ISH Documentation

Introduction

ISH (Input System Handler) is a library to provide simple input features for games by abstracting device handling and input check inside the library code. The purpose of ISH is to allow users to set custom group of input mapping in a GUI based editor and simply apply input check to them with a single line of code.

Using ISH to Retrieve Input Data

Before we start with the code, we must ensure that ish_setting.ini file is copied to the same directory as where the program is being run (i.e. same directory as the executable file or the project file). This file stores the setting used in input handling so ISH will not feature without it.

To use ISH, first you must make sure you call the following functions inside your program:

```
ISH::Initialize( fileName, numberOfPlayers );
ISH::Update();
ISH::Shutdown();
```

The function name describes what it does. Initialize() and Shutdown functions must be called at the start and the end of the application and Update() function should be called once per frame. The Initialize function take the path to key configuration file to load as the first parameter and applies the setting to all players.

Once you have them placed in your project, you can retrieve input data with the following function:

```
float value = ISH::GetInputValue( "MoveX", playerIndex );
```

First parameter is the "Group name" (or "key name") for the input. This checks for input defined in the configuration file by checking the input sets in [name] section (e.g. [MoveX]).

Second parameter sets which player and xbox controller to check for input. This will be useful as each player is likely to have different controller and each player could use different configuration file.

Returned value is a float rather than a bool because it considers axis input, such as triggers and thumbsticks on xbox controllers. In addition, ISH provides option to return positive or negative value so that one group can be used for wide range of inputs e.g. MoveX contains movement in x-axis by checking thumbstick, Dpad left and right, left and right keys on the keyboard. If no input was found, it simply returns 0.0f (which can be used as false in if statements).

Using ISH to Load or Setup Key Maps at Runtime

To load new setting for a player from a file, use the following function:

```
bool success = ISH::LoadKeyData( fileName, playerToApplySetting );
```

As this function gets called during Initialize() function, it takes in similar parameter. The difference is that second parameter is to specify which player to load new setting to rather than setting number of players. If the function returns true, the new setting should be available to use straight away after the end of function call. Otherwise, it failed to load new setting.

To setup key map at runtime, use the following function:

```
std::string input = SetInput( fileName, name, option, index, isPositive, saveOption );
```

This function checks if there are any inputs in current frame and if there is one, it acts differently depending on the saveOption parameter. Under any options, returned string will be input button, key or triggers on 360 controller (it does not support thumbstick movement on 360 pads). If there was no input during the frame this function was called, it returns “”.

The different save options are the following:

- NoSave - do not save to file or to current setting and just return input as a string.
- SaveOnly - save new input to file but do not reload or add new input setting to player
- SaveReload - save new input to file and reload setting with the given file
- SaveAdd - save new input to file and add (or replace) new input to current setting

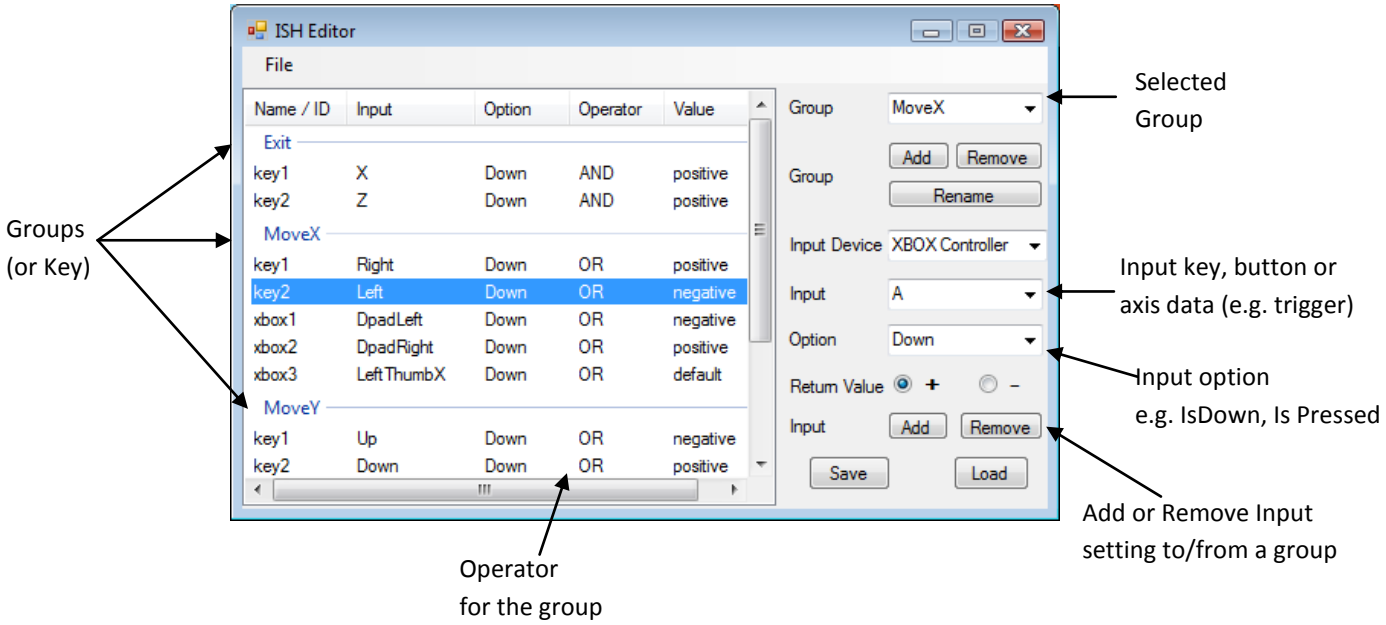
The rest of the parameter is used to setup the input option:

- Option - is to check what type of input it accepts i.e. IsDown, IsUp, IsPressed or IsReleased.
- Index - indicates which player to check for input and apply new setting to.
- isPositive - defines if return value of newly set input is positive or negative.

Currently, `SetInput()` can only store one input setting per group and it does not take in thumbstick input. Therefore, if you want to setup an input group that accepts multiple inputs or axis input form thumbsticks, it is recommended that you use the editor.

Using the Editor

Before running the editor, you must ensure that ish_setting.ini file is copied to the same directory as the editor just like using ISH in an application.



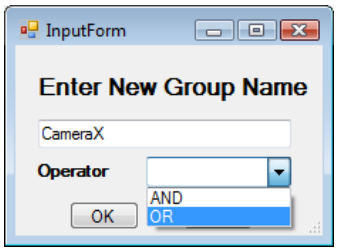
Input Data

Group / Key – This is what you use to check for the input e.g. to check for inputs in [MoveX], call `ISH::GetInputValue( "MoveX" ).`

Input – This is the input key, button or axis info to check for input e.g. [Exit] checks for keyboard input for 'X' and 'Z' keys being used together.

Option – Option on type of input check to apply e.g. [MoveX] checks if keys or buttons are down by using IsDown() function to check and [EmitParticle] checks if keys and buttons were pressed using IsPressed() function.

Operator – This is the operator to use during an input check for a group. A group can only contain single operator e.g. to [Exit] the game, 'X' and 'Z' must be used together and to move in x-axis, you can hold down any keys or button in the group.



Value – Determine whether the returned value is positive or negative. "default" indicates it is an axis input which will directly return its value.

Config File Format

The key config file (or key mapping file) is stored in .ini file format, which is easy to edit with a text editor such as Notepad++. It is recommended that you use the editor to create or edit a file but in case you like to directly edit the file, these are the basic info (the names used in example below is NOT the official naming, these are names I set to explain ISH):

```
[GroupName]

ElementName = value
```

An example config file would look this:

```
[Reload]

key1=R

Option1=Pressed

value1=positive

xbox1=X

Option2=Pressed

value2=positive

operator=OR

[Zap]

key1=Z

Option1=Down

value1=positive

operator=AND
```

key1 and xbox1 represents the input key, button or axis. In ISH, keyboard input take priority so Option and Value starts from 1 to map to keyboard input if there is one. The values on right hand side are defined in ish_setting.ini. This is one of the reasons for having the file in same directory is important. Inside ish_setting.ini file, the values that are used in key maps are defined on right hand side of [XBOX] section and [Keyboard] section. These files are **case sensitive** so if you are manually editing the file, make sure you make no mistake there.