DirectInput provider

DirectInput-supported device

Version 1.0.0

User’s guide

July 17, 2012

[Remarks]

[Revision history]

Version	Date	Content
1.0.0.0	2010-07-22	First edition.
1.0.0.1	2010-03-15	Added sending source information of OnMessage event.
1.0.0.2	2012-05-10	Added @State variable
1.0.0	2012-07-17	Modified the version rule of document management.

[Supported device]

Device	Version	Notes

Contents

1. Introduction

Microsoft DirectInput is application programming interface (API) that supports not only a mouse, keyboard, joy stick, or other game controller-related input devices but also force feedback (input/output) devices.

This document describes DirectInput provider that is a CAO provider to access input devices by using DirectInput.

2. Overview of provider

2.1. Overview

DirectInput provider accesses an input device by using DirectInput. An input device is assigned to a CaoExtension class.

There are two ways to obtain the information of input device, as follows.

- Obtaining a value by using CaoVariable class.
- Status change information of CaoMessage event.

Also, using CaoExtension::Execute() enables to use Forcefeedback of input devices. Effects of Forcefeedback are stored in Effect number 0 to 9.

Table 2-1 Effect number allocation table

Effect number	Contents
0	Effect of default setting. Execute a constant vibration for unlimited time.
1 to 9	Effects that are read from the effect definition file. To read effects, execute LoadEffectFile command of CaoFile::Execute method.

The file format of DirectInput provider is DLL (Dynamic Link Library) and Table 2-2 shows the details.

Table 2-2 DirectInput provider

File name	CaoProvDirectInput.dll
ProgID	CaoProv.DirectInput
Registration[1]	regsvr32 CaoProvDirectInput.dll
Deregistration	regsvr32 /u CaoProvDirectInput.dll

[1] You do not have to register/deregister it manually if the provider is installed by ORiN2 SDK

2.2. Method and Properties

// This section describes implemented methods.

2.2.1. CaoWorkspace::AddController method

At the timing of AddController (when creating an Controller object), DirectInput provider searches an input device. Connection to the input device is not established at this timing.

Syntax AddController(<bstrCtrlName:BSTR>, <bstrProvName:BSTR>,
<bstrPcName:BSTR> [, <bstrOption:BSTR>])

bstrCtrlName : [in] Controller name
bstrProvName : [in] Provider name. Fixed to =" CaoProv.DirectInput".
bstrPcName : [in] Computer name where provider runs.
bstrOption : [in] Option character strings (not used)

2.2.2. CaoController::AddExtension method

Connect an input device. At the time of connection, an input device name is required. Available input device name is obtained by CaoController::get_ExtensionNames.

Syntax AddExtension(<bstrName:BSTR>, [<bstrOption:BSTR>])

bstrName : [in] Input device name
bstrOption : [in] Option character strings (not used)

2.2.3. CaoController::get_ExtensionNames property

Obtain the list of input device names being connected. Input device names obtained here can be used in CaoController::AddExtension.

Syntax <pVal:VARIANT> = get_ExtensionNames (<bstrOption:BSTR>)

bstrOption : [in] Option character strings
pVal : [out] Device name list

Table 2-3 Option character strings of CaoController::get_ExtensionNames

Option	Description
@ForceEffect =<TRUE / FALSE>	Obtain Forcefeedback-supported joystick only (Default:FALSE)

2.2.4. CaoExtension::AddVariable method

Obtain CaoVariable object that obtains input device information. For about available input device names, refer to 2.3.1.

Syntax AddExtension(<bstrName:BSTR>, [<bstrOption:BSTR>])

bstrCtrlName : [in] Variable name

bstrOption : [in] Option character strings (not used)

2.2.5. CaoExtension::Execute method

Execute a command specified by bstrCommand.

Syntax <pVal:VARIANT> = Execute (<bstrCommand:BSTR>, <vntParam:VARIANT>)

bstrCommand : [in] Command name

vntParam : [in] Parameter

pVal : [out] Result value

Table 2-4 CaoExtension::Execute command list

Command	Description	
Start	Execute an effect	P. 7
Stop	Stop an effect	P. 8
LoadEffectFile	Read an Effect definition file.	P. 8

Start

Syntax *object*.Start <Effect ID>

Argument <Camera ID> = VT_I4: Effect number

Return value (none)

Description Start the specified ID's effect. This command immediately returns the process without waiting the completion of effect.
An effect can be started even if another effect is running. Doing so, multiple effects can be run simultaneously.

Stop

Syntax *object*.Stop <Effect ID>

Argument <Camera ID> = VT_I4: Effect number

Return value (none)

Description Stop the specified ID's effect.

LoadEffectFile

Syntax *object*.LoadEffectFile <Path>

Argument <Camera ID> = VT_I4: File path of the Effect definition file

Return value (none)

Description Read the specified Effect definition file.
Effect definition file is created by Force Editor that is provided with DirectX SDK.
Contents of an effect read from a file will be assigned to Effect number starting from 1.
Up to 9 types of effects can be assigned. If a file defines 10 or more types of effects, only the first nine types are registered.
Once this command is executed, the contents of existing Effect number 1 to 9 will be discarded.

2.2.6. CaoVariable::get_Value property

Obtain a value of variables created by 2.2.4. For details about variables, refer to 2.3.1.

2.2.7. CaoController::OnMessage event

When a state of input device is updated, an OnMessage event is issued. At that time, type of event is stored in Message::Number property, input device name is stored in Message::Source property, and the changed data is stored in Message::Value property.

Type of event	Description

0	Connect/Disconnect event (0: disconnect, 1:connect)
1	Button state change event
2	X-axis change event
3	Y-axis change event
4	Z-axis change event
5	X-axis rotation event
6	Y-axis rotation event
7	Z-axis rotation event
8	Additional axis change event
9	POV change event

2.3. Variable list

2.3.1. Extension board class

Table 2-5 Extension board class System variable list

Variable name	Data type	Description	Attribute	
			get	put
@BUTTONS	VT_ARRAY | VT_UI1	Button layout. Upper bits of byte are set while the corresponding button is pressed, and are deleted when the corresponding button is released or not exist.	✓	-
@X	VT_I4	X-axis. In general, this represents the right/left motion of a stick.	✓	-
@Y	VT_I4	Y-axis. In general, this represents the back/forward motion of a stick.	✓	-
@Z	VT_I4	Z-axis. This normally represents the throttle control. If a joystick does not equip this axis, this value will be 0.	✓	-
@SLIDER	VT_ARRAY | VT_I4	Additional two axes. The meaning of this value differs depending on a joystick used.	✓	-
@RX	VT_I4	X-axis rotation. If a joystick does not equip this axis, this value will be 0.	✓	-
@RY	VT_I4	Y-axis rotation. If a joystick does not equip this axis, this value will be 0.	✓	-
@RZ	VT_I4	Z-axis rotation (it is called "rudder"). If a joystick does not equip this axis, this value will be 0.	✓	-
@POV	VT_ARRAY | VT_UI4	Direction control for point of view or alike. This shows the actual position though, the value displayed here is the 100 times of actual angle with the clockwise rotation starting from the North (the front side of a user). In general, the center position is reported as "0xFFFF". If an indicator has only five positions, the value of controller will be one of the followings; 0xFFFF, 0, 9000, 18000, or 27000.	✓	-
@STATE	VT_BOOL	Connection state of a joystick TRUE: connected, FALSE: disconnected	✓	-