

InputMaster

v6.3.0

Generated by Doxygen 1.8.8

Fri Oct 10 2014 12:11:47

Contents

1	Namespace Index	1
1.1	Packages	1
2	Hierarchical Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	Namespace Documentation	7
4.1	Package BSGTools	7
4.2	Package BSGTools.IO	7
4.2.1	Enumeration Type Documentation	8
4.2.1.1	ControlState	8
4.3	Package BSGTools.IO.Tools	8
4.4	Package BSGTools.IO.Xbox	8
4.4.1	Enumeration Type Documentation	8
4.4.1.1	XButton	8
4.4.1.2	XStick	9
4.4.1.3	XTrigger	9
5	Class Documentation	11
5.1	BSGTools.IO.Control Class Reference	11
5.1.1	Detailed Description	12
5.1.2	Member Function Documentation	12
5.1.2.1	operator+	12
5.1.2.2	operator+	13
5.1.2.3	Reset	13
5.1.2.4	Reset	13
5.1.2.5	SoftReset	13
5.1.2.6	ToString	13
5.1.2.7	ToStringBlock	13
5.1.2.8	Update	14

5.1.2.9	UpdateStates	14
5.1.2.10	UpdateValues	14
5.1.3	Property Documentation	14
5.1.3.1	Dead	14
5.1.3.2	Down	14
5.1.3.3	FixedValue	14
5.1.3.4	Gravity	14
5.1.3.5	Held	14
5.1.3.6	Invert	14
5.1.3.7	IsBlocked	15
5.1.3.8	IsDebugControl	15
5.1.3.9	Name	15
5.1.3.10	RealValue	15
5.1.3.11	Sensitivity	15
5.1.3.12	Snap	15
5.1.3.13	Up	15
5.1.3.14	Value	15
5.2	BSGTools.IO.Tools.InputManagerWizard Class Reference	15
5.3	BSGTools.IO.InputMaster Class Reference	16
5.3.1	Detailed Description	16
5.3.2	Member Function Documentation	17
5.3.2.1	CreateMaster	17
5.3.2.2	CreateMaster	17
5.3.2.3	DestroyMaster	17
5.3.2.4	ResetAll	17
5.3.2.5	SetBlockAll	17
5.3.2.6	UpdateControls	17
5.3.3	Property Documentation	18
5.3.3.1	AnyControlDown	18
5.3.3.2	AnyControlHeld	18
5.3.3.3	AnyControlUp	18
5.3.3.4	MouseX	18
5.3.3.5	MouseXAxisName	18
5.3.3.6	MouseXRaw	18
5.3.3.7	MouseY	18
5.3.3.8	MouseYAxisName	18
5.3.3.9	MouseYRaw	18
5.4	BSGTools.IO.Xbox.IXboxControl Interface Reference	18
5.4.1	Detailed Description	19
5.5	BSGTools.IO.KeyControl Class Reference	19

5.5.1	Detailed Description	19
5.5.2	Constructor & Destructor Documentation	20
5.5.2.1	KeyControl	20
5.5.2.2	KeyControl	20
5.5.3	Member Function Documentation	20
5.5.3.1	ToStringBlock	20
5.5.3.2	UpdateStates	20
5.5.4	Property Documentation	20
5.5.4.1	Modifier	20
5.5.4.2	Negative	20
5.5.4.3	Positive	20
5.6	BSGTools.IO.ModifierKey Class Reference	21
5.6.1	Detailed Description	22
5.6.2	Member Enumeration Documentation	22
5.6.2.1	ModEnums	22
5.6.3	Member Function Documentation	22
5.6.3.1	FromMEnum	22
5.6.3.2	operator KeyCode	22
5.6.3.3	operator ModifierKey	22
5.6.4	Member Data Documentation	22
5.6.4.1	LAlt	22
5.6.4.2	LCommand	22
5.6.4.3	LCtrl	23
5.6.4.4	LShift	23
5.6.4.5	LWindows	23
5.6.4.6	modifiers	23
5.6.4.7	None	23
5.6.4.8	RAlt	23
5.6.4.9	RCommand	23
5.6.4.10	RCtrl	23
5.6.4.11	RShift	24
5.6.4.12	RWindows	24
5.6.5	Property Documentation	24
5.6.5.1	DisplayName	24
5.6.5.2	UKeyCode	24
5.7	BSGTools.IO.Xbox.XboxControl< T > Class Template Reference	24
5.7.1	Detailed Description	25
5.7.2	Constructor & Destructor Documentation	26
5.7.2.1	XboxControl	26
5.7.3	Member Function Documentation	26

5.7.3.1	CreateClone	26
5.7.3.2	CreateMultiple	26
5.7.4	Property Documentation	27
5.7.4.1	ControllerIndex	27
5.8	BSGTools.IO.Xbox.XButtonControl Class Reference	27
5.8.1	Detailed Description	27
5.8.2	Constructor & Destructor Documentation	28
5.8.2.1	XButtonControl	28
5.8.2.2	XButtonControl	28
5.8.3	Member Function Documentation	28
5.8.3.1	CreateClone	28
5.8.3.2	UpdateStates	28
5.8.4	Property Documentation	28
5.8.4.1	Negative	28
5.8.4.2	Positive	28
5.9	BSGTools.IO.Xbox.XStickControl Class Reference	28
5.9.1	Detailed Description	29
5.9.2	Member Enumeration Documentation	29
5.9.2.1	InvertMode	29
5.9.3	Constructor & Destructor Documentation	30
5.9.3.1	XStickControl	30
5.9.4	Member Function Documentation	31
5.9.4.1	CreateClone	31
5.9.4.2	UpdateStates	31
5.9.4.3	UpdateValues	31
5.9.5	Property Documentation	31
5.9.5.1	InversionMode	31
5.9.5.2	Stick	31
5.9.5.3	StickValue	31
5.10	BSGTools.IO.Xbox.XTriggerControl Class Reference	31
5.10.1	Detailed Description	32
5.10.2	Constructor & Destructor Documentation	32
5.10.2.1	XTriggerControl	32
5.10.3	Member Function Documentation	32
5.10.3.1	CreateClone	32
5.10.3.2	UpdateStates	33
5.10.3.3	UpdateValues	33
5.10.4	Property Documentation	33
5.10.4.1	Trigger	33

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

BSGTools	7
BSGTools.IO	7
BSGTools.IO.Tools	8
BSGTools.IO.Xbox	8

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

BSGTools.IO.Control	11
BSGTools.IO.KeyControl	19
BSGTools.IO.Xbox.XboxControl< T >	24
BSGTools.IO.Xbox.XButtonControl	27
BSGTools.IO.Xbox.XStickControl	28
BSGTools.IO.Xbox.XTriggerControl	31
EditorWindow	
BSGTools.IO.Tools.InputManagerWizard	15
BSGTools.IO.Xbox.IXboxControl	18
BSGTools.IO.Xbox.XButtonControl	27
BSGTools.IO.Xbox.XStickControl	28
BSGTools.IO.Xbox.XTriggerControl	31
BSGTools.IO.ModifierKey	21
MonoBehaviour	
BSGTools.IO.InputMaster	16

Chapter 3

Class Index

3.1 Class List

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

BSGTools.IO.Control	
Provided to give a common base class and common functionality for KeyControl and XboxControl	11
BSGTools.IO.Tools.InputManagerWizard	15
BSGTools.IO.InputMaster	
A single instance of this exists in the application. Updates and maintains all Control states.	16
BSGTools.IO.Xbox.IXboxControl	
Used for type constraints. Contains nor enforces any functionality.	18
BSGTools.IO.KeyControl	
Used for all Keyboard and Mouse Button controls. Any binding present in Unity's KeyCode enumeration is valid here.	19
BSGTools.IO.ModifierKey	
Represents a Modifier for a control.	21
BSGTools.IO.Xbox.XboxControl< T >	24
BSGTools.IO.Xbox.XButtonControl	
Used for integrating Xbox 360 controller digital buttons/inputs into InputMaster's Control system.	27
BSGTools.IO.Xbox.XStickControl	
Used for integrating Xbox 360 controller analog sticks into InputMaster's Control system.	28
BSGTools.IO.Xbox.XTriggerControl	
Used for integrating Xbox 360 controller triggers into InputMaster's Control system.	31

Chapter 4

Namespace Documentation

4.1 Package BSGTools

Namespaces

- package [IO](#)

4.2 Package BSGTools.IO

Namespaces

- package [Tools](#)
- package [Xbox](#)

Classes

- class [Control](#)
Provided to give a common base class and common functionality for [KeyControl](#) and [XboxControl](#).
- class **EnumExt**
Simple extensions class for commonly used enum functionality.
- class [InputMaster](#)
A single instance of this exists in the application. Updates and maintains all [Control](#) states.
- class [KeyControl](#)
Used for all Keyboard and Mouse Button controls. Any binding present in Unity's KeyCode enumeration is valid here.
- class [ModifierKey](#)
Represents a Modifier for a control.

Enumerations

- enum [ControlState](#) {
Positive = 1 << 0, **Negative** = 1 << 1, **Neither** = Positive & Negative, **Both** = Positive | Negative,
Either = Positive ^ Negative }
*Describes the current state of a control's specific state. Use (control.Down/Held/Up & flag) != 0 for Positive & Negative.
Use == for Neither, Both, and Either.*

4.2.1 Enumeration Type Documentation

4.2.1.1 enum BSGTools.IO.ControlState

Describes the current state of a control's specific state. Use (control.Down/Held/Up & flag) != 0 for Positive & Negative. Use == for Neither, Both, and Either.

4.3 Package BSGTools.IO.Tools

Classes

- class [InputManagerWizard](#)

4.4 Package BSGTools.IO.Xbox

Classes

- interface [IXboxControl](#)
Used for type constraints. Contains nor enforces any functionality.
- class [XboxControl< T >](#)
- class **XboxUtils**
A static utility class for minimal required updates for XboxControls.
- class [XButtonControl](#)
Used for integrating [Xbox](#) 360 controller digital buttons/inputs into [InputMaster's Control](#) system.
- class [XStickControl](#)
Used for integrating [Xbox](#) 360 controller analog sticks into [InputMaster's Control](#) system.
- class [XTriggerControl](#)
Used for integrating [Xbox](#) 360 controller triggers into [InputMaster's Control](#) system.

Enumerations

- enum [XStick](#) { **StickLeft, StickRight** }
Represents the two analog sticks of an Xbox 360 Controller.
- enum [XTrigger](#) { **TriggerLeft, TriggerRight** }
Represents the two triggers of an Xbox 360 Controller.
- enum [XButton](#) {
None, A, B, X,
Y, Back, Guide, Start,
StickLeft, StickRight, ShoulderLeft, ShoulderRight,
DPadUp, DPadDown, DPadLeft, DPadRight }
Represents all of the digital buttons of an Xbox 360 Controller.

4.4.1 Enumeration Type Documentation

4.4.1.1 enum BSGTools.IO.Xbox.XButton

Represents all of the digital buttons of an [Xbox](#) 360 Controller.

4.4.1.2 enum BSGTools.IO.Xbox.XStick

Represents the two analog sticks of an Xbox 360 Controller.

4.4.1.3 enum BSGTools.IO.Xbox.XTrigger

Represents the two triggers of an Xbox 360 Controller.

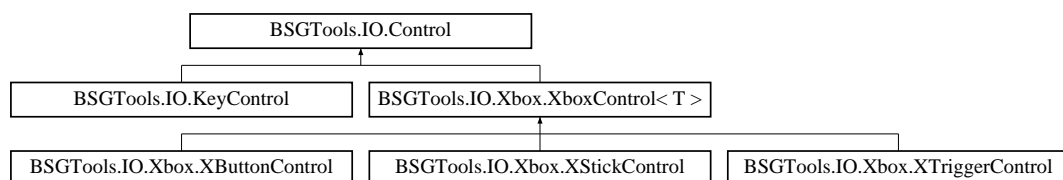
Chapter 5

Class Documentation

5.1 BSGTools.IO.Control Class Reference

Provided to give a common base class and common functionality for [KeyControl](#) and [XboxControl](#).

Inheritance diagram for BSGTools.IO.Control:



Public Member Functions

- void [Update](#) ()
Updates the control. This should never be used by any user-made script. This is public specifically for the use of [InputMaster](#).
- void [Reset](#) ()
Reset all non-configuration values and states for this control.
- void [Reset](#) (bool block)
Reset all non-configuration values for this control. This is the best method to use for cutscenes.
- override string [ToString](#) ()
Returns debug information in a single line.
- virtual string [ToStringBlock](#) ()
Returns debug information as a formatted string block.

Static Public Member Functions

- static float [operator+](#) ([Control](#) one, [Control](#) two)
Adds the [Value](#) properties of a variable amount of controls. The return value is clamped between -1...1 If you have multiple control options for a single in-game action, this is how you gather the final input value for the action.
- static float [operator+](#) ([Control](#) one, float val)
Adds the [Value](#) properties of a variable amount of controls and floats. The return value is clamped between -1...1 This has the same use-case as [+\(Control, Control\)](#), except this is useful for when adding certain [Xbox](#) Controls like [XTrigger](#) and [XStick](#), which do not use the [Value](#) property of [Control](#).

Protected Member Functions

- abstract void [UpdateStates](#) ()
Internally used for updating the Up/Held/Down states of a control.
- virtual void [UpdateValues](#) ()
Internally used for updating a control's values using it's current states.
- void [SoftReset](#) ()
Internally used for maintaining the [RealValue](#) inbetween updates while resetting everything else.

Properties

- string [Name](#) [get, set]
- bool [IsBlocked](#) [get, set]
- bool [IsDebugControl](#) [get, set]
- float [Gravity](#) [get, set]
- float [Sensitivity](#) [get, set]
- float [Dead](#) [get, set]
- bool [Snap](#) [get, set]
- bool [Invert](#) [get, set]
- sbyte [FixedValue](#) [get, protected set]
- float [Value](#) [get, protected set]
- float [RealValue](#) [get, set]
- [ControlState Down](#) [get, set]
- [ControlState Up](#) [get, set]
- [ControlState Held](#) [get, set]

5.1.1 Detailed Description

Provided to give a common base class and common functionality for [KeyControl](#) and [XboxControl](#).

Controls should be instantiated as parameters in a custom [MonoBehaviour](#) using block initialization. See example below.

Declaration Example

```
KeyControl use = new KeyControl(KeyCode.E) {
    Name = "Jump",
    Gravity = 2f,
    Sensitivity = 2f
};
```

5.1.2 Member Function Documentation

5.1.2.1 static float BSGTools.IO.Control.operator+ (Control *one*, Control *two*) [static]

Adds the [Value](#) properties of a variable amount of controls. The return value is clamped between -1...1. If you have multiple control options for a single in-game action, this is how you gather the final input value for the action.

Parameters

<i>one</i>	The first control.
<i>two</i>	The second control.

Returns

The clamped sum of the control values.

5.1.2.2 static float BSGTools.IO.Control.operator+ (Control *one*, float *val*) [static]

Adds the [Value](#) properties of a variable amount of controls and floats. The return value is clamped between -1...1. This has the same use-case as +(Control, Control), except this is useful for when adding certain [Xbox](#) Controls like XTrigger and XStick, which do not use the Value property of [Control](#).

Parameters

<i>one</i>	The first control.
<i>val</i>	A float value.

Returns

The clamped sum of the control value and the provided float argument.

5.1.2.3 void BSGTools.IO.Control.Reset ()

Reset all non-configuration values and states for this control.

See also

[Reset\(bool\)](#)

5.1.2.4 void BSGTools.IO.Control.Reset (bool *block*)

Reset all non-configuration values for this control. This is the best method to use for cutscenes.

Parameters

<i>block</i>	Whether or not to block this input after resetting.
--------------	---

See also

[Reset](#), [IsBlocked](#)

5.1.2.5 void BSGTools.IO.Control.SoftReset () [protected]

Internally used for maintaining the [RealValue](#) inbetween updates while resetting everything else.

5.1.2.6 override string BSGTools.IO.Control.ToString ()

Returns debug information in a single line.

Returns

The debug information.

5.1.2.7 virtual string BSGTools.IO.Control.ToStringBlock () [virtual]

Returns debug information as a formatted string block.

Returns

The debug information.

Reimplemented in [BSGTools.IO.KeyControl](#).

5.1.2.8 void BSGTools.IO.Control.Update ()

Updates the control. This should never be used by any user-made script. This is public specifically for the use of [InputMaster](#).

5.1.2.9 abstract void BSGTools.IO.Control.UpdateStates () [protected],[pure virtual]

Internally used for updating the Up/Held/Down states of a control.

Implemented in [BSGTools.IO.KeyControl](#), [BSGTools.IO.Xbox.XTriggerControl](#), [BSGTools.IO.Xbox.XStickControl](#), and [BSGTools.IO.Xbox.XButtonControl](#).

5.1.2.10 virtual void BSGTools.IO.Control.UpdateValues () [protected],[virtual]

Internally used for updating a control's values using it's current states.

Reimplemented in [BSGTools.IO.Xbox.XTriggerControl](#), and [BSGTools.IO.Xbox.XStickControl](#).

5.1.3 Property Documentation

5.1.3.1 float BSGTools.IO.Control.Dead [get],[set]

Functionally identical to the Dead property of Unity's native Input system. The absolute value of a control's real value reports as 0 if it's less than this value.

5.1.3.2 ControlState BSGTools.IO.Control.Down [get],[set]

The current "down" state of the control.

5.1.3.3 sbyte BSGTools.IO.Control.FixedValue [get],[protected set]

Returns a digital, ceiling-rounded representation of [Value](#). This is functionally identical to calling `Input.GetAxisRaw()` from Unity's native Input system.

5.1.3.4 float BSGTools.IO.Control.Gravity [get],[set]

Functionally identical to the Gravity property of Unity's native Input system. Speed per second that a control at rest returns to 0.

5.1.3.5 ControlState BSGTools.IO.Control.Held [get],[set]

The current "held" state of the control.

5.1.3.6 bool BSGTools.IO.Control.Invert [get],[set]

Functionally identical to the Invert property of Unity's native Input system. If true, the control's value will report as `-(value)`. However, the state of the control will remain the same (positive down will still report as positive down, etc). Keep in mind that this functions whether or not a negative binding is supplied.

5.1.3.7 `bool BSGTools.IO.Control.IsBlocked` `[get], [set]`

This is used to block any control from receiving updates. Keep in mind that if you block a control, it maintains its values from its most recent update. If you want to block and reset a control, you can use the [Reset\(bool\)](#) function.

5.1.3.8 `bool BSGTools.IO.Control.IsDebugControl` `[get], [set]`

Used to specify controls that automatically only work in the Editor or in Debug builds.

5.1.3.9 `string BSGTools.IO.Control.Name` `[get], [set]`

Can be used as a display name or for string metadata.

5.1.3.10 `float BSGTools.IO.Control.RealValue` `[get], [set], [protected]`

An internally used property that keeps track of the "real value" across updates. This is necessary so that properties like [Dead](#) can be applied to the final value. Think of this as the "real value" and the [Value](#) property as this value after post processing. This value is not necessary to use for input.

5.1.3.11 `float BSGTools.IO.Control.Sensitivity` `[get], [set]`

Functionally identical to the Sensitivity property of Unity's native Input system. Speed per second that a control in motion approaches 1.

5.1.3.12 `bool BSGTools.IO.Control.Snap` `[get], [set]`

Functionally identical to the Snap property of Unity's native Input system. If true, and if the control has a positive and negative binding, the control's value will snap to 0 if provided an opposite input.

5.1.3.13 `ControlState BSGTools.IO.Control.Up` `[get], [set]`

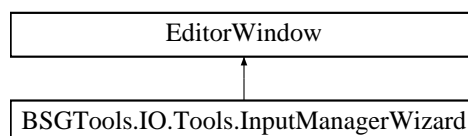
The current "up" state of the control.

5.1.3.14 `float BSGTools.IO.Control.Value` `[get], [protected set]`

Returns an analog representation of the current real value. This is functionally identical to calling `Input.GetAxis()` from Unity's native Input system.

5.2 BSGTools.IO.Tools.InputManagerWizard Class Reference

Inheritance diagram for BSGTools.IO.Tools.InputManagerWizard:



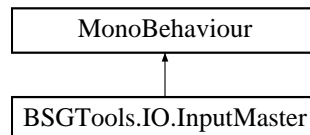
Static Public Member Functions

- static void **ShowWizard** ()

5.3 BSGTools.IO.InputMaster Class Reference

A single instance of this exists in the application. Updates and maintains all [Control](#) states.

Inheritance diagram for BSGTools.IO.InputMaster:



Public Member Functions

- void [ResetAll](#) ()
Resets all states/values on all controls.
- void [SetBlockAll](#) (bool blocked)
Blocks or unblocks all controls. This has the side effect of resetting all control states.
- void [DestroyMaster](#) ()
Destroys the [InputMaster](#) object.
- void [UpdateControls](#) (params [Control](#)[] controls)
Searches the control list for the provided [Control](#) objects and replaces them with said provided object

Static Public Member Functions

- static [InputMaster CreateMaster](#) (object controlClass)
Uses reflection to get all controls in a class. Depending on the control count from your controlClass, this could have a noticable performance spike unless used during loading.
- static [InputMaster CreateMaster](#) (params [Control](#)[] controls)
Creates a new, empty, hidden GameObject, adds a new instance of [InputMaster](#) to it, and adds the provided controls to the master's control list.

Properties

- bool [AnyControlDown](#) [get]
- bool [AnyControlHeld](#) [get]
- bool [AnyControlUp](#) [get]
- string [MouseXAxisName](#) [get, set]
- string [MouseYAxisName](#) [get, set]
- float [MouseX](#) [get]
- float [MouseY](#) [get]
- float [MouseXRaw](#) [get]
- float [MouseYRaw](#) [get]

5.3.1 Detailed Description

A single instance of this exists in the application. Updates and maintains all [Control](#) states.

5.3.2 Member Function Documentation

5.3.2.1 static InputMaster BSGTools.IO.InputMaster.CreateMaster (object *controlClass*) [static]

Uses reflection to get all controls in a class. Depending on the control count from your *controlClass*, this could have a noticeable performance spike unless used during loading.

Parameters

<i>controlClass</i>	The instance of a class to get the controls from.
---------------------	---

Returns

The new [InputMaster](#) instance.

5.3.2.2 static InputMaster BSGTools.IO.InputMaster.CreateMaster (params Control[] *controls*) [static]

Creates a new, empty, hidden GameObject, adds a new instance of [InputMaster](#) to it, and adds the provided controls to the master's control list.

Parameters

<i>controls</i>	A full listing of all of the games controls with the default bindings.
-----------------	--

Returns

The new [InputMaster](#) instance.

5.3.2.3 void BSGTools.IO.InputMaster.DestroyMaster ()

Destroys the [InputMaster](#) object.

5.3.2.4 void BSGTools.IO.InputMaster.ResetAll ()

Resets all states/values on all controls.

See also

[ResetAll\(bool\)](#)

5.3.2.5 void BSGTools.IO.InputMaster.SetBlockAll (bool *blocked*)

Blocks or unblocks all controls. This has the side effect of resetting all control states.

Parameters

<i>blocked</i>	To block/unblock.
----------------	-------------------

See also

[Control.IsBlocked](#)

5.3.2.6 void BSGTools.IO.InputMaster.UpdateControls (params Control[] *controls*)

Searches the control list for the provided [Control](#) objects and replaces them with said provided object

Parameters

<i>controls</i>	The Control objects to search and replace
-----------------	---

5.3.3 Property Documentation

5.3.3.1 `bool BSGTools.IO.InputMaster.AnyControlDown` `[get]`

Are any controls in an active Down state?

5.3.3.2 `bool BSGTools.IO.InputMaster.AnyControlHeld` `[get]`

Are any controls in an active Held state?

5.3.3.3 `bool BSGTools.IO.InputMaster.AnyControlUp` `[get]`

Are any controls in an active Up state?

5.3.3.4 `float BSGTools.IO.InputMaster.MouseX` `[get]`

The Mouse X Axis axis value from Unity's native Input system.

5.3.3.5 `string BSGTools.IO.InputMaster.MouseXAxisName` `[get]`, `[set]`

The Mouse X Axis axis name in Unity's Input Manager

5.3.3.6 `float BSGTools.IO.InputMaster.MouseXRaw` `[get]`

The Mouse X Axis raw axis value from Unity's native Input system.

5.3.3.7 `float BSGTools.IO.InputMaster.MouseY` `[get]`

The Mouse Y Axis axis value from Unity's native Input system.

5.3.3.8 `string BSGTools.IO.InputMaster.MouseYAxisName` `[get]`, `[set]`

The Mouse Y Axis axis name in Unity's Input Manager

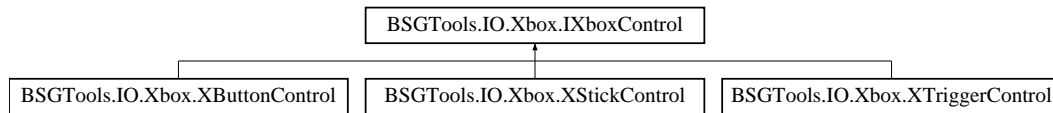
5.3.3.9 `float BSGTools.IO.InputMaster.MouseYRaw` `[get]`

The Mouse Y Axis raw axis value from Unity's native Input system.

5.4 BSGTools.IO.Xbox.IXboxControl Interface Reference

Used for type constraints. Contains nor enforces any functionality.

Inheritance diagram for BSGTools.IO.Xbox.IXboxControl:



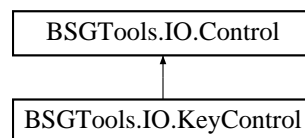
5.4.1 Detailed Description

Used for type constraints. Contains nor enforces any functionality.

5.5 BSGTools.IO.KeyControl Class Reference

Used for all Keyboard and Mouse Button controls. Any binding present in Unity's KeyCode enumeration is valid here.

Inheritance diagram for BSGTools.IO.KeyControl:



Public Member Functions

- [KeyControl](#) (KeyCode positive)
Creates a new [KeyControl](#). This is the "new version" of OneWayControl from previous versions of [InputMaster](#).
- [KeyControl](#) (KeyCode positive, KeyCode negative)
Creates a new [KeyControl](#) with a negative binding. This is the "new version" of AxisControl from previous versions of [InputMaster](#).
- override string [ToStringBlock](#) ()
Returns debug information as a formatted string block.

Protected Member Functions

- override void [UpdateStates](#) ()
Updates the current states of this control.

Properties

- KeyCode [Positive](#) [get, set]
- KeyCode [Negative](#) [get, set]
- ModifierKey [Modifier](#) [get, set]

Additional Inherited Members

5.5.1 Detailed Description

Used for all Keyboard and Mouse Button controls. Any binding present in Unity's KeyCode enumeration is valid here.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 BSGTools.IO.KeyControl.KeyControl (KeyCode *positive*)

Creates a new [KeyControl](#). This is the "new version" of OneWayControl from previous versions of [InputMaster](#).

Parameters

<i>positive</i>	Positive
-----------------	--------------------------

5.5.2.2 BSGTools.IO.KeyControl.KeyControl (KeyCode *positive*, KeyCode *negative*)

Creates a new [KeyControl](#) with a negative binding. This is the "new version" of AxisControl from previous versions of [InputMaster](#).

Parameters

<i>positive</i>	Positive
<i>negative</i>	Negative

5.5.3 Member Function Documentation

5.5.3.1 override string BSGTools.IO.KeyControl.ToStringBlock () [virtual]

Returns debug information as a formatted string block.

Returns

The debug information.

Reimplemented from [BSGTools.IO.Control](#).

5.5.3.2 override void BSGTools.IO.KeyControl.UpdateStates () [protected],[virtual]

Updates the current states of this control.

Implements [BSGTools.IO.Control](#).

5.5.4 Property Documentation

5.5.4.1 ModifierKey BSGTools.IO.KeyControl.Modifier [get],[set]

The OPTIONAL modifier key for this control.

5.5.4.2 KeyCode BSGTools.IO.KeyControl.Negative [get],[set]

The OPTIONAL negative binding for this control.

5.5.4.3 KeyCode BSGTools.IO.KeyControl.Positive [get],[set]

The REQUIRED positive binding for this control. CANNOT BE KeyCode.None!

5.6 BSGTools.IO.ModifierKey Class Reference

Represents a Modifier for a control.

Public Types

- enum [ModEnums](#) {
None, LShift, LCtrl, LAlt,
LWindows, LCommand, RShift, RCtrl,
RAlt, RWindows, RCommand }

Used with [FromMEnum](#) for listing options.

Static Public Member Functions

- static implicit [operator KeyCode](#) ([ModifierKey](#) modifier)
Assists in checking key status.
- static implicit [operator ModifierKey](#) ([ModEnums](#) modifier)
Utility operator overload for listing purposes.
- static [ModifierKey FromMEnum](#) ([ModEnums](#) me)
Utility method for listing purposes.

Static Public Attributes

- static readonly [ModifierKey None](#) = new [ModifierKey](#)(KeyCode.None, "None")
Default Modifier for KeyControls with no modifier.
- static readonly [ModifierKey LShift](#) = new [ModifierKey](#)(KeyCode.LeftShift, "Left Shift")
The Left Shift modifier key.
- static readonly [ModifierKey LCtrl](#) = new [ModifierKey](#)(KeyCode.LeftControl, "Left [Control](#)")
The Left [Control](#) modifier key.
- static readonly [ModifierKey LAlt](#) = new [ModifierKey](#)(KeyCode.LeftAlt, "Left Alt")
The Left Alt modifier key.
- static readonly [ModifierKey LWindows](#) = new [ModifierKey](#)(KeyCode.LeftWindows, "Left Windows")
The Left Windows modifier key.
- static readonly [ModifierKey LCommand](#) = new [ModifierKey](#)(KeyCode.LeftCommand, "Left Command")
The Left Command modifier key on Apple based keyboards.
- static readonly [ModifierKey RShift](#) = new [ModifierKey](#)(KeyCode.RightShift, "Right Shift")
The Right Shift modifier key.
- static readonly [ModifierKey RCtrl](#) = new [ModifierKey](#)(KeyCode.RightControl, "Right [Control](#)")
The Right [Control](#) modifier key.
- static readonly [ModifierKey RAlt](#) = new [ModifierKey](#)(KeyCode.RightAlt, "Right Alt")
The Right Alt modifier key.
- static readonly [ModifierKey RWindows](#) = new [ModifierKey](#)(KeyCode.RightWindows, "Right Windows")
The Right Windows modifier key.
- static readonly [ModifierKey RCommand](#) = new [ModifierKey](#)(KeyCode.RightCommand, "Right Command")
The Right Command modifier key on Apple based keyboards.
- static readonly [ModifierKey\[\] modifiers](#)
Can be used for listing.

Properties

- `KeyCode` [UKeyCode](#) [get]
Unity's keycode for this Modifier.
- `string` [DisplayName](#) [get]
Full name of this modifier key.

5.6.1 Detailed Description

Represents a Modifier for a control.

5.6.2 Member Enumeration Documentation

5.6.2.1 `enum` `BSGTools.IO.ModifierKey.ModEnums`

Used with [FromMEnum](#) for listing options.

5.6.3 Member Function Documentation

5.6.3.1 `static ModifierKey` `BSGTools.IO.ModifierKey.FromMEnum (ModEnums me)` [static]

Utility method for listing purposes.

Parameters

<i>me</i>	The ModEnums to convert.
-----------	--

Returns

The proper static modifier.

5.6.3.2 `static implicit` `BSGTools.IO.ModifierKey.operator KeyCode (ModifierKey modifier)` [static]

Assists in checking key status.

5.6.3.3 `static implicit` `BSGTools.IO.ModifierKey.operator ModifierKey (ModEnums modifier)` [static]

Utility operator overload for listing purposes.

5.6.4 Member Data Documentation

5.6.4.1 `readonly ModifierKey` `BSGTools.IO.ModifierKey.LAlt = new ModifierKey(KeyCode.LeftAlt, "Left Alt")` [static]

The Left Alt modifier key.

5.6.4.2 `readonly ModifierKey` `BSGTools.IO.ModifierKey.LCommand = new ModifierKey(KeyCode.LeftCommand, "Left Command")` [static]

The Left Command modifier key on Apple based keyboards.

5.6.4.3 readonly **ModifierKey** BSGTools.IO.ModifierKey.LCtrl = new **ModifierKey**(KeyCode.LeftControl, "Left Control")
[static]

The Left **Control** modifier key.

5.6.4.4 readonly **ModifierKey** BSGTools.IO.ModifierKey.LShift = new **ModifierKey**(KeyCode.LeftShift, "Left Shift")
[static]

The Left Shift modifier key.

5.6.4.5 readonly **ModifierKey** BSGTools.IO.ModifierKey.LWindows = new **ModifierKey**(KeyCode.LeftWindows, "Left Windows") [static]

The Left Windows modifier key.

5.6.4.6 readonly **ModifierKey** [] BSGTools.IO.ModifierKey.modifiers [static]

Initial value:

```
= new ModifierKey[]{  
    None,  
    LShift,  
    LCtrl,  
    LAlt,  
    LWindows,  
    LCommand,  
    RShift,  
    RCtrl,  
    RAlt,  
    RWindows,  
    RCommand,  
}
```

Can be used for listing.

5.6.4.7 readonly **ModifierKey** BSGTools.IO.ModifierKey.None = new **ModifierKey**(KeyCode.None, "None") [static]

Default Modifier for KeyControls with no modifier.

5.6.4.8 readonly **ModifierKey** BSGTools.IO.ModifierKey.RAlt = new **ModifierKey**(KeyCode.RightAlt, "Right Alt")
[static]

The Right Alt modifier key.

5.6.4.9 readonly **ModifierKey** BSGTools.IO.ModifierKey.RCommand = new **ModifierKey**(KeyCode.RightCommand, "Right Command") [static]

The Right Command modifier key on Apple based keyboards.

5.6.4.10 readonly **ModifierKey** BSGTools.IO.ModifierKey.RCtrl = new **ModifierKey**(KeyCode.RightControl, "Right Control")
[static]

The Right **Control** modifier key.

5.6.4.11 readonly **ModifierKey** BSGTools.IO.ModifierKey.RShift = new ModifierKey(KeyCode.RightShift, "Right Shift")
[static]

The Right Shift modifier key.

5.6.4.12 readonly **ModifierKey** BSGTools.IO.ModifierKey.RWindows = new ModifierKey(KeyCode.RightWindows, "Right Windows") [static]

The Right Windows modifier key.

5.6.5 Property Documentation

5.6.5.1 string BSGTools.IO.ModifierKey.DisplayName [get]

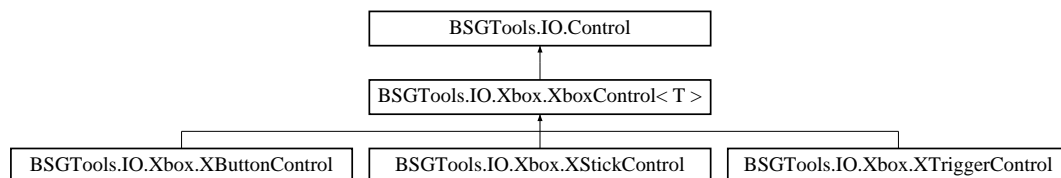
Full name of this modifier key.

5.6.5.2 KeyCode BSGTools.IO.ModifierKey.UKeyCode [get]

Unity's keycode for this Modifier.

5.7 BSGTools.IO.Xbox.XboxControl< T > Class Template Reference

Inheritance diagram for BSGTools.IO.Xbox.XboxControl< T >:



Static Public Member Functions

- static T[] [CreateMultiple](#) (byte count, T toClone)

Allows for the creation of up to 4 XboxControls at the same time (one for each controller index). An XboxControl must be provided for cloning. The [ControllerIndex](#) of the provided clone is ignored.

Protected Member Functions

- [XboxControl](#) (byte controllerIndex)
Creates a new XboxControl .
- abstract T [CreateClone](#) (byte controller)
Creates a clone of an this XboxControl.

Properties

- byte [ControllerIndex](#) [get]

Additional Inherited Members

5.7.1 Detailed Description

The base class for all [Xbox](#) 360 control classes.

Template Parameters

<i>T</i>	Used for generic self-creation.
----------	---------------------------------

Type Constraints

T: [IXboxControl](#)

5.7.2 Constructor & Destructor Documentation

5.7.2.1 BSGTools.IO.Xbox.XboxControl< T >.XboxControl (byte *controllerIndex*) [protected]

Creates a new *XboxControl* .

Parameters

<i>controllerIndex</i>	ControllerIndex
------------------------	---------------------------------

5.7.3 Member Function Documentation

5.7.3.1 abstract T BSGTools.IO.Xbox.XboxControl< T >.CreateClone (byte *controller*) [protected], [pure virtual]

Creates a clone of an this XboxControl.

Parameters

<i>controller</i>	The index of the controller that will manipulate this control's states and values.
-------------------	--

Returns

Implemented in [BSGTools.IO.Xbox.XTriggerControl](#), [BSGTools.IO.Xbox.XStickControl](#), and [BSGTools.IO.Xbox.X↵ButtonControl](#).

5.7.3.2 static T [] BSGTools.IO.Xbox.XboxControl< T >.CreateMultiple (byte *count*, T *toClone*) [static]

Allows for the creation of up to 4 XboxControls at the same time (one for each controller index). An *XboxControl* must be provided for cloning. The [ControllerIndex](#) of the provided clone is ignored.

Parameters

<i>count</i>	The number of controllers to create this control for.
<i>toClone</i>	The instance to take values from for clone creation.

Returns

An array of type T, each assigned to a specific controller index.

Example

```
// Defines a jump control for all 4 players.
XButtonControl[] jump = XButtonControl.CreateMultiple(4, new XButtonControl(0, XButton.A) {
    Name = "Jump",
    Gravity = 2f,
    Sensitivity = 2f
});
```



```
// Usage
jump[0].Value; //Player One's jump value
jump[3].Value; //Player Four's jump value
```

5.7.4 Property Documentation

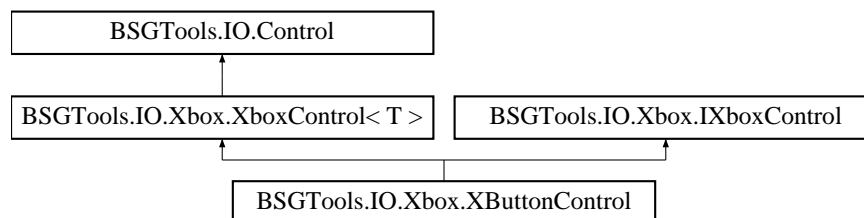
5.7.4.1 byte BSGTools.IO.Xbox.XboxControl< T >.ControllerIndex [get]

The index of the controller that will manipulate this control's states and values. This is used in conjunction to the PlayerIndex enumeration in XInput.

5.8 BSGTools.IO.Xbox.XButtonControl Class Reference

Used for integrating [Xbox](#) 360 controller digital buttons/inputs into [InputMaster's Control](#) system.

Inheritance diagram for BSGTools.IO.Xbox.XButtonControl:



Public Member Functions

- [XButtonControl](#) ([XButton](#) positive)
Creates an [XButtonControl](#) for a single player game.
- [XButtonControl](#) ([XButton](#) positive, [XButton](#) negative)
Creates an [XButtonControl](#) for a single player game.

Protected Member Functions

- override void [UpdateStates](#) ()
Updates this control's states.
- override [XButtonControl CreateClone](#) (byte controller)
Creates a clone of this [XButtonControl](#).

Properties

- [XButton Positive](#) [get, set]
- [XButton Negative](#) [get, set]

Additional Inherited Members

5.8.1 Detailed Description

Used for integrating [Xbox](#) 360 controller digital buttons/inputs into [InputMaster's Control](#) system.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 BSGTools.IO.Xbox.XButtonControl.XButtonControl (*XButton positive*)

Creates an [XButtonControl](#) for a single player game.

Parameters

<i>positive</i>	The positive binding. CANNOT BE XButton.None!
-----------------	---

5.8.2.2 BSGTools.IO.Xbox.XButtonControl.XButtonControl (*XButton positive, XButton negative*)

Creates an [XButtonControl](#) for a single player game.

Parameters

<i>positive</i>	The positive binding. CANNOT BE XButton.None!
<i>negative</i>	The negative binding.

5.8.3 Member Function Documentation

5.8.3.1 override [XButtonControl](#) BSGTools.IO.Xbox.XButtonControl.CreateClone (*byte controller*) [protected], [virtual]

Creates a clone of this [XButtonControl](#).

Parameters

<i>controller</i>	The index of the controller that will manipulate this control's states and values.
-------------------	--

Returns

The cloned control.

Implements [BSGTools.IO.Xbox.XboxControl< T >](#).

5.8.3.2 override void BSGTools.IO.Xbox.XButtonControl.UpdateStates () [protected], [virtual]

Updates this control's states.

Implements [BSGTools.IO.Control](#).

5.8.4 Property Documentation

5.8.4.1 [XButton](#) BSGTools.IO.Xbox.XButtonControl.Negative [get], [set]

The negative binding for this control.

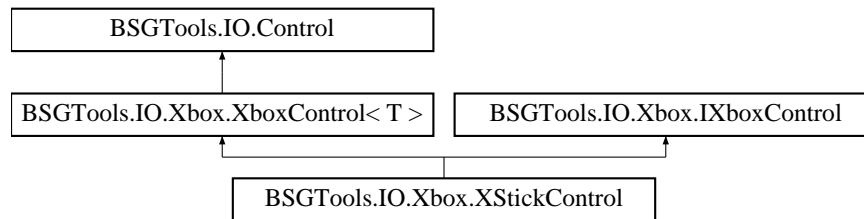
5.8.4.2 [XButton](#) BSGTools.IO.Xbox.XButtonControl.Positive [get], [set]

The positive binding for this control. CANNOT BE XBinding.None!

5.9 BSGTools.IO.Xbox.XStickControl Class Reference

Used for integrating [Xbox](#) 360 controller analog sticks into [InputMaster's Control](#) system.

Inheritance diagram for BSGTools.IO.Xbox.XStickControl:



Public Types

- enum [InvertMode](#) { **X** = 1 >> 0, **Y** = 1 >> 1 }

Because there are 2 axes to worry about, a specialized enumeration is used to allow for different inversion modes.

Public Member Functions

- [XStickControl](#) ([XStick](#) stick)

Creates an [XStickControl](#) for a single player game.

Protected Member Functions

- override void [UpdateStates](#) ()

States are not used for stick controls.

- override void [UpdateValues](#) ()

Updates this control's values. The [StickValue](#) property is updated in [UpdateStates](#).

- override [XStickControl](#) [CreateClone](#) (byte controller)

Creates a clone of this [XStickControl](#).

Properties

- [XStick](#) [Stick](#) [get, set]
- [Vector2](#) [StickValue](#) [get]
- [InvertMode](#) [InversionMode](#) [get, set]

Additional Inherited Members

5.9.1 Detailed Description

Used for integrating [Xbox](#) 360 controller analog sticks into [InputMaster](#)'s [Control](#) system.

5.9.2 Member Enumeration Documentation

5.9.2.1 enum BSGTools.IO.Xbox.XStickControl.InvertMode

Because there are 2 axes to worry about, a specialized enumeration is used to allow for different inversion modes.

5.9.3 Constructor & Destructor Documentation

5.9.3.1 BSGTools.IO.Xbox.XStickControl.XStickControl (XStick *stick*)

Creates an [XStickControl](#) for a single player game.

Parameters

<i>stick</i>	The bound stick.
--------------	------------------

5.9.4 Member Function Documentation

5.9.4.1 `override XStickControl BSGTools.IO.Xbox.XStickControl.CreateClone (byte controller)` [protected], [virtual]

Creates a clone of this [XStickControl](#).

Parameters

<i>controller</i>	The ControllerIndex to assign to the new clone.
-------------------	---

Returns

The cloned [XStickControl](#).

Implements [BSGTools.IO.Xbox.XboxControl< T >](#).

5.9.4.2 `override void BSGTools.IO.Xbox.XStickControl.UpdateStates ()` [protected], [virtual]

States are not used for stick controls.

Implements [BSGTools.IO.Control](#).

5.9.4.3 `override void BSGTools.IO.Xbox.XStickControl.UpdateValues ()` [protected], [virtual]

Updates this control's values. The [StickValue](#) property is updated in [UpdateStates](#).

Reimplemented from [BSGTools.IO.Control](#).

5.9.5 Property Documentation

5.9.5.1 `InvertMode BSGTools.IO.Xbox.XStickControl.InversionMode` [get], [set]

How inversion should be applied to this control.

5.9.5.2 `XStick BSGTools.IO.Xbox.XStickControl.Stick` [get], [set]

The assigned analog stick.

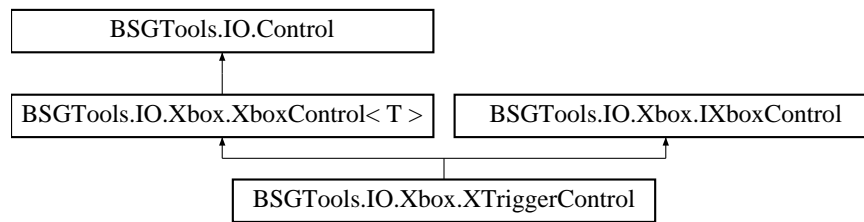
5.9.5.3 `Vector2 BSGTools.IO.Xbox.XStickControl.StickValue` [get]

The X/Y axis values for this control.

5.10 BSGTools.IO.Xbox.XTriggerControl Class Reference

Used for integrating [Xbox](#) 360 controller triggers into [InputMaster's Control](#) system.

Inheritance diagram for [BSGTools.IO.Xbox.XTriggerControl](#):



Public Member Functions

- [XTriggerControl](#) ([XTrigger](#) trigger)
Creates an [XTriggerControl](#) for a single player game.

Protected Member Functions

- override void [UpdateStates](#) ()
In this specialized case, the values are updated here, not the states.
- override void [UpdateValues](#) ()
- override [XTriggerControl](#) [CreateClone](#) (byte controller)
Creates a clone of this [XTriggerControl](#).

Properties

- [XTrigger](#) [Trigger](#) [get, set]

Additional Inherited Members

5.10.1 Detailed Description

Used for integrating [Xbox](#) 360 controller triggers into [InputMaster](#)'s [Control](#) system.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 BSGTools.IO.Xbox.XTriggerControl.XTriggerControl ([XTrigger](#) trigger)

Creates an [XTriggerControl](#) for a single player game.

Parameters

<i>trigger</i>	The bound trigger.
----------------	--------------------

5.10.3 Member Function Documentation

5.10.3.1 override [XTriggerControl](#) BSGTools.IO.Xbox.XTriggerControl.CreateClone (byte controller) [protected], [virtual]

Creates a clone of this [XTriggerControl](#).

Parameters

<i>controller</i>	The ControllerIndex to assign to the new clone.
-------------------	---

Returns

The cloned [XTriggerControl](#).

Implements [BSGTools.IO.Xbox.XboxControl< T >](#).

5.10.3.2 `override void BSGTools.IO.Xbox.XTriggerControl.UpdateStates ()` `[protected]`, `[virtual]`

In this specialized case, the values are updated here, not the states.

Implements [BSGTools.IO.Control](#).

5.10.3.3 `override void BSGTools.IO.Xbox.XTriggerControl.UpdateValues ()` `[protected]`, `[virtual]`

[UpdateStates](#)

Reimplemented from [BSGTools.IO.Control](#).

5.10.4 Property Documentation

5.10.4.1 `XTrigger BSGTools.IO.Xbox.XTriggerControl.Trigger` `[get]`, `[set]`

The assigned trigger that will manipulate this control's states and values.