InputMaster

Generated by Doxygen 1.8.8

Wed Sep 17 2014 17:14:47

Contents

Chapter 1

Namespace Index

1.1 Packages

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

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

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.

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.

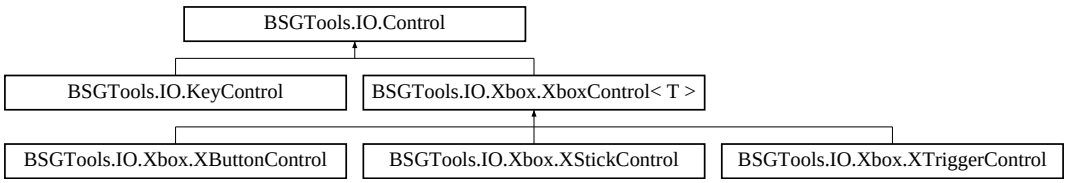
Chapter 5

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

one	The first control.
two	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

one	The first control.
val	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

block	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 contol'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 it's 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]

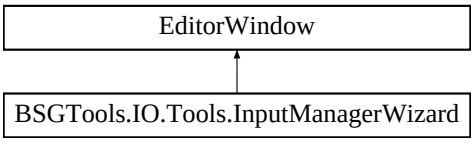
The current "up" state of the control.

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:



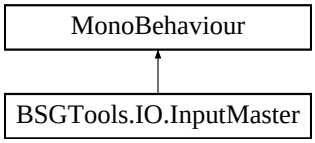
Static Public Member Functions

- 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 ResetAll (bool block)

  Resets all states/values on all Controls. This is useful for when you want to show a cutscene.
- void SetBlockAll (bool blocked)

  Blocks or unblocks all controls.
- 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 AnyKeyDown `[get]`

  Not yet implemented.
- bool AnyKeyHeld `[get]`

  Not yet implemented.
- bool AnyKeyUp `[get]`

  Not yet implemented.

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 noticable performance spike unless used during loading.

Parameters

controlClass	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

controls	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.ResetAll (bool *block*)

Resets all states/values on all Controls. This is useful for when you want to show a cutscene.

Parameters

block	Whether or not to set each control to be blocked.

See also

ResetAll, Control.IsBlocked

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

Blocks or unblocks all controls.

Parameters

blocked	To block/unblock.

See also

Control.IsBlocked

5.3.2.7 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

controls	The Control objects to search and replace

5.3.3 Property Documentation

5.3.3.1 bool BSGTools.IO.InputMaster.AnyKeyDown [get]

Not yet implemented.

5.3.3.2 bool BSGTools.IO.InputMaster.AnyKeyHeld [get]

Not yet implemented.

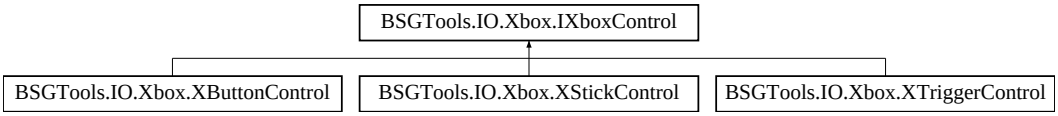
5.3.3.3 bool BSGTools.IO.InputMaster.AnyKeyUp [get]

Not yet implemented.

5.4 BSGTools.IO.Xbox.IXboxControl Interface Reference

Used for type constraints. Contains nor enforces any functionality.

Inheritance diagram for BSGTools.IO.Xbox.IXboxControl:



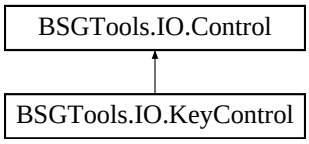
5.4.1 Detailed Description

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]`

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

positive	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

positive	Positive
negative	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 KeyCode BSGTools.IO.KeyControl.Negative [get], [set]

The negative binding for this control. Optional.

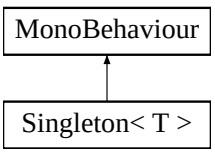
5.5.4.2 KeyCode BSGTools.IO.KeyControl.Positive [get], [set]

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

5.6 Singleton< T > Class Template Reference

Be aware this will not prevent a non singleton constructor such as `T myT = new T();` To prevent that, add `protected T () {}` to your singleton class. As a note, this is made as MonoBehaviour for Coroutines.

Inheritance diagram for Singleton< T >:



Public Member Functions

- void DummyCreate ()

  Strictly for testing Singleton creation.
- void OnDestroy ()

  When Unity quits, it destroys objects in a random order. In principle, a Singleton is only destroyed when application quits. If any script calls Instance after it have been destroyed, it will create a buggy ghost object that will stay on the Editor scene even after stopping playing the Application. So, this is to be sure that a ghost object is not created.

Properties

- static T Instance [get]

5.6.1 Detailed Description

Be aware this will not prevent a non singleton constructor such as `T myT = new T();` To prevent that, add `protected T () {}` to your singleton class. As a note, this is made as MonoBehaviour for Coroutines.

Type Constraints

T* : *MonoBehaviour

5.6.2 Member Function Documentation

5.6.2.1 void Singleton< T >.DummyCreate ( )

Strictly for testing Singleton creation.

5.6.2.2 void Singleton< T >.OnDestroy ( )

When Unity quits, it destroys objects in a random order. In principle, a Singleton is only destroyed when application quits. If any script calls Instance after it have been destroyed, it will create a buggy ghost object that will stay on the Editor scene even after stopping playing the Application. So, this is to be sure that a ghost object is not created.

From: http://wiki.unity3d.com/index.php/Singleton

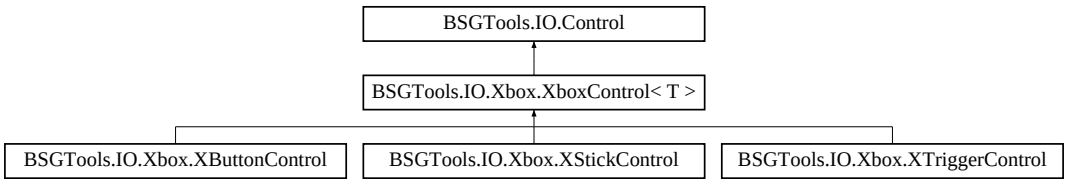
5.6.3 Property Documentation

5.6.3.1 T Singleton< T >.Instance [static], [get]

Does singleton checking, then returns the instance of this class.

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

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



Public Member Functions

- XboxControl (byte controllerIndex)

  Creates a new XboxControl .

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

- abstract T CreateClone (byte controller)

  Creates a clone of an this XboxControl.

Properties

- byte ControllerIndex `[get, set]`

5.7.1 Detailed Description

The base class for all Xbox 360 control classes.

Template Parameters

T	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*)

Creates a new *XboxControl* .

Parameters

controllerIndex	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

controller	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

count	The number of controllers to create this control for.
toClone	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

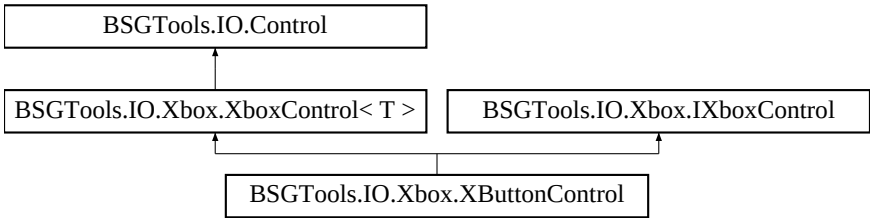
5.7.4.1 byte BSGTools.IO.Xbox.XboxControl< T >.ControllerIndex [get],[set]

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 (byte controllerIndex, XButton positive)

  Creates a new XButtonControl.
- XButtonControl (byte controllerIndex, XButton positive, XButton negative)

  Creates a new XButtonControl.

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 (byte *controllerIndex*, XButton *positive*)

Creates a new XButtonControl.

Parameters

controllerIndex	ControllerIndex
positive	Positive

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

Creates a new XButtonControl.

Parameters

controllerIndex	The index of the controller that will manipulate this control's states and values.
positive	The positive binding for this control. CANNOT BE XBinding.None!
negative	The negative binding for this control.

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

controller	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.

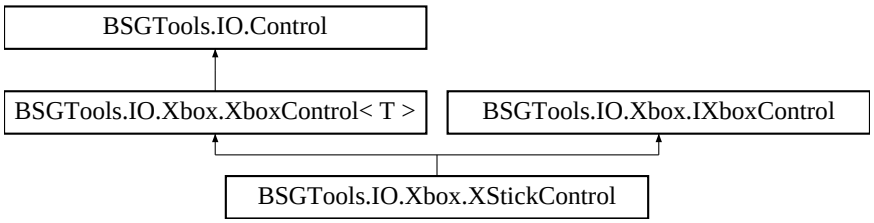
5.8.4.2 XButton BSGTools.IO.Xbox.XButtonControl.Positive [get], [set]

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 (byte controllerIndex, XStick stick)

  Creates a new XStickControl.

Protected Member Functions

- override void UpdateStates ()

  Updates this control's states.
- 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 (byte *controllerIndex*, XStick *stick*)

Creates a new XStickControl.

Parameters

controllerIndex	ControllerIndex
stick	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

controller	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]

Updates this control's states.

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.

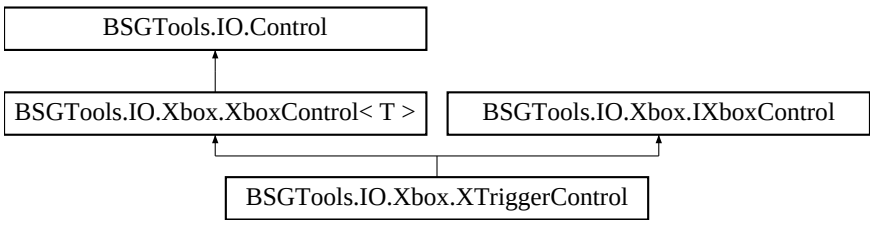
5.9.5.3 Vector2 BSGTools.IO.Xbox.XStickControl.StickValue [get]

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 (byte controllerIndex, XTrigger trigger)
  Creates a new XTriggerControl.

Protected Member Functions

- override void UpdateStates ()
  In this specialized case, the values are updated here, not the states.
- override void UpdateValues ()
  Values are updated through the UpdateStates method.
- 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 (byte *controllerIndex,* XTrigger *trigger*)

Creates a new XTriggerControl.

Parameters

controllerIndex	ControllerIndex
trigger	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

controller	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]

Values are updated through the UpdateStates method.

UpdateStates

Reimplemented from BSGTools.IO.Control.

5.10.4 Property Documentation

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

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

Index