also stores the Slider as the tween's target so it can be used for filtered operations

```
public static TweenerCore<float, float, FloatOptions> DOValue(this Slider target, float
endValue, float duration, bool snapping = false)
```

Parameters

`target` Slider

`endValue` float

The end value to reach

`duration` float

The duration of the tween

`snapping` bool

If TRUE the tween will smoothly snap all values to integers

Returns

TweenerCore<float, float, FloatOptions>

DOVerticalNormalizedPos(ScrollRect, float, float, bool)

Tweens a ScrollRect's verticalNormalizedPosition to the given value. Also stores the ScrollRect as the tween's target so it can be used for filtered operations

```
public static Tweener DOVerticalNormalizedPos(this ScrollRect target, float endValue, float
duration, bool snapping = false)
```

Parameters

`target` ScrollRect

`endValue` float

The end value to reach

`duration` float

The duration of the tween

`snapping` bool

If TRUE the tween will smoothly snap all values to integers

Returns

Tweener

Class DOTweenModuleUI.Utils

Namespace: DG.Tweening

Assembly: MasterMot.dll

```
public static class DOTweenModuleUI.Utils
```

Inheritance

object ← DOTweenModuleUI.Utils

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

SwitchToRectTransform(RectTransform, RectTransform)

Converts the anchoredPosition of the first RectTransform to the second RectTransform, taking into consideration offset, anchors and pivot, and returns the new anchoredPosition

```
public static Vector2 SwitchToRectTransform(RectTransform from, RectTransform to)
```

Parameters

`from` RectTransform

`to` RectTransform

Returns

Vector2

Class DOTweenModuleUnityVersion

Namespace: DG.Tweening

Assembly: MasterMot.dll

Shortcuts/functions that are not strictly related to specific Modules but are available only on some Unity versions

```
public static class DOTweenModuleUnityVersion
```

Inheritance

object ← DOTweenModuleUnityVersion

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

DOGradientColor(Material, Gradient, float)

Tweens a Material's color using the given gradient (NOTE 1: only uses the colors of the gradient, not the alphas - NOTE 2: creates a Sequence, not a Tweener). Also stores the image as the tween's target so it can be used for filtered operations

```
public static Sequence DOGradientColor(this Material target, Gradient gradient,
float duration)
```

Parameters

`target` Material

`gradient` Gradient

The gradient to use

`duration` float

The duration of the tween

Returns

Sequence

DOGradientColor(Material, Gradient, string, float)

Tweens a Material's named color property using the given gradient (NOTE 1: only uses the colors of the gradient, not the alphas - NOTE 2: creates a Sequence, not a Tweener). Also stores the image as the tween's target so it can be used for filtered operations

```
public static Sequence DOGradientColor(this Material target, Gradient gradient, string
property, float duration)
```

Parameters

`target` Material

`gradient` Gradient

The gradient to use

`property` string

The name of the material property to tween (like _Tint or _SpecColor)

`duration` float

The duration of the tween

Returns

Sequence

WaitForCompletion(Tween, bool)

Returns a UnityEngine.CustomYieldInstruction that waits until the tween is killed or complete. It can be used inside a coroutine as a yield.

Example usage:

```
yield return myTween.WaitForCompletion(true);
```

```
public static CustomYieldInstruction WaitForCompletion(this Tween t,
bool returnCustomYieldInstruction)
```

Parameters

`t` Tween

`returnCustomYieldInstruction` bool

Returns

CustomYieldInstruction

WaitForElapsedLoops(Tween, int, bool)

Returns a UnityEngine.CustomYieldInstruction that waits until the tween is killed or has gone through the given amount of loops. It can be used inside a coroutine as a yield.

Example usage:

```
yield return myTween.WaitForElapsedLoops(2);
```

```
public static CustomYieldInstruction WaitForElapsedLoops(this Tween t, int elapsedLoops,
bool returnCustomYieldInstruction)
```

Parameters

`t` Tween

`elapsedLoops` int

Elapsed loops to wait for

`returnCustomYieldInstruction` bool

Returns

CustomYieldInstruction

WaitForKill(Tween, bool)

Returns a UnityEngine.CustomYieldInstruction that waits until the tween is killed. It can be used inside a coroutine as a yield.

Example usage:

```
yield return myTween.WaitForKill();
```

```
public static CustomYieldInstruction WaitForKill(this Tween t,
bool returnCustomYieldInstruction)
```

Parameters

`t` Tween

`returnCustomYieldInstruction` bool

Returns

CustomYieldInstruction

WaitForPosition(Tween, float, bool)

Returns a UnityEngine.CustomYieldInstruction that waits until the tween is killed or has reached the given time position (loops included, delays excluded). It can be used inside a coroutine as a yield.

Example usage:

```
yield return myTween.WaitForPosition(2.5f);
```

```
public static CustomYieldInstruction WaitForPosition(this Tween t, float position,
bool returnCustomYieldInstruction)
```

Parameters

`t` Tween

`position` [float](#)

Position (loops included, delays excluded) to wait for

`returnCustomYieldInstruction` [bool](#)

Returns

CustomYieldInstruction

WaitForRewind(Tween, bool)

Returns a UnityEngine.CustomYieldInstruction that waits until the tween is killed or rewinded. It can be used inside a coroutine as a yield.

Example usage:

```
yield return myTween.WaitForRewind();
```

```
public static CustomYieldInstruction WaitForRewind(this Tween t,
bool returnCustomYieldInstruction)
```

Parameters

`t` Tween

`returnCustomYieldInstruction` [bool](#)

Returns

CustomYieldInstruction

WaitForStart(Tween, bool)

Returns a UnityEngine.CustomYieldInstruction that waits until the tween is killed or started (meaning when the tween is set in a playing state the first time, after any eventual delay). It can be used inside a coroutine as a yield.

Example usage:

```
yield return myTween.WaitForStart();
```

```
public static CustomYieldInstruction WaitForStart(this Tween t,
bool returnCustomYieldInstruction)
```

Parameters

`t` Tween

`returnCustomYieldInstruction` bool

Returns

CustomYieldInstruction

Class DOTweenModuleUtils

Namespace: DG.Tweening

Assembly: MasterMot.dll

Utility functions that deal with available Modules. Modules defines:

- DOTAUDIO
- DOTPHYSICS
- DOTPHYSICS2D
- DOTSPRITE
- DOTUI Extra defines set and used for implementation of external assets:
- DOTWEEN_TMP ► TextMesh Pro
- DOTWEEN_TK2D ► 2D Toolkit

```
public static class DOTweenModuleUtils
```

Inheritance

object ← DOTweenModuleUtils

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

Init()

Called via Reflection by DOTweenComponent on Awake

```
public static void Init()
```

Class DOTweenModuleUtils.Physics

Namespace: DG.Tweening

Assembly: MasterMot.dll

```
public static class DOTweenModuleUtils.Physics
```

Inheritance

object ← DOTweenModuleUtils.Physics

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

CreateDOTweenPathTween(MonoBehaviour, bool, bool, Path, float, PathMode)

```
public static TweenerCore<Vector3, Path, PathOptions> CreateDOTweenPathTween(MonoBehaviour
target, bool tweenRigidbody, bool isLocal, Path path, float duration, PathMode pathMode)
```

Parameters

`target` MonoBehaviour

`tweenRigidbody` bool

`isLocal` bool

`path` Path

`duration` float

`pathMode` PathMode

Returns

TweenerCore<Vector3, Path, PathOptions>

HasRigidbody(Component)

```
public static bool HasRigidbody(Component target)
```

Parameters

`target` Component

Returns

bool

HasRigidbody2D(Component)

```
public static bool HasRigidbody2D(Component target)
```

Parameters

`target` Component

Returns

bool

SetOrientationOnPath(PathOptions, Tween, Quaternion, Transform)

```
public static void SetOrientationOnPath(PathOptions options, Tween t, Quaternion newRot,
Transform trans)
```

Parameters

`options` PathOptions

`t` Tween

`newRot` Quaternion

`trans` Transform

Namespace MasterMot

Classes

APIManager

AudioController

GameManager

InputController

PlayFabController

UIController

WordGameController

WordGameRenderer

Class APIManager

Namespace: MasterMot

Assembly: MasterMot.dll

```
public class APIManager : MonoBehaviour
```

Inheritance

object ← Object ← Component ← Behaviour ← MonoBehaviour ← APIManager

Inherited Members

MonoBehaviour.IsInvoking() , MonoBehaviour.CancelInvoke() , MonoBehaviour.Invoke(string, float) , MonoBehaviour.InvokeRepeating(string, float, float) , MonoBehaviour.CancelInvoke(string) , MonoBehaviour.IsInvoking(string) , MonoBehaviour.StartCoroutine(string) , MonoBehaviour.StartCoroutine(string, object) , MonoBehaviour.StartCoroutine(IEnumerator) , MonoBehaviour.StartCoroutine_Auto(IEnumerator) , MonoBehaviour.StopCoroutine(IEnumerator) , MonoBehaviour.StopCoroutine(Coroutine) , MonoBehaviour.StopCoroutine(string) , MonoBehaviour.StopAllCoroutines() , MonoBehaviour.print(object) , MonoBehaviour.destroyCancellationToken , MonoBehaviour.useGUILayout , MonoBehaviour.runInEditMode , Behaviour.enabled , Behaviour.isActiveAndEnabled , Component.GetComponent(Type) , Component.GetComponent<T>() , Component.TryGetComponent(Type, out Component) , Component.TryGetComponent<T>(out T) , Component.GetComponent(string) , Component.GetComponentInChildren(Type, bool) , Component.GetComponentInChildren(Type) , Component.GetComponentInChildren<T>(bool) , Component.GetComponentInChildren<T>() , Component.GetComponentsInChildren(Type, bool) , Component.GetComponentsInChildren(Type) , Component.GetComponentsInChildren<T>(bool) , Component.GetComponentsInChildren<T>(bool, List<T>) , Component.GetComponentsInChildren<T>() , Component.GetComponentsInChildren<T>(List<T>) , Component.GetComponentInParent(Type, bool) , Component.GetComponentInParent(Type) , Component.GetComponentInParent<T>(bool) , Component.GetComponentInParent<T>() , Component.GetComponentsInParent(Type, bool) , Component.GetComponentsInParent(Type) , Component.GetComponentsInParent<T>(bool) , Component.GetComponentsInParent<T>(bool, List<T>) , Component.GetComponentsInParent<T>() , Component.GetComponents(Type) , Component.GetComponents(Type, List<Component>) , Component.GetComponents<T>(List<T>) , Component.GetComponents<T>() , Component.GetComponentIndex() , Component.CompareTag(string) , Component.SendMessageUpwards(string, object, SendMessageOptions) , Component.SendMessageUpwards(string, object) , Component.SendMessageUpwards(string) , Component.SendMessageUpwards(string, SendMessageOptions) ,

Component.SendMessage(string, object) , Component.SendMessage(string) ,
Component.SendMessage(string, object, SendMessageOptions) ,
Component.SendMessage(string, SendMessageOptions) ,
Component.BroadcastMessage(string, object, SendMessageOptions) ,
Component.BroadcastMessage(string, object) , Component.BroadcastMessage(string) ,
Component.BroadcastMessage(string, SendMessageOptions) , Component.transform ,
Component.gameObject , Component.tag , Object.GetInstanceID() , Object.GetHashCode() ,
Object.Equals(object) , Object.InstantiateAsync<T>(T) , Object.InstantiateAsync<T>(T, Transform) ,
Object.InstantiateAsync<T>(T, Vector3, Quaternion) ,
Object.InstantiateAsync<T>(T, Transform, Vector3, Quaternion) , Object.InstantiateAsync<T>(T, int) ,
Object.InstantiateAsync<T>(T, int, Transform) ,
Object.InstantiateAsync<T>(T, int, Vector3, Quaternion) ,
Object.InstantiateAsync<T>(T, int, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>) ,
Object.InstantiateAsync<T>(T, int, Transform, Vector3, Quaternion) ,
Object.InstantiateAsync<T>(T, int, Transform, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>) ,
Object.Instantiate(Object, Vector3, Quaternion) ,
Object.Instantiate(Object, Vector3, Quaternion, Transform) , Object.Instantiate(Object) ,
Object.Instantiate(Object, Scene) , Object.Instantiate(Object, Transform) ,
Object.Instantiate(Object, Transform, bool) , Object.Instantiate<T>(T) ,
Object.Instantiate<T>(T, Vector3, Quaternion) ,
Object.Instantiate<T>(T, Vector3, Quaternion, Transform) , Object.Instantiate<T>(T, Transform) ,
Object.Instantiate<T>(T, Transform, bool) , Object.Destroy(Object, float) , Object.Destroy(Object) ,
Object.DestroyImmediate(Object, bool) , Object.DestroyImmediate(Object) ,
Object.FindObjectsOfType(Type) , Object.FindObjectsOfType(Type, bool) ,
Object.FindObjectsByType(Type, FindObjectsSortMode) ,
Object.FindObjectsByType(Type, FindObjectsInactive, FindObjectsSortMode) ,
Object.DontDestroyOnLoad(Object) , Object.DestroyObject(Object, float) ,
Object.DestroyObject(Object) , Object.FindSceneObjectsOfType(Type) ,
Object.FindObjectsOfTypeIncludingAssets(Type) , Object.FindObjectsOfType<T>() ,
Object.FindObjectsByType<T>(FindObjectsSortMode) , Object.FindObjectsOfType<T>(bool) ,
Object.FindObjectsByType<T>(FindObjectsInactive, FindObjectsSortMode) ,
Object.FindObjectOfType<T>() , Object.FindObjectOfType<T>(bool) ,
Object.FindFirstObjectByType<T>() , Object.FindAnyObjectByType<T>() ,
Object.FindFirstObjectByType<T>(FindObjectsInactive) ,
Object.FindAnyObjectByType<T>(FindObjectsInactive) , Object.FindObjectsOfTypeAll(Type) ,
Object.FindObjectOfType(Type) , Object.FindFirstObjectByType(Type) ,
Object.FindAnyObjectByType(Type) , Object.FindObjectOfType(Type, bool) ,
Object.FindFirstObjectByType(Type, FindObjectsInactive) ,
Object.FindAnyObjectByType(Type, FindObjectsInactive) , Object.ToString() , Object.name ,

Object.hideFlags , object.Equals(object, object) , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Methods

GetEasyWord()

Cette méthode fait appel à l'API Trouvemot afin d'avoir un mot aléatoire de difficulté facile, c'est à dire avec 5 lettres ou moins.

```
public static IEnumerator GetEasyWord()
```

Returns

IEnumerator

GetHardWord()

Cette méthode fait appel à l'API Trouvemot afin d'avoir un mot aléatoire de difficulté difficile, d'au moins 7 lettres de longueur.

```
public static IEnumerator GetHardWord()
```

Returns

IEnumerator

GetMediumWord()

Cette méthode fait appel à l'API Trouvemot afin d'avoir un mot aléatoire de difficulté moyenne, avec 6 lettres.

```
public static IEnumerator GetMediumWord()
```

Returns

IEnumerator

Class AudioController

Namespace: MasterMot

Assembly: MasterMot.dll

```
public class AudioController : MonoBehaviour
```

Inheritance

object ← Object ← Component ← Behaviour ← MonoBehaviour ← AudioController

Inherited Members

MonoBehaviour.IsInvoking() , MonoBehaviour.CancelInvoke() , MonoBehaviour.Invoke(string, float) , MonoBehaviour.InvokeRepeating(string, float, float) , MonoBehaviour.CancelInvoke(string) , MonoBehaviour.IsInvoking(string) , MonoBehaviour.StartCoroutine(string) , MonoBehaviour.StartCoroutine(string, object) , MonoBehaviour.StartCoroutine(IEnumerator) , MonoBehaviour.StartCoroutine_Auto(IEnumerator) , MonoBehaviour.StopCoroutine(IEnumerator) , MonoBehaviour.StopCoroutine(Coroutine) , MonoBehaviour.StopCoroutine(string) , MonoBehaviour.StopAllCoroutines() , MonoBehaviour.print(object) , MonoBehaviour.destroyCancellationToken , MonoBehaviour.useGUILayout , MonoBehaviour.runInEditMode , Behaviour.enabled , Behaviour.isActiveAndEnabled , Component.GetComponent(Type) , Component.GetComponent<T>() , Component.TryGetComponent(Type, out Component) , Component.TryGetComponent<T>(out T) , Component.GetComponent(string) , Component.GetComponentInChildren(Type, bool) , Component.GetComponentInChildren(Type) , Component.GetComponentInChildren<T>(bool) , Component.GetComponentInChildren<T>() , Component.GetComponentsInChildren(Type, bool) , Component.GetComponentsInChildren(Type) , Component.GetComponentsInChildren<T>(bool) , Component.GetComponentsInChildren<T>(bool, List<T>) , Component.GetComponentsInChildren<T>() , Component.GetComponentsInChildren<T>(List<T>) , Component.GetComponentInParent(Type, bool) , Component.GetComponentInParent(Type) , Component.GetComponentInParent<T>(bool) , Component.GetComponentInParent<T>() , Component.GetComponentsInParent(Type, bool) , Component.GetComponentsInParent(Type) , Component.GetComponentsInParent<T>(bool) , Component.GetComponentsInParent<T>(bool, List<T>) , Component.GetComponentsInParent<T>() , Component.GetComponents(Type) , Component.GetComponents(Type, List<Component>) , Component.GetComponents<T>(List<T>) , Component.GetComponents<T>() , Component.GetComponentIndex() , Component.CompareTag(string) , Component.SendMessageUpwards(string, object, SendMessageOptions) , Component.SendMessageUpwards(string, object) , Component.SendMessageUpwards(string) , Component.SendMessageUpwards(string, SendMessageOptions) ,

Component.SendMessage(string, object) , Component.SendMessage(string) ,
Component.SendMessage(string, object, SendMessageOptions) ,
Component.SendMessage(string, SendMessageOptions) ,
Component.BroadcastMessage(string, object, SendMessageOptions) ,
Component.BroadcastMessage(string, object) , Component.BroadcastMessage(string) ,
Component.BroadcastMessage(string, SendMessageOptions) , Component.transform ,
Component.gameObject , Component.tag , Object.GetInstanceID() , Object.GetHashCode() ,
Object.Equals(object) , Object.InstantiateAsync<T>(T) , Object.InstantiateAsync<T>(T, Transform) ,
Object.InstantiateAsync<T>(T, Vector3, Quaternion) ,
Object.InstantiateAsync<T>(T, Transform, Vector3, Quaternion) , Object.InstantiateAsync<T>(T, int) ,
Object.InstantiateAsync<T>(T, int, Transform) ,
Object.InstantiateAsync<T>(T, int, Vector3, Quaternion) ,
Object.InstantiateAsync<T>(T, int, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>) ,
Object.InstantiateAsync<T>(T, int, Transform, Vector3, Quaternion) ,
Object.InstantiateAsync<T>(T, int, Transform, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>) ,
Object.Instantiate(Object, Vector3, Quaternion) ,
Object.Instantiate(Object, Vector3, Quaternion, Transform) , Object.Instantiate(Object) ,
Object.Instantiate(Object, Scene) , Object.Instantiate(Object, Transform) ,
Object.Instantiate(Object, Transform, bool) , Object.Instantiate<T>(T) ,
Object.Instantiate<T>(T, Vector3, Quaternion) ,
Object.Instantiate<T>(T, Vector3, Quaternion, Transform) , Object.Instantiate<T>(T, Transform) ,
Object.Instantiate<T>(T, Transform, bool) , Object.Destroy(Object, float) , Object.Destroy(Object) ,
Object.DestroyImmediate(Object, bool) , Object.DestroyImmediate(Object) ,
Object.FindObjectsOfType(Type) , Object.FindObjectsOfType(Type, bool) ,
Object.FindObjectsByType(Type, FindObjectsSortMode) ,
Object.FindObjectsByType(Type, FindObjectsInactive, FindObjectsSortMode) ,
Object.DontDestroyOnLoad(Object) , Object.DestroyObject(Object, float) ,
Object.DestroyObject(Object) , Object.FindSceneObjectsOfType(Type) ,
Object.FindObjectsOfTypeIncludingAssets(Type) , Object.FindObjectsOfType<T>() ,
Object.FindObjectsByType<T>(FindObjectsSortMode) , Object.FindObjectsOfType<T>(bool) ,
Object.FindObjectsByType<T>(FindObjectsInactive, FindObjectsSortMode) ,
Object.FindObjectOfType<T>() , Object.FindObjectOfType<T>(bool) ,
Object.FindFirstObjectByType<T>() , Object.FindAnyObjectByType<T>() ,
Object.FindFirstObjectByType<T>(FindObjectsInactive) ,
Object.FindAnyObjectByType<T>(FindObjectsInactive) , Object.FindObjectsOfTypeAll(Type) ,
Object.FindObjectOfType(Type) , Object.FindFirstObjectByType(Type) ,
Object.FindAnyObjectByType(Type) , Object.FindObjectOfType(Type, bool) ,
Object.FindFirstObjectByType(Type, FindObjectsInactive) ,
Object.FindAnyObjectByType(Type, FindObjectsInactive) , Object.ToString() , Object.name ,

Object.hideFlags , object.Equals(object, object) , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Methods

GetThis()

Cette fonction statique retourne l'instance d'AudioController active. Vu qu'il s'agit d'un Singleton, il ne peut y en avoir qu'un seul.

```
public static AudioController GetThis()
```

Returns

AudioController

L'instance d'AudioController existante.

MusicValueIsChanged(int)

Cette méthode permet de changer le volume de la musique du jeu. Cette valeur est entre 0 et 100, et indiquée par le SliderInt de musique dans les options de jeu.

```
public void MusicValueIsChanged(int newValue)
```

Parameters

`newValue` int

La nouvelle valeur de volume de musique.

PlayBadLetter()

Cette méthode déclenche l'effet sonore d'une lettre qui n'est pas dans le mot.

```
public void PlayBadLetter()
```

PlayButtonPress()

Cette méthode déclenche l'effet sonore d'un bouton qui est appuyé.

```
public void PlayButtonPress()
```

PlayGameMusic()

Cette méthode permet de jouer la musique du jeu.

```
public void PlayGameMusic()
```

PlayGoodLetter()

Cette méthode déclenche l'effet sonore d'une bonne lettre qui est découverte.

```
public void PlayGoodLetter()
```

PlayWrongPlace()

Cette méthode déclenche l'effet sonore d'une lettre qui est au mauvais endroit.

```
public void PlayWrongPlace()
```

SFXValueIsChanged(int)

Cette méthode permet de changer le volume des effets sonores du jeu. Cette valeur est entre 0 et 100, et indiquée par le SliderInt d'effets sonores dans les options de jeu.

```
public void SFXValueIsChanged(int newValue)
```

Parameters

`newValue` int

La nouvelle valeur de volume d'effets sonores.

StopGameMusic()

Cette méthode arrête la musique du jeu.

```
public void StopGameMusic()
```

Class GameManager

Namespace: MasterMot

Assembly: MasterMot.dll

```
public class GameManager : MonoBehaviour
```

Inheritance

object ← Object ← Component ← Behaviour ← MonoBehaviour ← GameManager

Inherited Members

MonoBehaviour.IsInvoking() , MonoBehaviour.CancelInvoke() , MonoBehaviour.Invoke(string, float) , MonoBehaviour.InvokeRepeating(string, float, float) , MonoBehaviour.CancelInvoke(string) , MonoBehaviour.IsInvoking(string) , MonoBehaviour.StartCoroutine(string) , MonoBehaviour.StartCoroutine(string, object) , MonoBehaviour.StartCoroutine(IEnumerator) , MonoBehaviour.StartCoroutine_Auto(IEnumerator) , MonoBehaviour.StopCoroutine(IEnumerator) , MonoBehaviour.StopCoroutine(Coroutine) , MonoBehaviour.StopCoroutine(string) , MonoBehaviour.StopAllCoroutines() , MonoBehaviour.print(object) , MonoBehaviour.destroyCancellationToken , MonoBehaviour.useGUILayout , MonoBehaviour.runInEditMode , Behaviour.enabled , Behaviour.isActiveAndEnabled , Component.GetComponent(Type) , Component.GetComponent<T>() , Component.TryGetComponent(Type, out Component) , Component.TryGetComponent<T>(out T) , Component.GetComponent(string) , Component.GetComponentInChildren(Type, bool) , Component.GetComponentInChildren(Type) , Component.GetComponentInChildren<T>(bool) , Component.GetComponentInChildren<T>() , Component.GetComponentsInChildren(Type, bool) , Component.GetComponentsInChildren(Type) , Component.GetComponentsInChildren<T>(bool) , Component.GetComponentsInChildren<T>(bool, List<T>) , Component.GetComponentsInChildren<T>() , Component.GetComponentsInChildren<T>(List<T>) , Component.GetComponentInParent(Type, bool) , Component.GetComponentInParent(Type) , Component.GetComponentInParent<T>(bool) , Component.GetComponentInParent<T>() , Component.GetComponentsInParent(Type, bool) , Component.GetComponentsInParent(Type) , Component.GetComponentsInParent<T>(bool) , Component.GetComponentsInParent<T>(bool, List<T>) , Component.GetComponentsInParent<T>() , Component.GetComponents(Type) , Component.GetComponents(Type, List<Component>) , Component.GetComponents<T>(List<T>) , Component.GetComponents<T>() , Component.GetComponentIndex() , Component.CompareTag(string) , Component.SendMessageUpwards(string, object, SendMessageOptions) , Component.SendMessageUpwards(string, object) , Component.SendMessageUpwards(string) , Component.SendMessageUpwards(string, SendMessageOptions) ,

Component.SendMessage(string, object) , Component.SendMessage(string) , Component.SendMessage(string, object, SendMessageOptions) , Component.SendMessage(string, SendMessageOptions) , Component.BroadcastMessage(string, object, SendMessageOptions) , Component.BroadcastMessage(string, object) , Component.BroadcastMessage(string) , Component.BroadcastMessage(string, SendMessageOptions) , Component.transform , Component.gameObject , Component.tag , Object.GetInstanceID() , Object.GetHashCode() , Object.Equals(object) , Object.InstantiateAsync<T>(T) , Object.InstantiateAsync<T>(T, Transform) , Object.InstantiateAsync<T>(T, Vector3, Quaternion) , Object.InstantiateAsync<T>(T, Transform, Vector3, Quaternion) , Object.InstantiateAsync<T>(T, int) , Object.InstantiateAsync<T>(T, int, Transform) , Object.InstantiateAsync<T>(T, int, Vector3, Quaternion) , Object.InstantiateAsync<T>(T, int, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>) , Object.InstantiateAsync<T>(T, int, Transform, Vector3, Quaternion) , Object.InstantiateAsync<T>(T, int, Transform, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>) , Object.Instantiate(Object, Vector3, Quaternion) , Object.Instantiate(Object, Vector3, Quaternion, Transform) , Object.Instantiate(Object) , Object.Instantiate(Object, Scene) , Object.Instantiate(Object, Transform) , Object.Instantiate(Object, Transform, bool) , Object.Instantiate<T>(T) , Object.Instantiate<T>(T, Vector3, Quaternion) , Object.Instantiate<T>(T, Vector3, Quaternion, Transform) , Object.Instantiate<T>(T, Transform) , Object.Instantiate<T>(T, Transform, bool) , Object.Destroy(Object, float) , Object.Destroy(Object) , Object.DestroyImmediate(Object, bool) , Object.DestroyImmediate(Object) , Object.FindObjectsOfType(Type) , Object.FindObjectsOfType(Type, bool) , Object.FindObjectsByType(Type, FindObjectsSortMode) , Object.FindObjectsByType(Type, FindObjectsInactive, FindObjectsSortMode) , Object.DontDestroyOnLoad(Object) , Object.DestroyObject(Object, float) , Object.DestroyObject(Object) , Object.FindSceneObjectsOfType(Type) , Object.FindObjectsOfTypeIncludingAssets(Type) , Object.FindObjectsOfType<T>() , Object.FindObjectsByType<T>(FindObjectsSortMode) , Object.FindObjectsOfType<T>(bool) , Object.FindObjectsByType<T>(FindObjectsInactive, FindObjectsSortMode) , Object.FindObjectOfType<T>() , Object.FindObjectOfType<T>(bool) , Object.FindFirstObjectByType<T>() , Object.FindAnyObjectByType<T>() , Object.FindFirstObjectByType<T>(FindObjectsInactive) , Object.FindAnyObjectByType<T>(FindObjectsInactive) , Object.FindObjectsOfTypeAll(Type) , Object.FindObjectOfType(Type) , Object.FindFirstObjectByType(Type) , Object.FindAnyObjectByType(Type) , Object.FindObjectOfType(Type, bool) , Object.FindFirstObjectByType(Type, FindObjectsInactive) , Object.FindAnyObjectByType(Type, FindObjectsInactive) , Object.ToString() , Object.name ,

Object.hideFlags , object.Equals(object, object) , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Methods

DailyGame()

Cette méthode permet de commencer une nouvelle partie en changeant la scène vers la scène de jeu. Dans le cas de cette méthode, il s'agit de la partie avec le mot du jour.

```
public void DailyGame()
```

GetThis()

Retourne l'instance de ce Singleton.

```
public static GameManager GetThis()
```

Returns

GameManager

L'instance de ce singleton

GoToLeaderboards()

Cette méthode permet d'accéder à l'écran de classements des joueurs globaux pour le mot du jour, en changeant de scène.

```
public void GoToLeaderboards()
```

GoToProfile()

Cette méthode permet d'accéder à l'écran de profil du joueur, en changeant de scène.

```
public void GoToProfile()
```

NewGame()

Cette méthode permet de commencer une nouvelle partie en changeant la scène vers la scène de jeu.

```
public void NewGame()
```

RestartGame()

Cette méthode permet de commencer une nouvelle partie en restant dans la scène de jeu.

```
public void RestartGame()
```

ReturnToMenu()

Cette méthode permet de retourner dans le menu principal en changeant de scène.

```
public void ReturnToMenu()
```

Class InputController

Namespace: MasterMot

Assembly: MasterMot.dll

```
public class InputController : MonoBehaviour
```

Inheritance

object ← Object ← Component ← Behaviour ← MonoBehaviour ← InputController

Inherited Members

MonoBehaviour.IsInvoking() , MonoBehaviour.CancelInvoke() , MonoBehaviour.Invoke(string, float) , MonoBehaviour.InvokeRepeating(string, float, float) , MonoBehaviour.CancelInvoke(string) , MonoBehaviour.IsInvoking(string) , MonoBehaviour.StartCoroutine(string) , MonoBehaviour.StartCoroutine(string, object) , MonoBehaviour.StartCoroutine(IEnumerator) , MonoBehaviour.StartCoroutine_Auto(IEnumerator) , MonoBehaviour.StopCoroutine(IEnumerator) , MonoBehaviour.StopCoroutine(Coroutine) , MonoBehaviour.StopCoroutine(string) , MonoBehaviour.StopAllCoroutines() , MonoBehaviour.print(object) , MonoBehaviour.destroyCancellationToken , MonoBehaviour.useGUILayout , MonoBehaviour.runInEditMode , Behaviour.enabled , Behaviour.isActiveAndEnabled , Component.GetComponent(Type) , Component.GetComponent<T>() , Component.TryGetComponent(Type, out Component) , Component.TryGetComponent<T>(out T) , Component.GetComponent(string) , Component.GetComponentInChildren(Type, bool) , Component.GetComponentInChildren(Type) , Component.GetComponentInChildren<T>(bool) , Component.GetComponentInChildren<T>() , Component.GetComponentsInChildren(Type, bool) , Component.GetComponentsInChildren(Type) , Component.GetComponentsInChildren<T>(bool) , Component.GetComponentsInChildren<T>(bool, List<T>) , Component.GetComponentsInChildren<T>() , Component.GetComponentsInChildren<T>(List<T>) , Component.GetComponentInParent(Type, bool) , Component.GetComponentInParent(Type) , Component.GetComponentInParent<T>(bool) , Component.GetComponentInParent<T>() , Component.GetComponentsInParent(Type, bool) , Component.GetComponentsInParent(Type) , Component.GetComponentsInParent<T>(bool) , Component.GetComponentsInParent<T>(bool, List<T>) , Component.GetComponentsInParent<T>() , Component.GetComponents(Type) , Component.GetComponents(Type, List<Component>) , Component.GetComponents<T>(List<T>) , Component.GetComponents<T>() , Component.GetComponentIndex() , Component.CompareTag(string) , Component.SendMessageUpwards(string, object, SendMessageOptions) , Component.SendMessageUpwards(string, object) , Component.SendMessageUpwards(string) , Component.SendMessageUpwards(string, SendMessageOptions) ,

Component.SendMessage(string, object) , Component.SendMessage(string) , Component.SendMessage(string, object, SendMessageOptions) , Component.SendMessage(string, SendMessageOptions) , Component.BroadcastMessage(string, object, SendMessageOptions) , Component.BroadcastMessage(string, object) , Component.BroadcastMessage(string) , Component.BroadcastMessage(string, SendMessageOptions) , Component.transform , Component.gameObject , Component.tag , Object.GetInstanceID() , Object.GetHashCode() , Object.Equals(object) , Object.InstantiateAsync<T>(T) , Object.InstantiateAsync<T>(T, Transform) , Object.InstantiateAsync<T>(T, Vector3, Quaternion) , Object.InstantiateAsync<T>(T, Transform, Vector3, Quaternion) , Object.InstantiateAsync<T>(T, int) , Object.InstantiateAsync<T>(T, int, Transform) , Object.InstantiateAsync<T>(T, int, Vector3, Quaternion) , Object.InstantiateAsync<T>(T, int, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>) , Object.InstantiateAsync<T>(T, int, Transform, Vector3, Quaternion) , Object.InstantiateAsync<T>(T, int, Transform, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>) , Object.Instantiate(Object, Vector3, Quaternion) , Object.Instantiate(Object, Vector3, Quaternion, Transform) , Object.Instantiate(Object) , Object.Instantiate(Object, Scene) , Object.Instantiate(Object, Transform) , Object.Instantiate(Object, Transform, bool) , Object.Instantiate<T>(T) , Object.Instantiate<T>(T, Vector3, Quaternion) , Object.Instantiate<T>(T, Vector3, Quaternion, Transform) , Object.Instantiate<T>(T, Transform) , Object.Instantiate<T>(T, Transform, bool) , Object.Destroy(Object, float) , Object.Destroy(Object) , Object.DestroyImmediate(Object, bool) , Object.DestroyImmediate(Object) , Object.FindObjectsOfType(Type) , Object.FindObjectsOfType(Type, bool) , Object.FindObjectsByType(Type, FindObjectsSortMode) , Object.FindObjectsByType(Type, FindObjectsInactive, FindObjectsSortMode) , Object.DontDestroyOnLoad(Object) , Object.DestroyObject(Object, float) , Object.DestroyObject(Object) , Object.FindSceneObjectsOfType(Type) , Object.FindObjectsOfTypeIncludingAssets(Type) , Object.FindObjectsOfType<T>() , Object.FindObjectsByType<T>(FindObjectsSortMode) , Object.FindObjectsOfType<T>(bool) , Object.FindObjectsByType<T>(FindObjectsInactive, FindObjectsSortMode) , Object.FindObjectOfType<T>() , Object.FindObjectOfType<T>(bool) , Object.FindFirstObjectByType<T>() , Object.FindAnyObjectByType<T>() , Object.FindFirstObjectByType<T>(FindObjectsInactive) , Object.FindAnyObjectByType<T>(FindObjectsInactive) , Object.FindObjectsOfTypeAll(Type) , Object.FindObjectOfType(Type) , Object.FindFirstObjectByType(Type) , Object.FindAnyObjectByType(Type) , Object.FindObjectOfType(Type, bool) , Object.FindFirstObjectByType(Type, FindObjectsInactive) , Object.FindAnyObjectByType(Type, FindObjectsInactive) , Object.ToString() , Object.name ,

Object.hideFlags , object.Equals(object, object) , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Methods

AssignDifficultyButtons()

Cette fonction attribue les callbacks aux boutons de l'écran de sélection de difficulté, avec un court délai initial pour s'assurer que l'écran soit bien chargé.

```
public IEnumerator AssignDifficultyButtons()
```

Returns

IEnumerator

Un énumérateur pour l'attente d'exécution.

AssignKeyboard()

Cette fonction attribue les callbacks aux boutons de clavier de l'écran de jeu, avec un court délai initial pour s'assurer que l'écran soit bien chargé.

```
public IEnumerator AssignKeyboard()
```

Returns

IEnumerator

Un énumérateur pour l'attente d'exécution.

AssignLeaderboardButtons()

Cette fonction attribue les callbacks aux boutons de l'écran de classement des joueurs, avec un court délai initial pour s'assurer que l'écran soit bien chargé.

```
public IEnumerator AssignLeaderboardButtons()
```

Returns

IEnumerator

Un énumérateur pour l'attente d'exécution.

AssignMenuButtons()

Cette fonction attribue les callbacks aux boutons du menu principal, avec un court délai initial pour s'assurer que l'écran soit bien chargé.

```
public IEnumerator AssignMenuButtons()
```

Returns

IEnumerator

Un énumérateur pour l'attente d'exécution.

AssignProfileButtons()

Cette fonction attribue les callbacks aux boutons de l'écran de profil du joueur, avec un court délai initial pour s'assurer que l'écran soit bien chargé.

```
public IEnumerator AssignProfileButtons()
```

Returns

IEnumerator

Un énumérateur pour l'attente d'exécution.

AssignSettingsButtons()

Cette fonction attribue les callbacks aux boutons de l'écran d'options, avec un court délai initial pour s'assurer que l'écran soit bien chargé.

```
public IEnumerator AssignSettingsButtons()
```

Returns

[IEnumerator](#)

Un énumérateur pour l'attente d'exécution.

AssignUsernameButtons()

Cette fonction attribue les callbacks au bouton de de confirmation de l'écran d'initialisation du pseudo joueur, avec un court délai initial pour s'assurer que l'écran soit bien chargé.

```
public IEnumerator AssignUsernameButtons()
```

Returns

[IEnumerator](#)

Un énumérateur pour l'attente d'exécution.

GetThis()

Cette fonction statique retourne l'instance d'InputController active. Vu qu'il s'agit d'un Singleton, il ne peut y en avoir qu'un seul.

```
public static InputController GetThis()
```

Returns

[InputController](#)

Retourne l'instance de InputController existante.

Class PlayFabController

Namespace: MasterMot

Assembly: MasterMot.dll

```
public class PlayFabController : MonoBehaviour
```

Inheritance

object ← Object ← Component ← Behaviour ← MonoBehaviour ← PlayFabController

Inherited Members

MonoBehaviour.IsInvoking() , MonoBehaviour.CancelInvoke() , MonoBehaviour.Invoke(string, float) , MonoBehaviour.InvokeRepeating(string, float, float) , MonoBehaviour.CancelInvoke(string) , MonoBehaviour.IsInvoking(string) , MonoBehaviour.StartCoroutine(string) , MonoBehaviour.StartCoroutine(string, object) , MonoBehaviour.StartCoroutine(IEnumerator) , MonoBehaviour.StartCoroutine_Auto(IEnumerator) , MonoBehaviour.StopCoroutine(IEnumerator) , MonoBehaviour.StopCoroutine(Coroutine) , MonoBehaviour.StopCoroutine(string) , MonoBehaviour.StopAllCoroutines() , MonoBehaviour.print(object) , MonoBehaviour.destroyCancellationToken , MonoBehaviour.useGUILayout , MonoBehaviour.runInEditMode , Behaviour.enabled , Behaviour.isActiveAndEnabled , Component.GetComponent(Type) , Component.GetComponent<T>() , Component.TryGetComponent(Type, out Component) , Component.TryGetComponent<T>(out T) , Component.GetComponent(string) , Component.GetComponentInChildren(Type, bool) , Component.GetComponentInChildren(Type) , Component.GetComponentInChildren<T>(bool) , Component.GetComponentInChildren<T>() , Component.GetComponentsInChildren(Type, bool) , Component.GetComponentsInChildren(Type) , Component.GetComponentsInChildren<T>(bool) , Component.GetComponentsInChildren<T>(bool, List<T>) , Component.GetComponentsInChildren<T>() , Component.GetComponentsInChildren<T>(List<T>) , Component.GetComponentInParent(Type, bool) , Component.GetComponentInParent(Type) , Component.GetComponentInParent<T>(bool) , Component.GetComponentInParent<T>() , Component.GetComponentsInParent(Type, bool) , Component.GetComponentsInParent(Type) , Component.GetComponentsInParent<T>(bool) , Component.GetComponentsInParent<T>(bool, List<T>) , Component.GetComponentsInParent<T>() , Component.GetComponents(Type) , Component.GetComponents(Type, List<Component>) , Component.GetComponents<T>(List<T>) , Component.GetComponents<T>() , Component.GetComponentIndex() , Component.CompareTag(string) , Component.SendMessageUpwards(string, object, SendMessageOptions) , Component.SendMessageUpwards(string, object) , Component.SendMessageUpwards(string) , Component.SendMessageUpwards(string, SendMessageOptions) ,

Component.SendMessage(string, object) , Component.SendMessage(string) , Component.SendMessage(string, object, SendMessageOptions) , Component.SendMessage(string, SendMessageOptions) , Component.BroadcastMessage(string, object, SendMessageOptions) , Component.BroadcastMessage(string, object) , Component.BroadcastMessage(string) , Component.BroadcastMessage(string, SendMessageOptions) , Component.transform , Component.gameObject , Component.tag , Object.GetInstanceID() , Object.GetHashCode() , Object.Equals(object) , Object.InstantiateAsync<T>(T) , Object.InstantiateAsync<T>(T, Transform) , Object.InstantiateAsync<T>(T, Vector3, Quaternion) , Object.InstantiateAsync<T>(T, Transform, Vector3, Quaternion) , Object.InstantiateAsync<T>(T, int) , Object.InstantiateAsync<T>(T, int, Transform) , Object.InstantiateAsync<T>(T, int, Vector3, Quaternion) , Object.InstantiateAsync<T>(T, int, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>) , Object.InstantiateAsync<T>(T, int, Transform, Vector3, Quaternion) , Object.InstantiateAsync<T>(T, int, Transform, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>) , Object.Instantiate(Object, Vector3, Quaternion) , Object.Instantiate(Object, Vector3, Quaternion, Transform) , Object.Instantiate(Object) , Object.Instantiate(Object, Scene) , Object.Instantiate(Object, Transform) , Object.Instantiate(Object, Transform, bool) , Object.Instantiate<T>(T) , Object.Instantiate<T>(T, Vector3, Quaternion) , Object.Instantiate<T>(T, Vector3, Quaternion, Transform) , Object.Instantiate<T>(T, Transform) , Object.Instantiate<T>(T, Transform, bool) , Object.Destroy(Object, float) , Object.Destroy(Object) , Object.DestroyImmediate(Object, bool) , Object.DestroyImmediate(Object) , Object.FindObjectsOfType(Type) , Object.FindObjectsOfType(Type, bool) , Object.FindObjectsByType(Type, FindObjectsSortMode) , Object.FindObjectsByType(Type, FindObjectsInactive, FindObjectsSortMode) , Object.DontDestroyOnLoad(Object) , Object.DestroyObject(Object, float) , Object.DestroyObject(Object) , Object.FindSceneObjectsOfType(Type) , Object.FindObjectsOfTypeIncludingAssets(Type) , Object.FindObjectsOfType<T>() , Object.FindObjectsByType<T>(FindObjectsSortMode) , Object.FindObjectsOfType<T>(bool) , Object.FindObjectsByType<T>(FindObjectsInactive, FindObjectsSortMode) , Object.FindObjectOfType<T>() , Object.FindObjectOfType<T>(bool) , Object.FindFirstObjectByType<T>() , Object.FindAnyObjectByType<T>() , Object.FindFirstObjectByType<T>(FindObjectsInactive) , Object.FindAnyObjectByType<T>(FindObjectsInactive) , Object.FindObjectsOfTypeAll(Type) , Object.FindObjectOfType(Type) , Object.FindFirstObjectByType(Type) , Object.FindAnyObjectByType(Type) , Object.FindObjectOfType(Type, bool) , Object.FindFirstObjectByType(Type, FindObjectsInactive) , Object.FindAnyObjectByType(Type, FindObjectsInactive) , Object.ToString() , Object.name ,

Object.hideFlags , object.Equals(object, object) , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Methods

GetDailyWord()

Cette méthode asynchrone demande le mot du jour à PlayFab, et le retourne. Ce mot du jour est mis à jour tous les jours, à 2h du matin.

```
public Task<string> GetDailyWord()
```

Returns

Task <string >

Le mot du jour.

GetDailyWordLeaderboard()

Cette fonction asynchrone récupère le classement des résultats du mot du jour du top 50 des joueurs globaux.

```
public Task<List<PlayerLeaderboardEntry>> GetDailyWordLeaderboard()
```

Returns

Task <List <PlayerLeaderboardEntry> >

La liste des classements des joueurs pour le mot du jour. Se réinitialise chaque jour à 2h du matin.

GetHasPlayedDailyWord()

Cette méthode asynchrone récupère si oui ou non le joueur à joué à la partie du jour aujourd'hui. Cette valeur se réinitialise tous les jours, à 2h du matin.

```
public Task<bool> GetHasPlayedDailyWord()
```

Returns

Task < bool >

True si le joueur à fini la partie du jour, False sinon.

GetPlayerCoins()

Cette fonction retourne le nombre de pièces possédées par le joueur sur PlayFab, par le biais d'une tâche. Il est fortement recommendé d'attendre la fin d'exécution de la tâche avant d'essayer d'extraire le résultat.

```
public Task<int> GetPlayerCoins()
```

Returns

Task < int >

Une tâche contenant le nombre de pièces du joueur une fois terminée, contenu dans la propriété Result.

GetPlayerGamesPlayed()

Cette fonction retourne le nombre de parties le joueur a joué sur PlayFab, par le biais d'une tâche. Il est fortement recommendé d'attendre la fin d'exécution de la tâche avant d'essayer d'extraire le résultat.

```
public Task<int> GetPlayerGamesPlayed()
```

Returns

Task < int >

Une tâche contenant le nombre de parties jouées par le joueur une fois terminée, contenu dans la propriété Result.

GetPlayerGamesWon()

Cette fonction retourne le nombre de parties le joueur a gagné sur PlayFab, par le biais d'une tâche. Il est fortement recommendé d'attendre la fin d'exécution de la tâche avant d'essayer d'extraire le résultat.

```
public Task<int> GetPlayerGamesWon()
```

Returns

Task<int>

Une tâche contenant le nombre de parties gagnées par le joueur une fois terminée, contenu dans la propriété Result.

GetStoredHasPlayedDailyWord()

Cette méthode retourne le booléen selon si le joueur à terminé la partie du jour ou non, stockée en mémoire pour économiser des appels API.

```
public bool GetStoredHasPlayedDailyWord()
```

Returns

bool

True si le joueur à fini la partie du jour, False sinon.

GetStoredUsername()

Cette fonction permet de retourner le nom d'utilisateur stocké en mémoire. Si aucune des méthodes GetUsername ou SetUsername n'a été appelé avant cela, cette valeur est null.

```
public string GetStoredUsername()
```

Returns

string

Le nom d'utilisateur du joueur tel que stocké en mémoire.

GetThis()

Cette fonction statique retourne l'instance de PlayFabController active. Vu qu'il s'agit d'un Singleton, il ne peut y en avoir qu'un seul.

```
public static PlayFabController GetThis()
```

Returns

PlayFabController

L'instance de PlayFabController existante.

GetUsername()

Cette fonction retourne le nom d'utilisateur du joueur sur PlayFab, par le biais d'une tâche. Il est fortement recommendé d'attendre la fin d'exécution de la tâche avant d'essayer d'extraire le résultat.

Afin d'éviter de faire des appels d'API constamment pour obtenir le nom d'utilisateur, il est aussi sauvegardé dans la variable storedUsername, pouvant être récupéré avec GetStoredUsername.

```
public Task<string> GetUsername()
```

Returns

Task<string>

Une tâche contenant le nom d'utilisateur une fois terminée, contenu dans la propriété Result.

InitialisePlayerProfile()

Cette méthode initialise les données du joueur, avec une valeur de départ de pièces, parties jouées et parties gagnées à 0. Cette méthode est appelée après la connexion du joueur, en cas de nouveau compte.

```
public void InitialisePlayerProfile()
```

SendDailyWordScore(int)

Cette méthode envoie le score obtenu dans la partie journalière sur la plateforme PlayFab en tant que statistique du joueur. Cette statistique est réinitialisée chaque jour.

```
public void SendDailyWordScore(int score)
```

Parameters

`score` int

SetPlayerCoins(int)

Cette péthode met à jour le nombre de pièces que le joueur possède. Il est recommendé de d'abord obtenir le nombre de pièces qu'il possède afin de pouvoir calculer cette valeur.

```
public void SetPlayerCoins(int coins)
```

Parameters

`coins` int

Le nombre de pièces que le joueur possédera après l'exécution de cette méthode. À noter que l'appel à l'API peut échouer.

SetPlayerGamesPlayed(int)

Cette méthode met à jour le nombre de parties que le joueur a jouées. Il est recommendé de d'abord obtenir le nombre de pièces qu'il possède afin de pouvoir calculer cette valeur.

```
public void SetPlayerGamesPlayed(int gamesPlayed)
```

Parameters

`gamesPlayed` int

Le nouveau nombre de parties jouées, généralement la valeur actuelle +1.

SetPlayerGamesWon(int)

Cette méthode met à jour le nombre de parties que le joueur a gagnées. Il est recommendé de d'abord obtenir le nombre de pièces qu'il possède afin de pouvoir calculer cette valeur.

```
public void SetPlayerGamesWon(int gamesWon)
```

Parameters

`gamesWon` int

Le nouveau nombre de parties gagnées, généralement la valeur actuelle +1.

SetUsername(string)

Cette méthode permet de changer le nom d'utilisateur du joueur sur la plateforme PlayFab.

Ce nouveau nom d'utilisateur sera aussi stocké en mémoire dans storedUsername, pouvant être récupéré avec GetStoredUsername.

```
public void SetUsername(string newUsername)
```

Parameters

`newUsername` string

Le nouveau nom d'utilisateur du joueur.

TellPlayedDailyWord()

Cette méthode indique à PlayFab que le joueur à fini une partie journalière, et ne peut donc plus y jouer à nouveau jusqu'au jour suivant.

```
public void TellPlayedDailyWord()
```

Class UIController

Namespace: MasterMot

Assembly: MasterMot.dll

```
public class UIController : MonoBehaviour
```

Inheritance

object ← Object ← Component ← Behaviour ← MonoBehaviour ← UIController

Inherited Members

MonoBehaviour.IsInvoking() , MonoBehaviour.CancelInvoke() , MonoBehaviour.Invoke(string, float) , MonoBehaviour.InvokeRepeating(string, float, float) , MonoBehaviour.CancelInvoke(string) , MonoBehaviour.IsInvoking(string) , MonoBehaviour.StartCoroutine(string) , MonoBehaviour.StartCoroutine(string, object) , MonoBehaviour.StartCoroutine(IEnumerator) , MonoBehaviour.StartCoroutine_Auto(IEnumerator) , MonoBehaviour.StopCoroutine(IEnumerator) , MonoBehaviour.StopCoroutine(Coroutine) , MonoBehaviour.StopCoroutine(string) , MonoBehaviour.StopAllCoroutines() , MonoBehaviour.print(object) , MonoBehaviour.destroyCancellationToken , MonoBehaviour.useGUILayout , MonoBehaviour.runInEditMode , Behaviour.enabled , Behaviour.isActiveAndEnabled , Component.GetComponent(Type) , Component.GetComponent<T>() , Component.TryGetComponent(Type, out Component) , Component.TryGetComponent<T>(out T) , Component.GetComponent(string) , Component.GetComponentInChildren(Type, bool) , Component.GetComponentInChildren(Type) , Component.GetComponentInChildren<T>(bool) , Component.GetComponentInChildren<T>() , Component.GetComponentsInChildren(Type, bool) , Component.GetComponentsInChildren(Type) , Component.GetComponentsInChildren<T>(bool) , Component.GetComponentsInChildren<T>(bool, List<T>) , Component.GetComponentsInChildren<T>() , Component.GetComponentsInChildren<T>(List<T>) , Component.GetComponentInParent(Type, bool) , Component.GetComponentInParent(Type) , Component.GetComponentInParent<T>(bool) , Component.GetComponentInParent<T>() , Component.GetComponentsInParent(Type, bool) , Component.GetComponentsInParent(Type) , Component.GetComponentsInParent<T>(bool) , Component.GetComponentsInParent<T>(bool, List<T>) , Component.GetComponentsInParent<T>() , Component.GetComponents(Type) , Component.GetComponents(Type, List<Component>) , Component.GetComponents<T>(List<T>) , Component.GetComponents<T>() , Component.GetComponentIndex() , Component.CompareTag(string) , Component.SendMessageUpwards(string, object, SendMessageOptions) , Component.SendMessageUpwards(string, object) , Component.SendMessageUpwards(string) , Component.SendMessageUpwards(string, SendMessageOptions) ,

Component.SendMessage(string, object) , Component.SendMessage(string) ,
Component.SendMessage(string, object, SendMessageOptions) ,
Component.SendMessage(string, SendMessageOptions) ,
Component.BroadcastMessage(string, object, SendMessageOptions) ,
Component.BroadcastMessage(string, object) , Component.BroadcastMessage(string) ,
Component.BroadcastMessage(string, SendMessageOptions) , Component.transform ,
Component.gameObject , Component.tag , Object.GetInstanceID() , Object.GetHashCode() ,
Object.Equals(object) , Object.InstantiateAsync<T>(T) , Object.InstantiateAsync<T>(T, Transform) ,
Object.InstantiateAsync<T>(T, Vector3, Quaternion) ,
Object.InstantiateAsync<T>(T, Transform, Vector3, Quaternion) , Object.InstantiateAsync<T>(T, int) ,
Object.InstantiateAsync<T>(T, int, Transform) ,
Object.InstantiateAsync<T>(T, int, Vector3, Quaternion) ,
Object.InstantiateAsync<T>(T, int, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>) ,
Object.InstantiateAsync<T>(T, int, Transform, Vector3, Quaternion) ,
Object.InstantiateAsync<T>(T, int, Transform, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>) ,
Object.Instantiate(Object, Vector3, Quaternion) ,
Object.Instantiate(Object, Vector3, Quaternion, Transform) , Object.Instantiate(Object) ,
Object.Instantiate(Object, Scene) , Object.Instantiate(Object, Transform) ,
Object.Instantiate(Object, Transform, bool) , Object.Instantiate<T>(T) ,
Object.Instantiate<T>(T, Vector3, Quaternion) ,
Object.Instantiate<T>(T, Vector3, Quaternion, Transform) , Object.Instantiate<T>(T, Transform) ,
Object.Instantiate<T>(T, Transform, bool) , Object.Destroy(Object, float) , Object.Destroy(Object) ,
Object.DestroyImmediate(Object, bool) , Object.DestroyImmediate(Object) ,
Object.FindObjectsOfType(Type) , Object.FindObjectsOfType(Type, bool) ,
Object.FindObjectsByType(Type, FindObjectsSortMode) ,
Object.FindObjectsByType(Type, FindObjectsInactive, FindObjectsSortMode) ,
Object.DontDestroyOnLoad(Object) , Object.DestroyObject(Object, float) ,
Object.DestroyObject(Object) , Object.FindSceneObjectsOfType(Type) ,
Object.FindObjectsOfTypeIncludingAssets(Type) , Object.FindObjectsOfType<T>() ,
Object.FindObjectsByType<T>(FindObjectsSortMode) , Object.FindObjectsOfType<T>(bool) ,
Object.FindObjectsByType<T>(FindObjectsInactive, FindObjectsSortMode) ,
Object.FindObjectOfType<T>() , Object.FindObjectOfType<T>(bool) ,
Object.FindFirstObjectByType<T>() , Object.FindAnyObjectByType<T>() ,
Object.FindFirstObjectByType<T>(FindObjectsInactive) ,
Object.FindAnyObjectByType<T>(FindObjectsInactive) , Object.FindObjectsOfTypeAll(Type) ,
Object.FindObjectOfType(Type) , Object.FindFirstObjectByType(Type) ,
Object.FindAnyObjectByType(Type) , Object.FindObjectOfType(Type, bool) ,
Object.FindFirstObjectByType(Type, FindObjectsInactive) ,
Object.FindAnyObjectByType(Type, FindObjectsInactive) , Object.ToString() , Object.name ,

Object.hideFlags , object.Equals(object, object) , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Methods

ChangeScreen(int)

Cette méthode permet de changer d'une scène à une autre.

```
public void ChangeScreen(int sceneID)
```

Parameters

`sceneID` int

Le numéro de scène tel qu'indiqué dans la fenêtre Build Settings de Unity.

DisableEndScreen()

Cette méthode permet de désactiver les UI Documents de fin de partie, et réinitialise leurs textes.

```
public void DisableEndScreen()
```

EnableDefeatScreen(string)

Cette méthode permet d'afficher l'écran de fin de partie en cas de défaite.

```
public void EnableDefeatScreen(string word)
```

Parameters

`word` string

Le mot qu'il fallait découvrir durant la partie.

EnableFirstTimeUsernameSetter()

Cette méthode permet d'afficher l'écran permettant de donner au joueur un nom d'utilisateur.

```
public void EnableFirstTimeUsernameSetter()
```

EnableVictoryScreen(string)

Cette méthode permet d'afficher l'écran de fin de partie en cas de victoire.

```
public void EnableVictoryScreen(string word)
```

Parameters

`word` string

Le mot qu'il fallait découvrir durant la partie.

GetDefeatDocument()

Cette fonction retourne l'UI Document de l'instance actuelle de l'écran de défaite.

```
public UIDocument GetDefeatDocument()
```

Returns

UIDocument

L'UI Document de l'écran de défaite actuellement affiché.

GetDifficultyPicker()

Cette fonction retourne l'UI Document de l'instance actuelle de sélection de difficultés.

```
public UIDocument GetDifficultyPicker()
```

Returns

UIDocument

L'UI Document de l'écran de sélection de difficultés actuellement affiché.

GetLoadedSettings()

Cette fonction retourne l'UI Document de l'instance actuelle de l'écran d'options.

```
public UIDocument GetLoadedSettings()
```

Returns

UIDocument

L'UI Document de l'écran d'options actuellement affiché.

GetMainMenuDocument()

Cette fonction retourne l'UI Document de l'instance actuelle de l'écran du menu principal.

```
public UIDocument GetMainMenuDocument()
```

Returns

UIDocument

L'UI Document de l'écran du menu principal actuellement affiché.

GetThis()

Cette fonction statique retourne l'instance d'UIController active. Vu qu'il s'agit d'un Singleton, il ne peut y en avoir qu'un seul.

```
public static UIController GetThis()
```

Returns

UIController

Retourne l'instance de UIController existante.

GetUsernameSetter()

Cette fonction retourne l'UI Document de l'écran de création de nom d'utilisateur.

```
public UIDocument GetUsernameSetter()
```

Returns

UIDocument

L'UI Document de l'écran de création de nom d'utilisateur actuellement affiché.

GetVictoryDocument()

Cette fonction retourne l'UI Document de l'instance actuelle de l'écran de victoire.

```
public UIDocument GetVictoryDocument()
```

Returns

UIDocument

L'UI Document de l'écran de victoire actuellement affiché.

GoBackFromDifficulty()

Cette méthode désactive l'écran de sélection de difficulté, retournant le joueur à l'écran principal.

```
public void GoBackFromDifficulty()
```

LoadSettingsMenu()

Cette méthode permet d'instancier et de charger l'écran d'options du jeu, là où le joueur peut changer les volumes de son, activer les notifications Push, et revenir au menu principal s'il était en partie.

```
public void LoadSettingsMenu()
```

PickDifficulty()

Cette méthode active l'écran de sélection de difficulté pour une nouvelle partie, et y attribue les callbacks pour les boutons.

```
public void PickDifficulty()
```

UnloadSettingsMenu()

Cette méthode détruit l'instance d'écran d'options, rammenant l'écran de jeu là où il était auparavant.

```
public void UnloadSettingsMenu()
```

Class WordGameController

Namespace: MasterMot

Assembly: MasterMot.dll

```
public class WordGameController : MonoBehaviour
```

Inheritance

object ← Object ← Component ← Behaviour ← MonoBehaviour ← WordGameController

Inherited Members

MonoBehaviour.IsInvoking() , MonoBehaviour.CancelInvoke() , MonoBehaviour.Invoke(string, float) , MonoBehaviour.InvokeRepeating(string, float, float) , MonoBehaviour.CancelInvoke(string) , MonoBehaviour.IsInvoking(string) , MonoBehaviour.StartCoroutine(string) , MonoBehaviour.StartCoroutine(string, object) , MonoBehaviour.StartCoroutine(IEnumerator) , MonoBehaviour.StartCoroutine_Auto(IEnumerator) , MonoBehaviour.StopCoroutine(IEnumerator) , MonoBehaviour.StopCoroutine(Coroutine) , MonoBehaviour.StopCoroutine(string) , MonoBehaviour.StopAllCoroutines() , MonoBehaviour.print(object) , MonoBehaviour.destroyCancellationToken , MonoBehaviour.useGUILayout , MonoBehaviour.runInEditMode , Behaviour.enabled , Behaviour.isActiveAndEnabled , Component.GetComponent(Type) , Component.GetComponent<T>() , Component.TryGetComponent(Type, out Component) , Component.TryGetComponent<T>(out T) , Component.GetComponent(string) , Component.GetComponentInChildren(Type, bool) , Component.GetComponentInChildren(Type) , Component.GetComponentInChildren<T>(bool) , Component.GetComponentInChildren<T>() , Component.GetComponentsInChildren(Type, bool) , Component.GetComponentsInChildren(Type) , Component.GetComponentsInChildren<T>(bool) , Component.GetComponentsInChildren<T>(bool, List<T>) , Component.GetComponentsInChildren<T>() , Component.GetComponentsInChildren<T>(List<T>) , Component.GetComponentInParent(Type, bool) , Component.GetComponentInParent(Type) , Component.GetComponentInParent<T>(bool) , Component.GetComponentInParent<T>() , Component.GetComponentsInParent(Type, bool) , Component.GetComponentsInParent(Type) , Component.GetComponentsInParent<T>(bool) , Component.GetComponentsInParent<T>(bool, List<T>) , Component.GetComponentsInParent<T>() , Component.GetComponents(Type) , Component.GetComponents(Type, List<Component>) , Component.GetComponents<T>(List<T>) , Component.GetComponents<T>() , Component.GetComponentIndex() , Component.CompareTag(string) , Component.SendMessageUpwards(string, object, SendMessageOptions) , Component.SendMessageUpwards(string, object) , Component.SendMessageUpwards(string) , Component.SendMessageUpwards(string, SendMessageOptions) ,

Component.SendMessage(string, object) , Component.SendMessage(string) , Component.SendMessage(string, object, SendMessageOptions) , Component.SendMessage(string, SendMessageOptions) , Component.BroadcastMessage(string, object, SendMessageOptions) , Component.BroadcastMessage(string, object) , Component.BroadcastMessage(string) , Component.BroadcastMessage(string, SendMessageOptions) , Component.transform , Component.gameObject , Component.tag , Object.GetInstanceID() , Object.GetHashCode() , Object.Equals(object) , Object.InstantiateAsync<T>(T) , Object.InstantiateAsync<T>(T, Transform) , Object.InstantiateAsync<T>(T, Vector3, Quaternion) , Object.InstantiateAsync<T>(T, Transform, Vector3, Quaternion) , Object.InstantiateAsync<T>(T, int) , Object.InstantiateAsync<T>(T, int, Transform) , Object.InstantiateAsync<T>(T, int, Vector3, Quaternion) , Object.InstantiateAsync<T>(T, int, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>) , Object.InstantiateAsync<T>(T, int, Transform, Vector3, Quaternion) , Object.InstantiateAsync<T>(T, int, Transform, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>) , Object.Instantiate(Object, Vector3, Quaternion) , Object.Instantiate(Object, Vector3, Quaternion, Transform) , Object.Instantiate(Object) , Object.Instantiate(Object, Scene) , Object.Instantiate(Object, Transform) , Object.Instantiate(Object, Transform, bool) , Object.Instantiate<T>(T) , Object.Instantiate<T>(T, Vector3, Quaternion) , Object.Instantiate<T>(T, Vector3, Quaternion, Transform) , Object.Instantiate<T>(T, Transform) , Object.Instantiate<T>(T, Transform, bool) , Object.Destroy(Object, float) , Object.Destroy(Object) , Object.DestroyImmediate(Object, bool) , Object.DestroyImmediate(Object) , Object.FindObjectsOfType(Type) , Object.FindObjectsOfType(Type, bool) , Object.FindObjectsByType(Type, FindObjectsSortMode) , Object.FindObjectsByType(Type, FindObjectsInactive, FindObjectsSortMode) , Object.DontDestroyOnLoad(Object) , Object.DestroyObject(Object, float) , Object.DestroyObject(Object) , Object.FindSceneObjectsOfType(Type) , Object.FindObjectsOfTypeIncludingAssets(Type) , Object.FindObjectsOfType<T>() , Object.FindObjectsByType<T>(FindObjectsSortMode) , Object.FindObjectsOfType<T>(bool) , Object.FindObjectsByType<T>(FindObjectsInactive, FindObjectsSortMode) , Object.FindObjectOfType<T>() , Object.FindObjectOfType<T>(bool) , Object.FindFirstObjectByType<T>() , Object.FindAnyObjectByType<T>() , Object.FindFirstObjectByType<T>(FindObjectsInactive) , Object.FindAnyObjectByType<T>(FindObjectsInactive) , Object.FindObjectsOfTypeAll(Type) , Object.FindObjectOfType(Type) , Object.FindFirstObjectByType(Type) , Object.FindAnyObjectByType(Type) , Object.FindObjectOfType(Type, bool) , Object.FindFirstObjectByType(Type, FindObjectsInactive) , Object.FindAnyObjectByType(Type, FindObjectsInactive) , Object.ToString() , Object.name ,

Object.hideFlags , object.Equals(object, object) , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Methods

AddAttempt(string)

Cette méthode ajoute le mot dans la liste de tentatives faites.

```
public void AddAttempt(string attempt)
```

Parameters

`attempt` string

La nouvelle tentative.

CheckWord(string)

Cette méthode génère une liste de LetterStatusEnum qu'elle passe ensuite au Renderer pour mettre à jour le rendu du jeu.

Elle commence d'abord par vérifier quelles lettres sont correctes et au bon endroit, puis passe à nouveau pour déterminer si des lettres sont corrects mais au mauvais endroit.

```
public void CheckWord(string word)
```

Parameters

`word` string

Le mot à comparer avec le mot à trouver.

GetAttempts()

Cette fonction retourne la liste de tentatives faites par le joueur.

```
public List<string> GetAttempts()
```

Returns

List<string>

La liste de tentatives faites par le joueur.

GetDailyWord()

Cette fonction statique retourne si, oui ou non, la partie en cours est une partie du jour ou une partie au hasard.

```
public static bool GetDailyWord()
```

Returns

bool

True s'il s'agit d'une partie du jour, False s'il s'agit d'une partie aléatoire.

GetMaxAttempts()

Cette fonction retourne le nombre de tentatives maximum que le joueur peut faire. Cette valeur est déterminée par la difficulté de jeu.

```
public int GetMaxAttempts()
```

Returns

int

Le nombre de tentatives maximum.

GetMot()

Cette fonction retourne le mot à deviner dans la partie en cours.

```
public string GetMot()
```

Returns

string

Le mot à deviner.

IsWordCorrect(string)

Cette fonction détermine si le mot entré est identique au mot à deviner.

```
public bool IsWordCorrect(string input)
```

Parameters

`input` string

Le mot à comparer.

Returns

bool

True si le mot entré est le même que le mot à deviner, False sinon.

IsWordValid(string)

Cette fonction détermine si le mot entré peut être comparé avec le mot à deviner, en s'assurant qu'il soit aussi long.

```
public bool IsWordValid(string input)
```

Parameters

`input` string

Le mot à vérifier.

Returns

bool

True si le mot entré est aussi long que le mot à deviner, False sinon.

MakeNewAttempt()

Cette méthode permet d'effectuer une nouvelle tentative avec le mot inséré dans le Renderer. Si le mot n'est pas valide, la tentative n'est pas effectuée.

Si la tentative est le mot à deviner, affiche l'écran de victoire. S'il s'agit de la dernière tentative, affiche l'écran de défaite.

```
public void MakeNewAttempt()
```

Restart()

Cette méthode réinitialise l'affichage de l'écran de jeu, et récupère un nouveau mot selon la difficulté de la partie précédente.

```
public void Restart()
```

SetDailyWord(bool)

Cette méthode statique détermine si la partie à lancer est la partie du jour ou une partie au hasard.

La partie du jour est fixe pour tous les joueurs chaque jour.

```
public static void SetDailyWord(bool setter)
```

Parameters

`setter` bool

Détermine si la partie est une partie du jour, oui ou non.

SetDifficulty(DifficultyEnum)

Cette méthode statique change la difficulté de la partie à lancer. Cette difficulté affecte la taille du mot à découvrir, ainsi que le nombre de tentatives.

```
public static void SetDifficulty(DifficultyEnum newDifficulty)
```

Parameters

`newDifficulty` DifficultyEnum

La difficulté de la future partie.

SetWord(string)

Cette méthode change le mot à deviner, généralement à l'initialisation de la partie lorsque l'appel de l'API est fait.

```
public void SetWord(string word)
```

Parameters

`word` string

Le nouveau mot à deviner.

Class WordGameRenderer

Namespace: MasterMot

Assembly: MasterMot.dll

```
public class WordGameRenderer : MonoBehaviour
```

Inheritance

object ← Object ← Component ← Behaviour ← MonoBehaviour ← WordGameRenderer

Inherited Members

MonoBehaviour.IsInvoking() , MonoBehaviour.CancelInvoke() , MonoBehaviour.Invoke(string, float) , MonoBehaviour.InvokeRepeating(string, float, float) , MonoBehaviour.CancelInvoke(string) , MonoBehaviour.IsInvoking(string) , MonoBehaviour.StartCoroutine(string) , MonoBehaviour.StartCoroutine(string, object) , MonoBehaviour.StartCoroutine(IEnumerator) , MonoBehaviour.StartCoroutine_Auto(IEnumerator) , MonoBehaviour.StopCoroutine(IEnumerator) , MonoBehaviour.StopCoroutine(Coroutine) , MonoBehaviour.StopCoroutine(string) , MonoBehaviour.StopAllCoroutines() , MonoBehaviour.print(object) , MonoBehaviour.destroyCancellationToken , MonoBehaviour.useGUILayout , MonoBehaviour.runInEditMode , Behaviour.enabled , Behaviour.isActiveAndEnabled , Component.GetComponent(Type) , Component.GetComponent<T>() , Component.TryGetComponent(Type, out Component) , Component.TryGetComponent<T>(out T) , Component.GetComponent(string) , Component.GetComponentInChildren(Type, bool) , Component.GetComponentInChildren(Type) , Component.GetComponentInChildren<T>(bool) , Component.GetComponentInChildren<T>() , Component.GetComponentsInChildren(Type, bool) , Component.GetComponentsInChildren(Type) , Component.GetComponentsInChildren<T>(bool) , Component.GetComponentsInChildren<T>(bool, List<T>) , Component.GetComponentsInChildren<T>() , Component.GetComponentsInChildren<T>(List<T>) , Component.GetComponentInParent(Type, bool) , Component.GetComponentInParent(Type) , Component.GetComponentInParent<T>(bool) , Component.GetComponentInParent<T>() , Component.GetComponentsInParent(Type, bool) , Component.GetComponentsInParent(Type) , Component.GetComponentsInParent<T>(bool) , Component.GetComponentsInParent<T>(bool, List<T>) , Component.GetComponentsInParent<T>() , Component.GetComponents(Type) , Component.GetComponents(Type, List<Component>) , Component.GetComponents<T>(List<T>) , Component.GetComponents<T>() , Component.GetComponentIndex() , Component.CompareTag(string) , Component.SendMessageUpwards(string, object, SendMessageOptions) , Component.SendMessageUpwards(string, object) , Component.SendMessageUpwards(string) , Component.SendMessageUpwards(string, SendMessageOptions) ,

Component.SendMessage(string, object) , Component.SendMessage(string) , Component.SendMessage(string, object, SendMessageOptions) , Component.SendMessage(string, SendMessageOptions) , Component.BroadcastMessage(string, object, SendMessageOptions) , Component.BroadcastMessage(string, object) , Component.BroadcastMessage(string) , Component.BroadcastMessage(string, SendMessageOptions) , Component.transform , Component.gameObject , Component.tag , Object.GetInstanceID() , Object.GetHashCode() , Object.Equals(object) , Object.InstantiateAsync<T>(T) , Object.InstantiateAsync<T>(T, Transform) , Object.InstantiateAsync<T>(T, Vector3, Quaternion) , Object.InstantiateAsync<T>(T, Transform, Vector3, Quaternion) , Object.InstantiateAsync<T>(T, int) , Object.InstantiateAsync<T>(T, int, Transform) , Object.InstantiateAsync<T>(T, int, Vector3, Quaternion) , Object.InstantiateAsync<T>(T, int, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>) , Object.InstantiateAsync<T>(T, int, Transform, Vector3, Quaternion) , Object.InstantiateAsync<T>(T, int, Transform, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>) , Object.Instantiate(Object, Vector3, Quaternion) , Object.Instantiate(Object, Vector3, Quaternion, Transform) , Object.Instantiate(Object) , Object.Instantiate(Object, Scene) , Object.Instantiate(Object, Transform) , Object.Instantiate(Object, Transform, bool) , Object.Instantiate<T>(T) , Object.Instantiate<T>(T, Vector3, Quaternion) , Object.Instantiate<T>(T, Vector3, Quaternion, Transform) , Object.Instantiate<T>(T, Transform) , Object.Instantiate<T>(T, Transform, bool) , Object.Destroy(Object, float) , Object.Destroy(Object) , Object.DestroyImmediate(Object, bool) , Object.DestroyImmediate(Object) , Object.FindObjectsOfType(Type) , Object.FindObjectsOfType(Type, bool) , Object.FindObjectsByType(Type, FindObjectsSortMode) , Object.FindObjectsByType(Type, FindObjectsInactive, FindObjectsSortMode) , Object.DontDestroyOnLoad(Object) , Object.DestroyObject(Object, float) , Object.DestroyObject(Object) , Object.FindSceneObjectsOfType(Type) , Object.FindObjectsOfTypeIncludingAssets(Type) , Object.FindObjectsOfType<T>() , Object.FindObjectsByType<T>(FindObjectsSortMode) , Object.FindObjectsOfType<T>(bool) , Object.FindObjectsByType<T>(FindObjectsInactive, FindObjectsSortMode) , Object.FindObjectOfType<T>() , Object.FindObjectOfType<T>(bool) , Object.FindFirstObjectByType<T>() , Object.FindAnyObjectByType<T>() , Object.FindFirstObjectByType<T>(FindObjectsInactive) , Object.FindAnyObjectByType<T>(FindObjectsInactive) , Object.FindObjectsOfTypeAll(Type) , Object.FindObjectOfType(Type) , Object.FindFirstObjectByType(Type) , Object.FindAnyObjectByType(Type) , Object.FindObjectOfType(Type, bool) , Object.FindFirstObjectByType(Type, FindObjectsInactive) , Object.FindAnyObjectByType(Type, FindObjectsInactive) , Object.ToString() , Object.name ,

Object.hideFlags , object.Equals(object, object) , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Methods

AnimateAttemptRow(List<LetterStatusEnum>)

Cette méthode asynchrone permet d'animer les cellules de lettre de la tentative actuelle, avec l'aide de la méthode DOShakeVisualElement définie ci-dessous.

```
public void AnimateAttemptRow(List<LetterStatusEnum> colourList)
```

Parameters

`colourList` List<LetterStatusEnum>

La liste de statuts de lettres pour chaque cellule. Cela affecte la coloration de la cellule, ainsi que l'effet sonore qui est joué.

ColorLetterCell(VisualElement, LetterStatusEnum)

Cette méthode permet de colorer l'arrière plan d'une cases de lettre. Cela se fait par l'ajout d'une classe USS en fonction du type de résultat obtenu pour la lettre.

Pour customiser l'arrière plan de chaque classe, référez-vous au fichier Assets > UI Toolkit > Style Sheets > WordGameUSS.uss .

```
public void ColorLetterCell(VisualElement cell, LetterStatusEnum letterStatus)
```

Parameters

`cell` VisualElement

Le VisualElement de la cellule de lettre à colorer.

`letterStatus` LetterStatusEnum

Le statut de la lettre à colorer, selon ses 3 états possibles : bonne lettre au bon endroit, bonne lettre au mauvais endroit, et mauvaise lettre.

DrawLetterCells()

Cette méthode crée une case de lettre par lettres dans le mot, par tentatives disponibles. Se sert du template inséré dans letterCellAsset.

```
public void DrawLetterCells()
```

GetKeyboardKeys()

Cette fonction retourne la liste de tous les boutons du clavier du document de jeu. Cela inclus le bouton de retour arrière, et le bouton d'envoie de tentatives.

```
public List<Button> GetKeyboardKeys()
```

Returns

List<Button>

La liste des boutons du clavier dans le document.

GetMot()

Cette fonction retourne le mot inséré dans la tentative courrante.

```
public string GetMot()
```

Returns

string

Le mot inséré dans la tentative courante. Il peut ne pas être aussi long que le mot à trouver.

GetSettingsButton()

Cette fonction retourne le bouton des options présent dans l'écran de jeu.

```
public Button GetSettingsButton()
```

Returns

Button

Le bouton d'options dans l'écran de jeu.

IncrementAttempts()

Cette méthode permet de passer à la tentative suivante en terme d'affichage.

```
public void IncrementAttempts()
```

InsertLetter(char)

Cette méthode insère la lettre à la fin du mot affiché dans la tentative courrante. Si le mot est déjà à sa taille maximale, cette méthode remplace la dernière lettre.

```
public void InsertLetter(char character)
```

Parameters

`character` char

La lettre à insérer à la fin du mot.

RemoveLastLetter()

Cette méthode retire la dernière lettre du mot dans la tentative courrante. Elle ne fait rien s'il n'y a aucune lettre dans le mot.

```
public void RemoveLastLetter()
```

Restart()

Cette méthode réinitialise l'affichage de la zone de tentatives, afin de pouvoir recommencer une nouvelle partie sans devoir recharger la scène.

```
public void Restart()
```